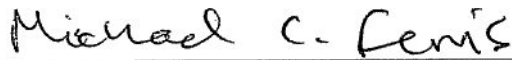

An Interactive GUI Tool for Radiotherapy Treatment Planning

Author:
Laura LEGAULT

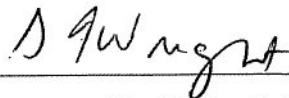
Advisor:
Dr. Michael C. FERRIS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN COMPUTER SCIENCE

UNIVERSITY OF WISCONSIN-MADISON



Dr. Michael C. FERRIS



Dr. Stephen J. WRIGHT

June 1, 2009

Contents

1	Introduction	1
1.1	Cancer Research	2
1.1.1	Treatment devices	2
1.1.2	Terminology	6
1.1.3	Data Generation	8
1.2	Optimization	8
1.2.1	Linear Programming	10
1.2.2	Mixed Integer Programming	11
1.2.3	Nonlinear Programming	12
1.2.4	Convexity	14
1.3	Summary	15
2	A GUI Tool for Treatment Planning	16
2.1	Original Matlab routines	16
2.2	DVH versus dose plot	18
2.3	Introducing an interactive GUI tool	19
2.3.1	Accessing patient data using <code>initdata.gms</code>	21
2.3.2	Getting meaningful and useful results	22
2.3.3	Required programs	22
2.3.4	Interpreting results	22
2.4	Features of the GUI tool	23
2.4.1	Number of iterations	23
2.4.2	Piecewise-linear vs. cumulative normal constraints	24
2.4.3	Regional trade-offs	26
2.4.4	Constraining only one region	27
2.4.5	Additional anchor points	28
3	Technical and Mathematical Details	30
3.1	A cutting plane algorithm	30
3.1.1	Algorithm fundamentals	31
3.1.2	Creating constraining distributions	34
3.1.3	Initial conditions	35
3.1.4	Drawbacks of the algorithm	36
3.2	Interfacing with GAMS	38
3.3	Future work	38
A	Implementation Details	40
A.1	C++ Source	40
A.2	GAMS Source	44
A.2.1	GAMS Models	45
A.2.2	Objective functions	46

List of Figures

1.1	From left to right, a patient with PTV and OAR structures in blue and red respectively, and two fluence maps representing different treatment plans.	3
1.2	A Tomotherapy treatment device.	5
1.3	The Gamma Knife Perfexion treatment device.	6
1.4	The nonlinear function $y = \frac{x}{2} \sin(x)$ from $x = -0.5$ to $x = 4$, which contains two local minimum values.	12
1.5	The convex function $y = (x - 1)^2$, a parabola with vertex (1,0).	15
2.1	Dose volume histograms produced by the Matlab tool, based on solutions from an early model employing the equivalent uniform dose (EUD) technique. In these graphs, the PTV is shown as a blue dashed curve, and the OAR as a solid red curve.	17
2.2	The user interface presented in our tool, including controls for constraining the PTV and OAR, limiting the number of iterations, weighting the volumes, running solves, clearing new solves and saving images.	19
2.3	A comparison of the progress made by the tool after various numbers of iterations using the same constraints. In each figure, the previous iteration's solution is displayed as the lighter lines.	24
2.4	A comparison of the cumulative normal and piecewise linear constraint curves.	25
2.5	A series of the normal curves calculated to fit given PTV points, demonstrating the difficulty of accurately controlling the normal distribution constraint curve.	25
2.6	Demonstrating the effects of weighting on a difficult constraint set - examine in particular the difference in the PTV. Both figures use the same initial solution (shown as lighter curves) and constraints.	26
2.7	Demonstrating the effects of constraining only one of the volumes rather than both. The starting solutions are displayed as lighter curves and are the same in both figures.	27
2.8	Two examples of PTV constraining with a very small vertical difference in the piecewise-linear curve resulting in a large difference in the quality of the solution. The light blue and light red curves show the original solution, and the darker curves depict the current solution.	29
3.1	The size of the violating set of voxels (labeled B) over 10 iterations.	36
3.2	Examples of constraint satisfaction in the OAR with point-wise violations that are not considered in the area-based constraints.	37
3.3	An early version of the tool which displays multiple OAR regions (in dark blue) and their initial dose profiles (in light blue).	39

Chapter 1

Introduction

According to the World Health Organization's (WHO) 2003 World Cancer Report [15], cancer rates are projected to increase by 50% by 2020, resulting in a total of 15 million new cases of cancer in that year alone. In 2000, malignant tumors caused 12% of all deaths worldwide, and most countries attribute nearly 25% of their deaths to cancer. While WHO advocates healthy lifestyle changes and public health action as cancer prevention, projecting that this could prevent nearly a third of cancer cases, the number of cases which are not preventable by lifestyle change or public health action is still a significant percentage of the world population. Cancer will remain a serious problem for the immediate future, even if action is taken, and thus it is very important for researchers to concentrate on this area.

Various sources ([3],[19]) claim that between 40 and 50 percent of individuals diagnosed with cancer will undergo some sort of radiation therapy treatment, rendering it a highly consequential subset of cancer research. The primary issue with radiation therapy is accurately planning a treatment procedure for the delivery device. After imaging has produced a representation of the internal structure of the patient, a clinician will prescribe a particular dose to be delivered to the cancerous areas. Unfortunately, it can be very difficult to non-invasively deliver this dose without causing damage to surrounding organs and other tissues.

The primary treatment planning problem is determining a delivery plan which uniformly doses the cancerous area with an appropriate level of radiation while causing as little collateral damage as possible.

In this introductory chapter of the thesis, we will provide the reader with a preliminary background so that subsequent discussion of tools, methods and concepts in the later portions of the thesis can be accessible and illuminating even to those readers without a formal background in the relevant topics. We assume for the purpose of this document only that the reader has a basic understanding of mathematics including integral and differential calculus, as well as an interest in the methods underlying treatment planning for cancer.

The remainder of this chapter will contain techniques and terminology involved with optimization and cancer research, an introductory discussion of optimization, and the methods used to generate data for our models. The next chapter will include a high-level discussion of the tool constructed during this research, including the original Matlab

inspiration as well as a discussion and demonstration of our tool. The final chapter is intended to be more theoretical in nature, and will contain a technical discussion of the tool's implementation as well as a mathematical overview of the algorithm used and the details of translating the model to the GAMS modeling language.

1.1 Cancer Research

Optimization has been used in many areas ranging from financial and business planning to scientific applications, including a recent increase of applications in the area of treatment planning for cancer patients. Previous work has ranged from classification of cells as cancerous or normal (as in [7]) based on expression data, to the design of many different types of models ([17],[14],[13],[8],[5],[1]) to deal with the problem of selecting treatment schemes for many different types of treatment methods. Our work is concerned with an implementation of one of these model designs.

In this particular section, we will discuss the various methods of delivering radiotherapy to cancer patients, including underlining which methods our tool is currently able to design plans for and which methods will require additional optimization outside of the capabilities of our tool. We also discuss some of the terminology common to discussions of dose planning, and conclude with a brief review of how data is generated for our tool.

1.1.1 Treatment devices

In recent years a number of different methods of delivering external beam radiation therapy have been developed. The devices all rely on effectively the same idea: deliver radiation to a patient through some aperture from an encircling device. The primary differences which we encounter in each type of device include size and shape of aperture, method of generating radiation for delivery, and whether the aperture and/or patient are in motion during the treatment.

While our current research is primarily geared toward one particular method of treatment delivery, we believe that it would be a simple exercise to modify our application to other methods of delivery.

3D-CRT

Three-dimensional conformal radiation therapy is the device for treating cancer for which our algorithm is primarily meant to be used. The process of 3D-CRT begins by creating three-dimensional data sets for each patient via a computed tomography (CT) scan, consisting of the cancerous area and any adjacent normal anatomy. These data sets are then divided into *voxels*, which are grouped into subsets of normal tissue, the *PTV*, and various *OARs* (see below).

The process of planning a treatment for these data involves selecting a subset of discrete angles for the delivery of radiation, choosing the intensity at which the radiation will be delivered, and designing a conformal, bird's-eye-view aperture shape for the radiation dose. The discrete angles available are determined by the particular treatment

mechanism, which consists of linear accelerators using microwaves to accelerate electrons into a small amount of tungsten.

Each dose releases radiation into the nearby human tissue, causing both normal and cancerous tissue to lose their ability to reproduce. Normal tissue can adapt over time, but cancerous tissue is less able. External beam therapy as a whole relies on the capability of normal tissue to retain its vitality even when exposed to radiation.

Any adverse effects in normal or sensitive tissue can be lessened, however, by delivering multiple beams of weaker intensity to the tumor. The superposition effects resulting from these weak beams' combined intensity when they intersect in the tumor create an area of relatively high dose in the cancerous tissue, but they deposit a much lower dose in the normal tissue regions.

The aperture for the radiation beams is created with a series of collimators which surround the beam and shape it to conform (hence *conformal* radiation therapy) to the view of the tumor from the particular angle of the beam. This can be done manually or automatically; however we do not treat this particular problem here.

IMRT

Intensity-modulated radiation therapy allows a more complex dosage profile than 3D-CRT. Where 3D-CRT conforms a uniform intensity beam to the shape of a tumor, IMRT attempts to conform not only its shape but generate dose intensities from each angle to the specifics of the tumor. Determining a dose plan for IMRT is a three step process:

1. The first optimization that must be performed is the creation of a *fluence map*, which determines the varying intensities for each small subset of the tumor silhouette (called *bixels*). These intensities may correspond to density of a particular area of a tumor, presence of sensitive organs between the aperture and the tumor, or a number of other factors. The sum of these fluence maps is subject to the same optimizations as the uniform beams in 3D-CRT; that is, attempting to deliver dose to the proper areas at the prescribed level.
2. Each fluence map is then subject to aperture design - given a set of intensity levels, what set of apertures and intensities from a certain angle will result in a dose that closely approximates the fluence map?

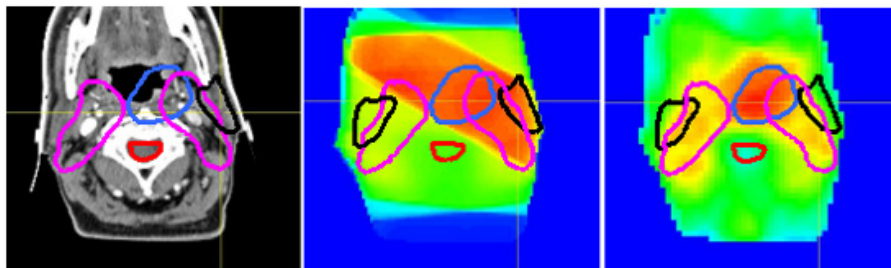


Figure 1.1: From left to right, a patient with PTV and OAR structures in blue and red respectively, and two fluence maps representing different treatment plans.

3. Finally, the sequence of apertures must be chosen with respect to the mechanical constraints of the particular device. Since collimators take time to move, and in consideration of the patient we wish to minimize treatment time, this selection problem strives to choose a sequence of apertures that requires the least time to move from one to the next.

While IMRT has the ability to very precisely conform to the intricacies and oddities of individual tumors, with the gain of customizability we sacrifice a great deal in treatment time. Each angle requires multiple doses, between which the machine must be recalibrated, resulting in a very long treatment time during which the patient must remain still. Also, the treatment planning must necessarily take longer than for 3D-CRT, as the planning process incorporates three steps rather than just one.

IMAT

Intensity-modulated arc therapy, or IMAT, is similar to 3D-CRT and IMRT in that it follows the same idea of delivering beams of various intensity to the tumor through apertures which are conformed to the silhouette of the tumor from the beam's-eye-view, or of approximating the IMRT fluence maps. However IMAT differs from these methods in that it strives to deliver its dose in continuous arcs around the patient's body, rather than from discrete angles. That is, whereas most methods of delivery simply position the device (called the *gantry*) around the body, select a position, choose an aperture, and activate their radiation dose, IMAT delivers a continual dose as the gantry rotates.

This introduces an additional problem in planning treatment schemes - we require not only a selection of positions (arcs rather than discrete angles) and intensities as before, but also a continuous mapping from aperture to aperture along the path of the gantry to maintain the conformal nature of the beam. Since the collimators require time to change position, the more varied the shapes of the apertures are, the slower the gantry will have to move along its arc, inducing a longer exposure time which will lead to a choice of either lower intensity or higher dose. Less change in aperture therefore allows a faster rotation time at a higher intensity.

The primary advantages of arc-based therapy rather than beam-based therapy is the elimination of the "hotspots" that appear where beams overlap outside of the tumor, which can result in a star-like shape, and the speed of delivery of dose. Because the doses are delivered continually rather than discretely, the device does not have to be recalibrated after every angle. However, the complications inherent in the interdependencies of aperture shape, gantry rotation speed, arc selection and intensity mean that while this is a potentially highly effective treatment method, it is computationally much more demanding than the previously outlined approaches.

Tomotherapy

Tomotherapy takes IMAT to the next logical step. Rather than simply delivering doses in flat arcs, Tomotherapy creates a helical dose path by rotating the gantry while moving the patient slowly through the device [9]. The primary advantage of this method of treatment is that the entire tumor volume can be treated at the same time, without having to realign the system or the patient.

Tomotherapy also makes use of *image-guided radiation therapy* [10]. Like the other methods of treatment, IGRT makes use of initial imaging through CT, magnetic resonance imaging (MRI) and positron emission tomography (PET) to locate the tumor and guide the initial treatment planning. However unlike the methods discussed above, IGRT makes additional use of frequent two- and three-dimensional imaging during the treatment process itself to adjust and refine the initial treatment plan, both in intensity and in shape.



Figure 1.2: A Tomotherapy treatment device.

The process of IGRT can be applied in addition to any of the methods outlined here. While we do not address replanning using IGRT in this thesis, our tool could also be used in this regard.

Stereotactic Radiotherapy

Stereotactic radiotherapy is a method primarily designed for the treatment of tumors and various vascular ailments in the brain. The patient wears a device directly attached to the body, designed to keep the afflicted region of the body completely still. After this device is attached (under local anaesthesia), three-dimensional data sets are created as in 3D-CRT. After the treatment has been planned (which must occur while the patient is still wearing the device), therapy begins immediately. Because of the immobility imposed by the devices attached to the patient, stereotactic radiotherapy is a highly accurate method of treatment; the clinically designated PTV and OAR volumes need not include much additional margin to account for movement.

Many small doses of radiation are delivered from angles around the body, which converge on the cancerous region, delivering a very low dose of radiation to the normal, healthy tissue but leaving a high amount of dose in the tumor. This process is repeated at low dosages many times to avoid damage to the brain, and may aid in the treatment of inoperable tumors.

Stereotactic radiotherapy differs from 3D-CRT in that the apertures through which the radiation passes are very small and circular, rather than being conformally shaped to the silhouette of the tumor. There are two primary devices for the delivery of stereotactic radiotherapy:

Gamma Knife Likely the most popular delivery device for stereotactic radiotherapy is the Gamma Knife, which is used for targets inside the brain. The Gamma Knife delivers dose via a continually decaying radiation source, focused through tiny apertures in various types of helmets, which are attached directly to the patient. These helmets vary in their aperture sizes (larger holes for larger targets) and in their position relative to the patient's head, which serves to focus the convergence point at specific locations.

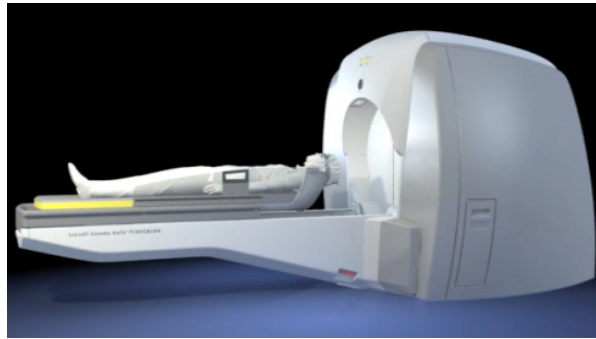


Figure 1.3: The Gamma Knife Perfexion treatment device.

Linear Accelerator Other forms of stereotactic delivery (for example, the CyberKnife [16]) use a more standard linear accelerator, which functions in the same way as with 3D-CRT. As with all stereotactic therapy, these delivery devices make use of a helmet for focusing rather than a conformal aperture.

1.1.2 Terminology

Voxels

A *voxel* is, quite simply, a small cubic subset of the total volume. Where image processing will divide a two-dimensional image into small square pixels, clinicians also find it useful to make the same divisions in three-dimensional space. The difficulty in dividing a volume into voxels is the same as dividing a plane into pixels: to balance the processing gains which arise from dividing a continuous space into a number of small, discrete spaces with the coarseness of the resulting image. While it may be very easy to do image processing on a picture with only 16 pixels, much of the meaning is lost. In treatment planning, if our voxels are too large, we may end up neglecting to protect sensitive areas or failing to plan a dose to a sufficiently small tumor.

PTV

The *planning target volume*, *principal tumor volume*, *primary treatment volume*, or PTV, is the fixed subset of voxels determined by a clinician to contain a cancerous tumor for treatment. This subset is chosen to account for all possible uncertainties which the treatment may incur; including internal organ shifting, patient movement during treatment, and inaccuracies in detection. The PTV should have a high probability of containing the tumor for the entire treatment.

Two subsets of this volume not directly referenced in this document but still of importance to treatment planning are the GTV (*gross tumor volume*) and the CTV (*clinical target volume*). The GTV is the subset of voxels containing the highest density of tumor cells, and can be thought of as the main body of the tumor. However, since this may not account for some cancerous cells outside of the high density regions, clinicians typically add a margin of error to the GTV to create the CTV, which will hopefully contain all cancerous cells which require treatment.

OAR

Along with the normal tissue and one (or more) PTVs, we also must give special consideration to any organs or tissue areas which are particularly vulnerable to the radiation delivered in our treatment plan. These areas we term the *organs at risk*, or OAR. In our treatment planning, we pay special attention to the dose delivered in these regions; while our goal is ultimately to eliminate all cancerous cells in the patient's body, we wish to cause as little collateral damage to unaffected, vital organs as possible.

Dose Plan Evaluation

Once a treatment plan has been designed, it is important to be able to evaluate the quality of that treatment plan. This is a function primarily of two factors: how well the plan emulates the prescription from a clinician, and how well it avoids damaging non-cancerous tissues; other factors such as uniformity of dose are also considered in some cases. While it's possible to achieve this in a purely numerical fashion, there are several common graphical methods which translate the data into a more human-readable format. Here we describe two of the more common methods, the dose plot and the dose-volume histogram.

Dose Plot Probably the more intuitive of the two methods, the dose plot is a heat-level representation of radiation within the body, split into a series of slices of the relevant volume (see Figure 1.1.1). Beams of dose appear as just that - wide rectangular areas of heat that dissipate as they make their way through the body. Organs are marked with simple curves, so that an observer can determine how well the plan's distribution matches the prescription in the PTV, and ensure that no important sections of the OAR are being overdosed. Most usefully, however, the dose plot allows the observer to immediately detect any streaking introduced by the treatment plan, and all local effects are clearly visible.

Since dose levels are presented as variations in color, however, it is difficult to tell from a dose plot alone how well the actual levels of dose conform to the prescribed levels. While we maintain the dose plot in our arsenal for evaluating regions receiving dose, for our more numerical analysis, we turn to the DVH.

Dose Volume Histogram The *dose volume histogram*, or DVH, graphs the levels of dose in various areas of the body using a histogram scheme (see Figure 2.1). The x -axis corresponds to the level of radiation dose received by a voxel, and the y -axis corresponds

to the percent of volume of a particular area receiving that dose. An ideal DVH will have doses of zero for 100% of normal tissue and the OAR, and a tight bell curve in the PTV around the prescribed level of dose. Typically, however, curves will be much more gradual and doses will be more varied within a particular volume.

The dose volume histogram that we use in our application is actually a cumulative DVH; that is, the y -axis corresponds to the percent of volume receiving *at least* that level of dose. In this situation, an ideal cDVH will have again doses of zero for 100% of normal and OAR tissue with 0% of those volumes receiving anything higher, and a vertical line in the PTV from 100% of volume to 0% of volume at the prescribed level of dose. Again, in the typical case these will have more gradual curves, implying a more varied dose level throughout the volume.

Since we use only the cDVH in our application and not the standard DVH, we will conventionally refer to the cDVH as the DVH for the rest of the document.

1.1.3 Data Generation

First, we must clarify what our program is *not*: a complete replacement for a clinician. In fact, we rely quite heavily on the input of a clinician for preprocessing the data, evaluating the quality of our solution, and providing any additional constraints required to improve the solution.

The process begins with a raw data set of three-dimensional information collected from scans of the patient, with various areas that a clinician has designated as normal, PTV, or OAR. This data is pre-processed through a Monte Carlo simulation, which generates a model of how much radiation would reach and deposit in a particular area based on a variety of factors. We then take this processed data and load it into our own model, which attempts to choose a dose scheme for our particular method of radiation therapy, 3D-CRT, so that the resulting dose distribution in the patient's body is as close a match to that prescribed by a clinician as possible, while carefully avoiding any sensitive areas.

The data used to develop our program is courtesy of D.M. Shepard and his group at the University of Maryland School of Medicine.

1.2 Optimization

The underlying mathematical concept of the present paper resides in the field of *optimization*. For the benefit of readers who have only a background in radiotherapy dose planning or cancer research, we have included the following section as a primer on the concepts we will use later in this document. We also hope that it will prove helpful to those readers with a strong background in optimization as either a quick refresher in the relevant areas or in recognition that optimization terminology is not entirely universal.

Problems for which optimization is a useful tool can range from finding the shortest commute to work, to creating the most lucrative stock portfolio, to determining the ideal ratio of lemon to honey in one's tea; but above all an optimizer wishes to improve the current result given a certain set of limitations. We cannot create new streets to drive on or remove other drivers from existing streets (much as we might like to), our portfolio is subject to the risks of the market, and our teacup has only a limited volume.

Treating these situations mathematically, we say that to optimize a problem is to minimize (or maximize, though for simplicity we consider only minimization in this section) some *objective function* subject to *constraints* on its *variables* [12].

- *objective function*: This is an expression that represents the situation which we want to minimize. Thus we need to create some quantitative measure of the situation – amount of time or distance of commute, risk vs. reward in our portfolio, or some measure of the ratio of sour to sweet in our tea – which we can drive to its minimum (or maximum) in order to create an optimal situation. This expression is written $f(x)$.
- *constraints*: The constraints of a problem are any physical, practical, or theoretical limitations our model must take into account when attempting to drive the objective to its minimum. If no constraints are present, minimizing $f(x)$ may be a simple matter of taking derivatives, as discussed in Section 1.1.4. But in most situations, models may need to take into account constraints ranging from traffic patterns and existing streets to the natural movement of the stock market or just how much room we have in our teacup. These constraints are also expressed as functions, written as $c_i(x)$, where i is some index from 1 to the number of constraints.
- *variables*: These are the unknown values contributing directly or indirectly to our objective function measure which we shall change in order to arrive at an optimal objective function value. Variables may include which streets to take and what speed to drive, which stocks to purchase and which to sell, or the amounts of lemon and honey to add to our tea. These variables we refer to here as simply x .
- *parameters*: Parameters, as we shall use the term here, are pieces of data specific to each instance of a problem which relate to the constraints (but which may vary between instances of a problem). These may include the length of various roads, average speeds of traffic, or prices of stocks at a given time. Parameters allow single models to be applied to many different situations: our tea tasting model could be applied to a different person with a larger teacup by simply updating the parameters for volume and taste preferences.

The process of determining the composition of the function, as well as the number and makeup of constraints, variables and parameters, is called *modeling*, and the resulting problem will be referred to from now on as a *model*.

A general mathematical optimization model can therefore be expressed as follows:

$$\min_x f(x) \text{ subject to } c_i(x) \geq 0 \tag{1.1}$$

Note however that the constraints in equation (1.1) are only inequalities (to change a \leq to a \geq , recall that multiplying by -1 will flip this sign), and not strict. Equality constraints *are* allowed, and are in fact used in the model described later in this paper. Strict inequalities, however, can cause difficulties when solving the model - because the real numbers allow one to get infinitely close to a number without actually attaining it, a solver that approaches the constraint $x < 1$ from the value $x = 0$ may keep increasing indefinitely and never terminate its search.

A number of models have been proposed in the past few decades for determining a treatment plan that accurately delivers a uniform dose of radiation to the tumor and avoiding sensitive areas and unnecessary damage to other healthy tissue. Various optimization approaches have been used, ranging from linear programming to a mixed integer problem to various quadratic and more complex approaches, all of which have a range of applications, quirks and disadvantages. We now briefly describe the fundamentals of each type of approach and discuss in greater detail their relationships with the treatment planning problem.

We wish to emphasize that there is no one best solution to every treatment planning problem. Each of the types of models described here has been used with results ranging from merely acceptable to spectacular. The advantages and disadvantages which we enumerate here are not then intended to show the inferiority of one algorithm over another; rather, we merely wish to demonstrate our reasoning in choosing the algorithm which we use in our tool.

1.2.1 Linear Programming

Linear programming is a subset of optimization that makes the requirement that the objective function and all constraints be of a linear form. The advantage of linear programming is that the area of potential solutions (which we shall call *feasible*) is a convex polyhedron, so any maximum value that we find is guaranteed to be the *global* maximum (the true maximum over all possible solutions of the problem) rather than just a *local* maximum (the maximum value in a particular range of solutions, not necessarily over all solutions); and analogously with a minimum.

A linear program is unsolvable in only two situations: if it is *infeasible*, or if it is *unbounded*. An infeasible program implies that two or more constraints are contradictory – for example, the combination of $x \geq 2$ and $x \leq 1$ would yield an empty feasible region. An unbounded program implies that the feasible region is unbounded in the direction of the objective – for example, when maximizing an objective, if the only constraint is $x \geq 2$ and increasing x increases the objective value, then the best value of x is ∞ and the problem is therefore unsolvable. Note that these conditions also apply to other forms of optimization models.

The simplicity of linear programming, when applied to radiotherapy treatment planning, is a double-edged sword: the human body is not, in any way, “simple”. It becomes very difficult to restrict one’s algorithms to be solely linear while taking into consideration the complexities inherent in the human body. However, the algorithm that we have chosen to implement in this tool does exactly that, by making direct use of the abstractions provided in the form of the dose volume histogram (DVH).

A linear model can make use of either nonlinear preprocessing in the form of a Monte Carlo simulation of dose distribution within the body or a great deal of data from patient scans to construct a dose matrix, \bar{D} , which is then combined with an angle weight to determine the dose D_i to a voxel i as follows:

$$D_i = \sum_b w_b \bar{D}_{b,i}$$

When combined with the range constraints $L_i \leq D_i \leq U_i$ for some lower bound L_i and upper bound U_i , we impose a linear objective function such as

$$obj = \sum_{i \in PTV} \gamma_i + \sum_{i \in OAR} D_i$$

where γ_i is used to model the difference between the delivered dose level D_i and the prescribed dose level P_i for the voxel i ; that is, by adding the constraints

$$\gamma_i \geq D_i - P_i$$

$$\gamma_i \geq P_i - D_i$$

This is in fact a simplified version of the objective function used in our model.

1.2.2 Mixed Integer Programming

A *mixed integer program* (MIP) employs use of the discrete integer variables $\mathbb{Z} \subset \mathbb{R}$ as well as variables in the real numbers. By using this scheme, one can effectively introduce boolean (true/false) constraints rather than strictly numerical constraints by restricting the range of a variable to be $\{0, 1\}$. The constraint and objective functions are still required to be linear in construction, however. The difficulty with MIP models arises in solving them - generally MIP problems are classified as NP-hard¹.

This integer-variable strategy can also be applied in situations where discrete answers are required – most notably in problems of personnel management for airlines, where one employee can only be on one plane at a time; or for vehicle routing, where a single truck cannot follow multiple roads.

MIPs are at once intuitive for use with the 3D-CRT treatment technique, employing a certain number of discrete delivery angles with a single dose level at each angle. Since the amount of time required to provide a dose to a patient from each individual angle is nontrivial, it may be advantageous to constrain the number of angles from which dose can be delivered to a small number, known to be achievable in some manageable time, and provided as a parameter based on the individual situation (a small child, for example, may be able to tolerate a smaller number of discrete angle treatments than an adult).

For angle weights w_i and a vector \bar{w}_i constraining the maximum weight allowed at angle i , we can introduce a binary variable y_i representing which angles i we are allowed to use:

$$w_i \leq \bar{w}_i y_i$$

and the resulting set of variables y_i will enumerate the angles which have been used in the current solution. Note that these constraints would be used in addition to those outlined in the previous section.

However, as we have just mentioned, the solution of MIPs is classified as NP-hard. This may prove prohibitive when the time of solution is of importance. Also, MIPs are subject to the same constraint as linear programs, in that the objective and constraint

¹Nondeterministic polynomial-time hard (NP-hard) problems are at least as difficult as problems which are considered NP (nondeterministic polynomial-time). These problems are, intuitively, those for which ‘yes’ answers can be **verified** quickly and simply, though finding a **solution** may be quite difficult.

functions must all be of linear form, so aside from the ability to constrain the number of angles from which dose can be delivered, they offer very little additional functionality over LPs.

1.2.3 Nonlinear Programming

Nonlinear programs (NLP), like linear programs, allow all of their variables to move in the space of real numbers, but their constraint and objective functions may have exponents other than 1. They may also make use of logarithms, exponential functions, and other nonlinear mathematical functions. These models have the advantage of flexibility - because they are not restricted to using only linear functions, they can much more closely model the real complexities of life and more accurately determine a solution to their model.

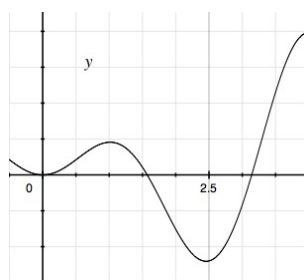


Figure 1.4: The nonlinear function $y = \frac{x}{2} \sin(x)$ from $x = -0.5$ to $x = 4$, which contains two local minimum values.

Unlike linear programming, however, the space of feasible solutions is not guaranteed to be convex, since the functions used in the program themselves may not be convex, and may in fact have multiple solutions which an automatic solver will recognize as “minimums”, as in figure 1.2.3. If a model were to contain the function $y = \frac{x}{2} \sin(x)$ as its objective function with a starting point such that $-0.5 \leq x < 1$, in order to find a smaller value of the objective, a solver with only local awareness would move in the decreasing direction toward $x = 0$. Finding a proven global minimum is an open problem in NLP.

A simple example of an NLP objective function for use in treatment planning is simply an extension of the linear objective. Rather than introducing additional constraints and variables to linearly model an absolute value, we instead consider the squared distance between the dose level D_i and the prescribed dose P_i :

$$obj = \sum_{i \in PTV} (D_i - P_i)^2 + \sum_{i \in OAR} D_i$$

However, we are not limited to these simple formulations. NLPs represent the most flexible of our modeling options, and for this very reason have been used in a number of treatment planning models. Nonlinear constraints can flatten the dose profiles to a more uniform level in various areas of the dose plot, or make use of a simulated annealing model to simultaneously optimize the shapes apertures and intensities of beams. These two types of constraints, while hardly representative of the entire field, are discussed in more detail here:

Equivalent Uniform Dose

The concept of Equivalent Uniform Dose (EUD) [11] was designed to summarize and report inhomogeneous dose distributions for comparison over a series of studies. Because of the wide variety of variables in dose delivery and planning, including a wide variety of normalization criteria that can be applied to the dose itself, until EUD was introduced there was a risk of flattening the dose-response relationship and missing the finer structure of the trial data. Any two doses, then, are considered to be equivalent if they cause the same radiobiological effect, regardless of the actual structure of the dose itself.

The target area is generally assumed to be composed of a large number of independent clonogens (cloned cells), and the radiobiological effect is given by the expected number of surviving clonogens. By the methods of EUD, then, two radiation doses are equivalent if their expected number of surviving clonogens are equivalent.

The equation for approximating EUD was originally suggested by Niemierko [11] as

$$\text{EUD} = \left(\frac{1}{N} \sum_i D_i^a \right)^{\frac{1}{a}}$$

where N is the number of voxels in the tissue area and D_i is the dose to the i th voxel. The value a , a tissue-specific parameter which serves as the exponent on the dose and the inverse exponent on the EUD itself, determines the type of approximation of the EUD. For $a = \infty$, EUD is the maximal dose to the tissue area; $a = -\infty$ gives EUD as the minimal dose; $a = 1$ gives EUD as the arithmetic mean; $a = 0$ gives EUD as the geometric mean.

The objective function for EUD was proposed by Wu et al ([20]), and is a product of subscores f_j for each tissue area. The formulation of this score varies based on the goal of the optimization - minimization of dose for normal tissues or organs at risk, and prescription attainment for tumors. This is done by setting the value of EUD_0 so that for the target volume, EUD_0 is a desired dose, but for normal structures or organs at risk, it is the maximal tolerable uniform dose. The value of EUD_0 would be determined experimentally.

A potential advantage of EUD is its convexity property. The EUD function above is convex for $a \geq 1$ and concave for $a \leq 1$. Since the normal tissues generally use a value of a that is greater than 1, and the objective is to minimize the dose to normal tissue, the convex function is being minimized. For target tissue, generally a is much smaller than 1, and dose is usually maximized, so we are maximizing a concave function. These convexity properties would help to speed optimization and guarantee globally optimal results.

Direct Aperture Optimization

Direct aperture optimization (DAO), as proposed by Shepard et al in [18], involves a direct optimization of the shapes and weights of apertures using a simulated annealing algorithm. This is a replacement for more traditional two-step process of optimizing first the weights at each angle and then performing a leaf-sequencing algorithm, as in the algorithm which we use in our tool. The advantage of this is the elimination of

the second step, which to a certain extent could provide greater overall control of the treatment planning; in our tool, the second step is not treated at all, but rather left to other resources as a time consideration.

The simulated annealing portion of this method uses the Boltzmann cooling schedule

$$P = 2B \frac{1}{1 + e^{\log(n_{succ}+1)/T_0^{prob}}}$$

(where B is the initial probability of acceptance, T_0^{prob} is the rate of cooling, and n_{succ} is the number of accepted changes so far) to determine the probability of accepting a change in leaf position that does not decrease the value of the objective function. By allowing the objective function to increase, the algorithm strives to eliminate the possibility of locating a local minimum while missing the global minimum.

As in the algorithm which we employ, this method uses DVH constraints to penalize overdoses and underdoses. The objective function in DAO is evaluated during the optimization after each change in a leaf position or aperture weight - but not if the change is rejected, which is a standard feature of any simulated annealing application.

One of the primary advantages of DAO is the wide range of possible intensity levels N per beam direction, $N_n = 2^n - 1$, where n is the number of apertures. While this means that if three apertures are used, only seven levels are possible, if six apertures are used, this number balloons to 63. This allows for a greater level of detail in each individual beam.

Disadvantages of NLP

As with MIPs, these non-linear models can prove difficult to solve, and time considerations may render these models less useful than a model which can be solved quickly. More importantly, if the model is not convex as described below, we may not be able to guarantee that the solution found is in fact the global solution. One of the techniques common in NLP to discover the global solution in a non-convex program is to solve the problem multiple times with random starting points, but when dealing with a solution that is already potentially prohibitively slow, multiple solutions may be completely out of the question.

While certain algorithms take the issues of convexity into account (as described in EUD), the complexity of the functions employed in the models may be prohibitive to efficient and effective implementation. In our early work with NLP, including the two models outlined above and several other formulations not included here, we found results to be inconsistent or frustratingly slow; whether due to imperfect implementations or inferiorities inherent in the models was unclear. For this reason, we returned to linear programming for our implementation.

1.2.4 Convexity

The concept of convexity as it relates to optimization is crucial when it comes to not only solving a problem but also evaluating the quality of a solution. With a strictly convex nonlinear function, finding the minimum value is simply a matter of setting the

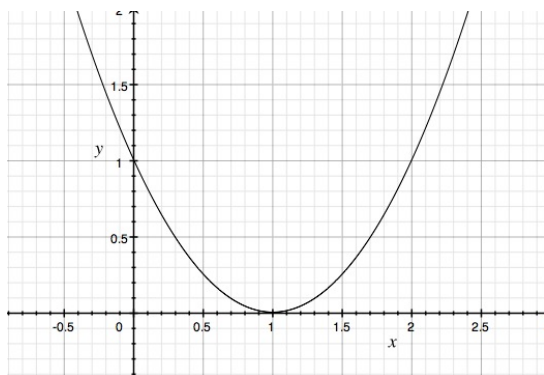


Figure 1.5: The convex function $y = (x - 1)^2$, a parabola with vertex $(1,0)$.

derivative of the function to zero and solving for the variables. For example, the function $y = (x - 1)^2$ describes the parabola seen in Figure 1.2.4. To find the minimum value of this function, we simply take the derivative of the function with respect to x :

$$\frac{\partial}{\partial x}y = 2(x - 1) = 2 - 2x$$

and set this expression equal to zero to find that the minimum value of our function occurs at the point where $x = 1$, which is easily verifiable by inspection of the image. However, with a non-convex function, there may be multiple points at which the derivative is zero, or the point at which the derivative is zero may be the maximum rather than the minimum value. In these situations, there are a number of algorithms available to find the solution, but based on the characteristics of the function, they may be very slow (undoubtedly slower than our single-step solution for an unconstrained convex function). If the function contains more than one point at which the derivative is zero, the algorithms may also report a local maximum rather than a global maximum, resulting in a non-optimal termination.

Because of these important characteristics - speed of solve and a guaranteed global solution - our research implements a convex approximation of the relevant functions, which may or may not be advantageous (see discussion in chapter 3).

1.3 Summary

At this point, the reader should have at least a preliminary understanding of various types of external beam therapy and their uses in radiation treatment, as well as some intuition regarding optimization models. We hope that we have synthesized these two topics sufficiently so that the reader will also have a sense of the role optimization plays in the planning stages of radiation therapy.

We have presented this section for the purposes of making this thesis as accessible as possible to readers in both the areas of optimization theory and cancer treatment planning research, without presuming a background inclusive of both fields. Even if the reader is well-versed in both areas, we hope that this has provided a useful review of topics relevant to our work as discussed in the next two chapters.

Chapter 2

A GUI Tool for Treatment Planning

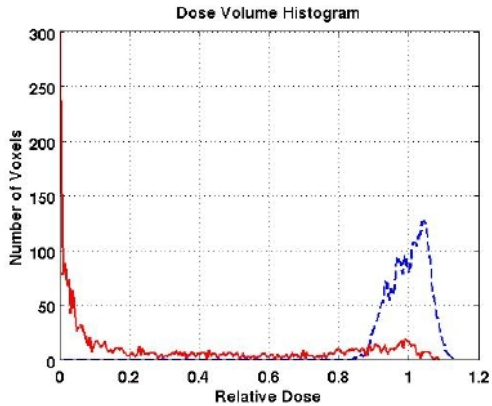
It is a readily apparent and accepted fact that most human beings can derive much more meaning out of an organized and well-executed graph than from well-ordered numerical data. Magnitudes of variance, average values, trends, and many other characteristics of data that may be completely lost to the average human in a jumble of numbers become immediately apparent even to those who lack the training and language to formally describe what they see. For this reason, nearly every publication that can include graphs, plots, and charts of their data rather than a list of the actual numerical data.

This reasoning has motivated scientists for hundreds, even thousands of years to design visualizations of the data they collect, and radiotherapy treatment planning has proved no exception. The complexity of data involved and the importance of designing a high-quality plan under quite possibly severe time constraints mean that it is crucial to eliminate any overhead possible, including first and foremost the overhead of understanding and analyzing the quality of a proposed solution. Various types of visualizations have been proposed, ranging from three-dimensional heat graphs of the areas of the body affected by radiation to abstracted histograms, completely divorced from anatomical representation.

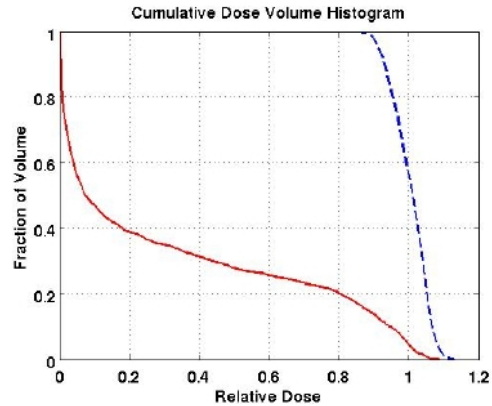
In this chapter, we discuss the advantages of enhancing visualization with an interactive element rather than using graphs only as a *post hoc* analysis tool. We also highlight a number of the features of the tool we have constructed, and hope that these discussions will prove illuminating to the user.

2.1 Original Matlab routines

When we originally began exploring the various types of optimization models that had been developed, our primary tool for visualizing their outputs was a collection of pre-existing routines in Matlab. Similar to a subset of the routines available in the CERR package from the Siteman Cancer Center at Washington University in St. Louis [2] but developed independently at the University of Wisconsin [6], these routines take as input the existing data compiled from scans of the patient, as well as a listing of the nonzero beam weights and the numbered angles from which they will be delivered, and provide a small variety of graphs for visualization. (Note that these routines have been set up specifically for a single device and for application to any other treatment device would



(a) The dose volume histogram corresponding to a particular solve.



(b) The cumulative dose volume histogram of the same solve.

Figure 2.1: Dose volume histograms produced by the Matlab tool, based on solutions from an early model employing the equivalent uniform dose (EUD) technique. In these graphs, the PTV is shown as a blue dashed curve, and the OAR as a solid red curve.

require modification.)

For ease of use we set up small programs which received beam weight and angle input from our external models (coded not in Matlab but in a scripting language as part of the GAMS system - see below) either manually as text or via a proprietary file format called the GAMS Data eXchange (gdx) using an in-house interface between GAMS and Matlab. Using this input as well as the patient scan data, the Matlab routines calculated dose levels to each voxel and created dose volume histograms, cumulative dose volume histograms, and dose plots for each set of angles and weights provided.

In figure 2.1, a dose volume histogram and its corresponding cumulative DVH are shown. These images display a solution which is not of a particularly good quality, but that can easily be determined from these graphs - in (a), the curve of the PTV ranges from nearly 0.9 to 1.1, and in (b), very little of the PTV's curve follows the idealized prescription level line at $x = 1$. What we wish to do is tighten the range of the curve in (a), or cause the curve to become more vertical in (b), but in order to do so, we must alter models outside of the visualization routine.

Since these utilities were only useful for *post hoc* visualization, any observations or insights gained from the resulting graphs had to be translated numerically, taken outside of the program, entered as additional constraints into the models, and reloaded into Matlab for verification. Since this seemed an inefficient and less effective use of the visualization abilities than could be offered, our decision was to incorporate the visualization, modification and resolving steps into one interactive GUI tool.

While the existing visualization routines in Matlab were motivational during the design phase of our tool, we chose not to use Matlab itself for the implementation of our tool for two reasons. First, we wished the plots to be directly interactive - that is, we wanted the ability to place constraints directly on the graph and see, on the same graph, the results of adding those constraints to the model. Lacking an easy way to implement this in Matlab, we turned elsewhere. However, another advantage of eschewing Matlab has

been the reduction in proprietary code - while we are still using a proprietary solver, the rest of our code is freely available.

2.2 DVH versus dose plot

The dose volume histogram and the dose plot, two of the primary visualization methods for radiotherapy treatment plans, are described in detail in Section 1.2.2. Each has its advantages and disadvantages. In this application we have chosen the DVH rather than the dose plot to represent our data for a number of reasons, including the following:

1. **Numerical advantages.** The dose plot offers useful intuition regarding relative dose levels in various voxels. However, what it does not offer is an intuitive, self-contained comparison between actual dose levels and intended dose levels. While it would be possible to achieve this comparison by presenting a prescription image alongside a current dose plan image, the comparison is a matter of differing color levels, which could prove difficult even to a user with normal vision, to say nothing of a color-blind user. In the DVH, judging whether doses are close to their prescription is a simple matter of examining the position of a curve.
2. **Ease of interpretation.** A prescription DVH is very easy to produce - an OAR curve which exactly follows the y -axis, and a PTV curve which follows the line $y = 1$ to the x value of the prescription, at which point it makes a right angle and follows $x = \text{dose}$ to the x -axis. “Good” treatment plans are those which most closely approximate this prescription; “bad” plans will exhibit much different behavior. A dose plot, however, may appear “good” to an observer, but in fact have an unacceptably high amount of variation within the tumor volume or affect a sufficiently large number of OAR voxels so as to be toxic to the patient, all of which must be judged subjectively, where definite lines can objectively be drawn on a DVH to prevent this sort of behavior.
3. **Global view.** The primary disadvantage of a dose plot over a DVH is that due to the three-dimensional nature of the dose plot, only small slices of the target volume can be viewed at once, making a full, global view difficult to conceptualize. The DVH, however, describes the dose levels in all voxels throughout the entire measured volume, immediately presenting a global view of the dose profile.
4. **Ease of implementation.** While a dose plot does present an intuitive view of what happens inside the body, it is inherently difficult to render - while lines and curves are much easier. It is also difficult to place meaningful constraints on a dose plot, aside from specifying further areas to receive certain levels of dose. Our particular model, being of probabilistic nature (see Chapter 3), in fact does not constrain the areas in which underdosed or overdosed voxels are allowed; it merely constrains the number of allowed under- or overdoses in the entire volume, making a DVH much more intuitive and appropriate for the constraints.

We do not wish to imply, however, that the DVH is completely without its disadvantages. Typically, DVH constraints (called first order stochastic dominance constraints) have

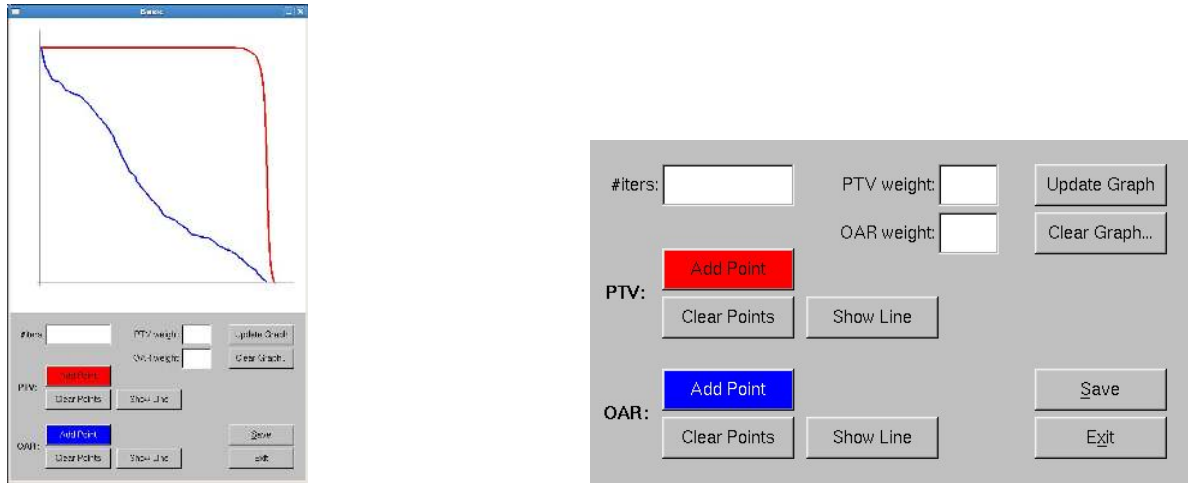


Figure 2.2: The user interface presented in our tool, including controls for constraining the PTV and OAR, limiting the number of iterations, weighting the volumes, running solves, clearing new solves and saving images.

been much more difficult to model - and more importantly, more difficult to enforce. Also, due to the global nature of the cumulative DVH in particular, it becomes necessary to model and enumerate *all* voxels in the volumes, which potentially adds a good deal of overhead to our time considerations.

2.3 Introducing an interactive GUI tool

This tool, whose inspiration is outlined in the previous section, is intended to be a device which consolidates the process of visualizing potential plans for radiotherapy treatment, translating desired adjustments accurately to additional model constraints, resolving the model with those constraints, and redisplaying the updated plan. For simplicity we have chosen a single type of graph to display (the cumulative dose volume histogram, or DVH) and a single underlying model (the Ruszczyński algorithm), though this process is undoubtedly extensible to other models and visualization techniques.

On startup, the user will observe that our tool’s window has two portions: an upper graphing area, and a lower control box. When the tool is first started up, the graphing area displays an initial solve of the current model. This initial solve is accomplished through use of the *initial.gms* model, as described in Section 2.3.1.

After starting up, the user is presented with a number of options as shown in Figure 2.2; of primary concern are the red and blue buttons marked “Add Point”. Clicking either one of these buttons and moving the cursor over the graph area of the window will change the cursor into a crosshair. The user can then place a point on the screen by clicking. What the point signifies depends on which button the user selected:

- PTV (red): The red Add Point button drops a small, red, upward-pointing triangle onto the graph in the closest allowable position to the point that the user has clicked (see below). This point represents a constraint which has been added to the model for further solves - the user now wishes that the red PTV curve have a y value

higher than this point's y value at the same x value; in terms of the model, that a higher percent of PTV voxels (the y value) receive at least that amount of dose (the x value). The user will observe that the first time this button is clicked (and before the user has placed his point), a point is added at (0,1). This is to give the curve an anchor point, though additional user-placed anchor points are also recommended (see Section 2.4.5).

- OAR (blue): The blue Add Point button drops a small, blue, downward-pointing triangle onto the graph in the closest allowable position to the point that the user has clicked. Similarly to the PTV, this represents a constraint added to the model, but rather than constraining the PTV's curve to be "over" this point, we rather constrain the blue OAR curve to be "under" it. That is, a lower percent of OAR voxels receive that amount or more of dose. The first time this button is clicked, two points are added: one at (0,1) and the other at the maximum OAR dose on the x -axis. The point at (0,1) is an anchor point as with the PTV; the point at (max,0) is a suggested anchor point.

We note here that all points, including anchor points, can be moved after they have been placed, though with anchor points we do not recommend this. For the mathematical details on these tail-shaping constraints, see Chapter 3.

The points placed using these buttons are subject to their own constraints. Since by definition our DVH curves are non-increasing, we require the points placed using these buttons to also mimic that non-increasing behavior. That is, if a user attempts to place a point at an illegal location, the program will simply place a point at the closest allowable location to that point. For example, if a user were to attempt to place a point above the line $y = 1$ (representing 100% of voxels), a point would simply be placed at that x value but with a y value of 1. This is actually particularly useful when placing constraints on the PTV; see Section 2.4.5.

We also wish to note that it is not necessary to place points of both types on the graph; for further discussion of this, see Section 2.4.4.

After the points have been placed on the graph to the user's satisfaction, the next buttons of note are the "Show Line" buttons under both PTV and OAR. Clicking these buttons will "connect the dots" of either the PTV or OAR points on the graph with a piecewise-linear green curve. This curve, even moreso than the points which signify its anchors, describes exactly the constraints which will be imposed - the PTV and OAR curves should not only be constrained to be over or under the points, but over or under these green constraint curves as well. If the constraint curves appear more stringent than required by the situation, additional points can be placed (as long as they maintain the non-decreasing characteristic of the curves) and the green curve can be re-displayed by simply clicking the button again. Note that these green constraint curves are NOT required to be convex.

Once the points and curves are set, there are two more options to consider before a resolve is begun. The first of these options is the iteration count, found in an input field in the upper left corner of the control box. If the user does not choose to enter a value, the resolve will run for a default of 20 iterations; the user may enter any positive value. (The merits of various iteration numbers are discussed in Section 2.4.1.) The second

relevant option includes the two weighting inputs for the PTV and OAR. These values default to 1, giving equal weight to the PTV and OAR in the solve, but in situations where a tradeoff is required to attempt constraint satisfaction, weighting will affect which constraints are solved first. (For further discussion, see Section 2.4.3.)

At this point, the constraints have been set, an iteration limit has been chosen, and a set of weights has been input. At this point, the user is ready to resolve, which is done by clicking on the “Update Graph” button in the upper right corner of the control box. At this point, the program initializes a GAMS solve, using saved data from the initial solve as a starting point, and including the additional constraints as specified with the green curve. If the program was started from the command line, the user can observe the GAMS output in the console. Each iteration as specified by the program corresponds to a full GAMS solve; for details of the modified constraints included in each iteration, see Chapter 3.

The GAMS runs will end either when the iteration limit has been reached, or a solution has been found with an error of less than 1×10^{-5} . At this point, the window will refresh with the new solution delineated in red and blue as before, and the old solution in light red and light blue. If further modifications are desired, the user can treat this new graph as he did the original graph, adding or moving points, redrawing the green curve, changing iteration numbers and weights, and resolving further. Note that the graph area will only show the results of the two most recent solves.

Two options remain in the interface which we have not yet discussed. The first, on the button located immediately below Update Graph is a button marked “Clear Graph”, which removes all updated solves, constraint curves, and reloads the original solution from startup. It does not remove the points that have been placed on the graph, however; to remove these, use the “Clear Points” buttons for either PTV or OAR. This way the user can choose to remove constraints on a single volume rather than having to start over entirely if he makes a mistake (at this time, there is no option to remove single points from the graph).

The second remaining option is immediately above the Exit button, and is marked “Save”. This button causes the current state of the graph area to be saved to a file called *image.tga* in the current working directory. Note that this will overwrite any previous *image.tga* files, and that the file should be renamed immediately if multiple images are to be recorded. Because of the availability of this workaround, we did not feel it was necessary to create a file naming dialogue for the user inside the program.

2.3.1 Accessing patient data using `initdata.gms`

In order to apply the present tool to properly processed data, the user must modify the *initdata.gms* file. This scan data should be imported using the `$GDXIN` function, currently located on line four of this file; the user should modify this line to include the path to their new data file. *All* data-specific modifications should be made in this file; all subsequent models use only the generic data sets PTV, OAR, and Normal. Also any specific sets should be specified here (in our current model, we use `prostate`, `bladder`, `rectum`, and `Normal`), and grouped appropriately into PTV, OAR, and Normal. Any modifications in these groupings should also be made here (for example, if an organ previously classified

as Normal should be reclassified as OAR or vice versa).

This file also contains a method for sampling over the normal tissue - since the normal tissue comprises such a large proportion of the relevant tissue, to speed up the solution time it has been prudent to reduce the number of normal voxels over which dose is calculated [4]. Models may also eliminate normal tissue entirely and concentrate only on the well-being of the OAR and the effective treatment of the PTV.

2.3.2 Getting meaningful and useful results

Aside from producing visual output and optionally images of the solution graph, the tool also produces a listing of the angles and intensity levels used to generate the weights on the voxels that comprise the graph. This file is found in the model's directory and is called *weights.txt* and includes the intensities of radiation that should be used from each angle to deliver a dose whose profile matches that displayed in the tool.

Also created after a solution is generated is the file *update_results.txt*, containing the amount of dose deposited in each of the OAR and PTV voxels, but the usefulness of this file outside of its intended purpose of relaying information between the solver and the display tool is unclear.

2.3.3 Required programs

The current version of our GUI tool is compiled to run only on the RedHat Enterprise distribution of the Linux operating system; it may be hospitable to other distributions of Linux but at this stage we cannot guarantee portability. The visual interface was written with a combination of OpenGL and the Fast Light ToolKit (FLTK) for ease of implementation, as the focus of the program was not on the aesthetic quality of the interface but rather on its functionality.

In order to run this tool, it is required that the user have a version of the General Algebraic Modeling System (GAMS) installed. The systems on which the program has been tested use a recent version, GAMS 22.7 (though the most recent release at time of writing is GAMS 23.0). GAMS incorporates a scripting language for the description of models of the form discussed in Section 1.1, and incorporates commercial solvers for all types of optimization problems. The ILOG CPLEX solver, or CPLEX, is included in GAMS and provides solutions for linear programs like the one described by our model using the simplex method¹ (though it also contains options for using various interior point methods).

2.3.4 Interpreting results

As described in Chapter 1, the shape of a prescription DVH graph for our applications will feature the blue OAR curve as overlapping the y -axis - that is, 100% of OAR voxels receive zero dose - and the red PTV curve as a right angle, horizontal along the line $y = 1$ and vertical at the x value corresponding to the prescribed level of dose for the

¹The simplex method is a popular algorithm for the solution of linear programs. Visually, it moves along edges of the feasible region until it finds the optimal solution. For more information, see the following article from Georgia Tech explaining both linear programming and the simplex method in detail: <http://www2.isye.gatech.edu/spyros/LP/LP.html>

target area. In reality it is impossible to replicate this prescription plot exactly, though we hope to come as close as possible to approximating it.

When visually evaluating quality of solution, a “better” solution is a set of curves which closely approximate the prescription DVH described above. The OAR should remain as close as possible to the y -axis, and any overdose tail should be as low to the x -axis as possible, translating to only a few voxels receiving a high dose; ideally this curve will lie under any curve that the user defines. It may in fact be impossible to achieve an OAR curve that does not reach nearly up to $x = 1$ without causing severe damage to the quality of the PTV, since it is possible that in designating the PTV some voxels from an adjacent OAR were included in that volume.

As much of the PTV curve should be as vertical as possible, as close as possible to the prescribed level of dose (normalized in our tool to appear at the value $x = 1$); ideally this curve will lie above any curve that the user defines. Long curves above or below $x = 1$ are to be avoided; their presence indicates a large hot or cold spot where the tumor could be dangerously overdosed or receiving insufficient dose to be effective in treating it. Our particular tool concentrates on the insufficient dose and provides only the option to constrain underdose in the PTV; if overdose is in fact an issue, the OAR’s overdose constraints are easily replicable.

2.4 Features of the GUI tool

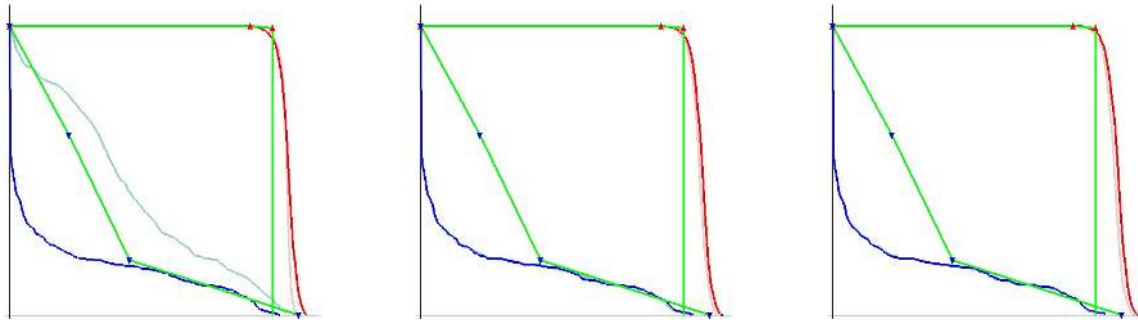
Aside from the basic functionality outlined above, the GUI tool includes several behaviors of particular interest to a user of the tool. These features may help the user to create better solutions more quickly, or to find constraints that more accurately reflect the behavior he wishes to observe. In this section, we describe the motivations behind these features (and some quirks) and how the user can best take advantage of their functionality.

2.4.1 Number of iterations

In certain situations, time may be a crucial factor in determining the desired quality of a solution. That is, the situation may arise where a “good” solution now is better than a “great” solution later, particularly if the relative gains afforded by waiting until later have a much smaller effect on the quality of the solution. In our tool, though the overall time to solve is relatively rapid, there are demonstrable differences in the amount of gain achieved in the first few steps over the gains achieved in the subsequent “adjusting” steps.

As shown in Figure 2.3, the amount of progress made in the first iteration may be most of the relevant progress needed, though the relative quality of the solution after several more iterations is undeniably higher. In fact, comparing the quality of the solution in part (b) to part (c), the marginal benefit of additional iterations may in fact be quite small.

One of the key steps in the solution of our algorithm is the construction of the set of voxels which violate the given constraints (for details, see chapter 3). The largest improvement in the number of violating voxels is achieved within the first few iterations. After these iterations, the size (and composition) of this set merely fluctuates slightly as the solver makes small adjustments to the weights and angles of the radiation beams



(a) The solution generated after one iteration of the algorithm.

(b) The improvement on the solution in (a) after one more iteration.

(c) The improvement on the solution in (b) after 10 iterations.

Figure 2.3: A comparison of the progress made by the tool after various numbers of iterations using the same constraints. In each figure, the previous iteration's solution is displayed as the lighter lines.

affecting these voxels. As we would hope, the overall trend of the number of violating voxels is decreasing, but we have no guarantee that it will decrease every time we run our algorithm.

Solution

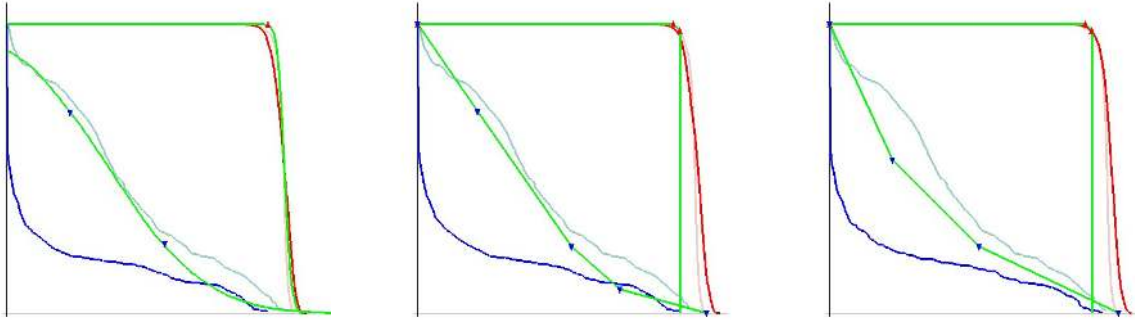
If time is in fact of the essence, we encourage the user to employ lower iteration counts - on the order of 2 or 3 - which may not satisfy the specified constraints entirely, but may reduce the solution time. However, we note here that each additional iteration adds only a few additional seconds to the solution time, as evidenced here.

Each iteration continues the solution from the point achieved by the previous iteration (or the initial solve for the first iteration) and adds only one or two linear constraints to the original problem. Linear programming solution restarts are fast when compared to other types of programs, and so these additional iterations are less expensive than if we were to employ nonlinear constraints.

2.4.2 Piecewise-linear vs. cumulative normal constraints

The nature of the underlying algorithm and model of our tool is the idea of uncertainty and probability. We observe in Section 2.1 that the distribution of PTV dose levels in the dose volume histogram roughly resembles the bell curve of a normal distribution. The algorithm we use takes advantage of this fact and aims to constrain the proportion of voxels in a volume receiving a dose less than or greater than some threshold value - that is, it attempts to constrain the upper or lower tails of the distribution to match that of some ideal random variable distribution, where the random variable is the level of dose received by a randomly selected voxel in the volume.

With this in mind, we originally decided to constrain our distributions with exactly that - a normal random variable distribution as described by user points (see figure 2.4(a)). As this figure shows, the distribution on the PTV actually very closely matches that of the cumulative normal distribution. However, these curves are normal curves



(a) A solution using a cumulative normal constraint curve.

(b) A solution using piecewise linear constraint curves which approximate the shape of the cumulative normal constraint.

(c) A solution using piecewise linear constraint curves constructed from similar points.

Figure 2.4: A comparison of the cumulative normal and piecewise linear constraint curves.

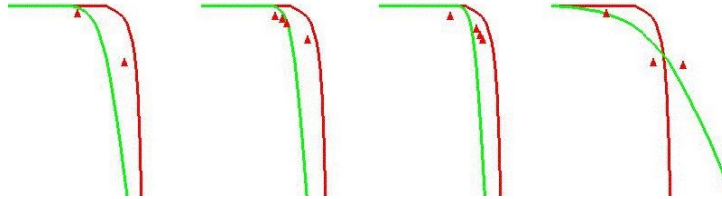


Figure 2.5: A series of the normal curves calculated to fit given PTV points, demonstrating the difficulty of accurately controlling the normal distribution constraint curve.

constructed using least squares fitting - they show the normal distribution which most closely approximates that described by the user placed points. This can create some difficulties in accurately developing constraints; see figure 2.5.

As we were increasingly frustrated with the difficulty of using a normal distribution to describe our constraints, we abandoned the idea of simulating a random variable entirely and opted for the much more predictable behavior of a piecewise-linear curve (see figure 2.4(b),(c)). While this particular curve may not directly correspond to any specific type of random variable, we recognize that all our algorithm requires is a *distribution* to fit, not a parametric form of a random variable. The theory motivating the algorithm we have chosen incorporates random variables to shape the dose distributions, but in theory and in practice we have found it to be extremely effective at achieving desired results.

In fact, a useful side effect of the use of a piecewise-linear distribution over a normal distribution is a reduction in the amount of inter-iteration calculation. Where the original implementation with normal distribution checked for constraint violations at a fixed number of thresholds (whose *position* varied at every iteration) and thus had to recalculate the violation constraints for each threshold after every iteration, the piecewise-linear implementation checks for violations only at the user-specified points and the violation constraints can therefore be pre-calculated and stored as constants for the duration of the run.

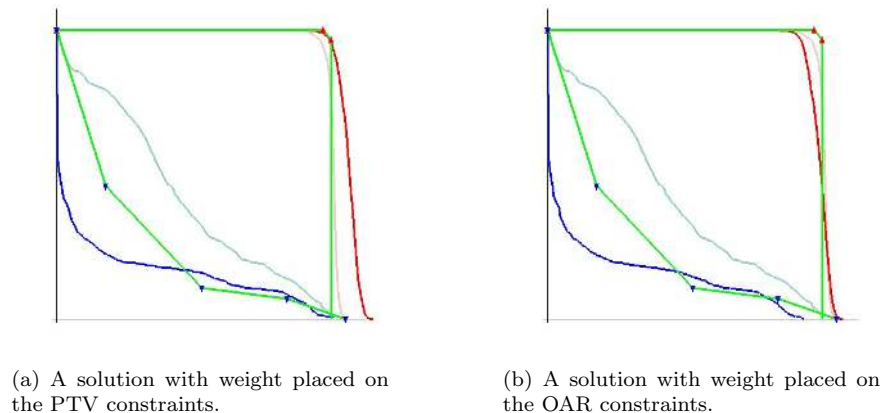


Figure 2.6: Demonstrating the effects of weighting on a difficult constraint set - examine in particular the difference in the PTV. Both figures use the same initial solution (shown as lighter curves) and constraints.

Remarks

While cumulative normal constraint curves are no longer supported in the current version of the tool, we believe it would be relatively easy to translate the functionality of the random variable distribution to the current tool if this is desired. We caution against using purely cumulative normal distributions, however, noting that as shown in Figure 2.4(a), the best-fitting normal curve may not pass through the point $(0,1)$ - representing the fact that 100% of voxels receive at least zero dose. Since this fact is important to both the theory and the practice of this mechanism, we suggest exploring other random variable distributions, such as exponential or geometric.

Future work may incorporate both options for constraints, and allow the user to choose a type of distribution; however we have not observed any practical advantages of using normal distributions, since all that the algorithm requires is their shape, which is quite easily approximated using a piecewise-linear curve.

2.4.3 Regional trade-offs

There are frequently times when, as a user of this tool, one's eyes will be bigger than one's stomach. That is, when providing constraints, the user may ask for more than is physically possible. In this situation, the volume weights described in Section 2.3 come into play. By default, we give equal priority to the constraints on the OAR and the PTV, but if one is determined to be of higher importance than the other, these constraints will allow the user to express this imbalance.

The interplay between these weights may have differing results. As shown in figure 2.6, this particular constraint set has characteristics such that weighting the OAR actually causes losses in the PTV, whereas putting a higher emphasis on the PTV still allows some gains to be made in the OAR while nearly satisfying the PTV constraints. We wish to draw special attention to the correspondence in figure 2.6(b) between the highest OAR dose and the lowest PTV dose, which suggests that the PTV and OAR may in fact share voxels, making a full satisfaction of both constraints impossible.

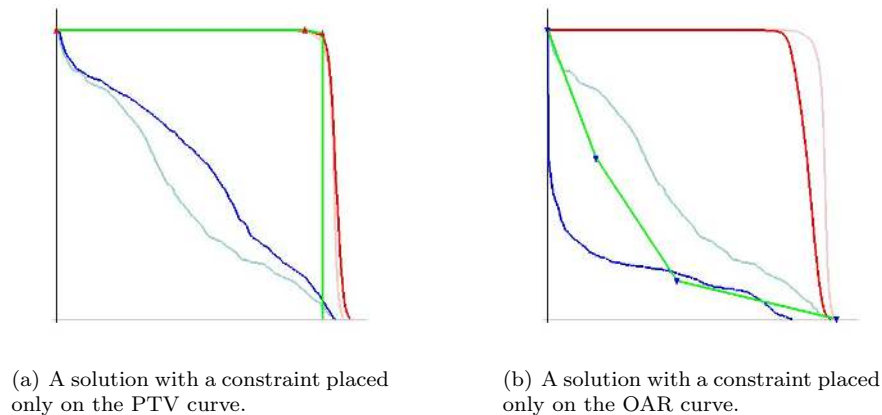


Figure 2.7: Demonstrating the effects of constraining only one of the volumes rather than both. The starting solutions are displayed as lighter curves and are the same in both figures.

These weighting constraints are of use in such situations; the user will have a much more intimate understanding of the particulars of each individual situation. If the user suspects that some of the OAR has in fact been included in the PTV, and that dosing that particular region would be highly detrimental to the patient, he can weight the OAR more heavily and willingly sacrifice some of the PTV. However, if the user knows that the tumor has in fact taken over part of the OAR, he can still reduce treatment to some of the OAR while placing a high priority on dosing the entire PTV adequately.

With the expectation that the user may design constraints describing dose profiles which are, in fact, impossible to achieve, our model includes slack variables for each of the PTV and OAR (for further details, see Chapter 3). We achieve the difference in importance of the quality of PTV or OAR solution by putting more importance on a smaller slack variable for either the PTV or OAR - a higher weight on the PTV corresponds to a higher penalty for having a positive slack, and similarly on the OAR.

2.4.4 Constraining only one region

There may arise instances wherein a user is satisfied with the initial solution for one volume but wishes to improve the solution on the other. In this case the temptation may be to provide constraints on only the PTV or the OAR, in an attempt to reduce solution time and eliminate the aforementioned tradeoffs inherent in the interplay between solutions. However, as we will show, leaving an adequate solution unconstrained can severely harm the quality of its solution in deference to another volume's constraints - tradeoffs in solution quality are inherent in the model whether a volume has been constrained or not.

Both solutions appearing in figure 2.7 begin from the same initial solution. In 2.7(a), the OAR curve was deemed adequate and the PTV was constrained; in 2.7(b), the PTV curve was deemed adequate and the OAR was constrained. However in both situations, we observe that the unconstrained volume's solution quality decreased dramatically: in 2.7(a), far more OAR voxels received a far higher dose than in any other solve reproduced in this paper; in 2.7(b), only the very few voxels at the highest doses even come close to

receiving the prescribed dose. This is due to the simple fact that a lack of constraints implies to the model a lack of importance of the quality of the current solution, rather than satisfaction with the current solution.

Solution

Quite simply, to avoid this issue, we recommend the practice of *pinning* current solutions; that is, introducing constraints that mimic the current characteristics of the solution so as to maintain as many of these characteristics as possible while attempting to improve the characteristics of the solution for the other curve. In some situations, it is in fact possible to nearly exactly maintain the quality of one solution curve while making significant improvements to the other (see, for example, figure 2.3(b)), though in other situations the user should be advised that tradeoffs must occur, as described in the previous section.

2.4.5 Additional anchor points

Before we discuss this particular aspect of our tool, we wish to enlighten the user as to the nature of the constraints used in our model, as they may provide some insight into the reasoning behind the advice in this section. The idea behind the constraints is not necessarily to require the dose distribution curves to out-perform the points placed by the user, but rather to match or out-perform the area underneath (for the OAR) or above (for the PTV) the constraint curves. Because of this, however, the user must exercise caution when placing constraints so as not to introduce areas that he did not intend.

During our use of the GUI tool, we noticed one particular issue with the solver's effectiveness: when constraining the PTV, it was difficult to achieve a solution which did not apparently violate the constraining distribution by a large amount. The dose distribution on the PTV would usually improve slightly in the direction of the constraints, but the model would terminate long before an acceptable solution was reached.

However, this was not a consistent problem. Certain PTV constraints would produce perfectly acceptable solutions, or terminate only on iteration count limit rather than constraint satisfaction but still produce increasingly good solutions as iteration limits were increased. This suggested that the problem was not mathematical, but rather one of perception.

The primary difficulty with the algorithm proposed by Ruszczynski is that rather than constraining the positions of the lines of the distribution, it places its constraints on the integrals of the distributions. While this does have the advantage of keeping the model convex and linear, it can result in the aforementioned misunderstandings.

In Figure 2.8(a), the constraining piecewise linear distribution has points at $(0, 1)$ and $(0.967113, 1)$. The area above the green curve is clearly 0, and only 25 (or approximately .9%) of the PTV voxels are violating the constraint. In Figure 2.8(b), however, the constraining piecewise linear distribution has points at $(0, 1)$ and $(0.967113, 0.9975)$. Even this small change causes the area above the green curve to increase to .0012, and this area constraint is satisfied with 161 (or 5.83%) of PTV voxels violating by an average of .02 Gy.

Unfortunately, this produces an area issue that is difficult to perceive visually - a triangle produced by $(0.9671 \times 0.0025)/2$ has approximately the same area as the rectangle

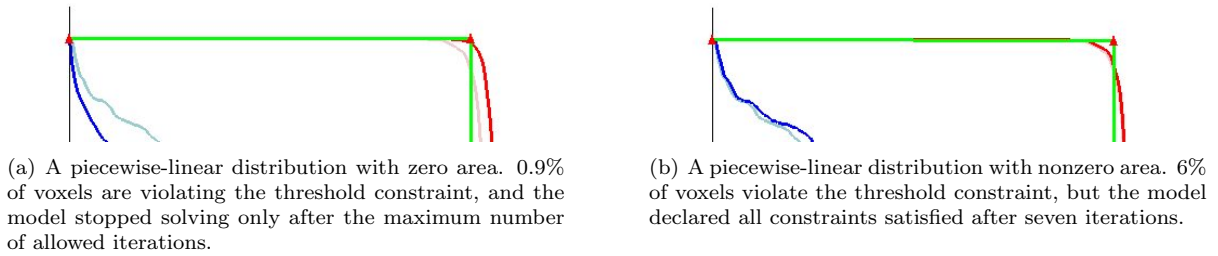


Figure 2.8: Two examples of PTV constraining with a very small vertical difference in the piecewise-linear curve resulting in a large difference in the quality of the solution. The light blue and light red curves show the original solution, and the darker curves depict the current solution.

produced by $.06 \times .02$ (here I approximate the PTV's area based on average underdose rather than directly calculating the integral simply for intuition's sake - this rectangle has approximately the same area as the triangle created by the distribution). However, as we see in the figure, this appears to the user as a gross miscalculation. Unfortunately, this cannot be solved within the program, as the problem exists entirely in human perception of area equivalencies (see extensive literature in psychology).

Solution

However, certain user-side solutions can be enacted. If the user desires the outcome of figure 1, he should ensure that when he places or moves points on the display that there is a point exactly on the line $y = 1$ near the location at which he wishes to begin constraining. If it is exactly on this line, the green curve will be exactly horizontal. Because of constraints that the y -values of the points on our piecewise linear curve be non-increasing, it may be difficult to replace a point on the $y = 1$ line once it has been moved; this is easily remedied by simply adding another point directly on the $y = 1$ line.

Chapter 3

Technical and Mathematical Details

This final chapter is intended for readers curious about the internal workings of both the program and the algorithm it implements. Where the previous two chapters have taken a broad, high-level approach to our tool, here we will discuss low-level implementation and mathematical complexities that we have encountered during the research process. The chapter will explore the technical considerations of the program we developed.

3.1 A cutting plane algorithm

It is interesting to note that according to practitioners of radiotherapy, clinically relevant effects are not a linear function of dose and irradiated volume according to radiotherapy practitioners. Niemierko ([11]) also registers his concern with computer optimized plans and their frequent inhomogeneities, suggesting that the main goal of creating an effective computer model should be a homogeneous dose distribution across the target area. His solution was, as described above, a method of equivalent uniform doses, centered on the idea of removing streaks and cold spots from the dose plot using complex nonlinear constraints. Here, however, we argue that it may be possible to eliminate these irregularities using linear constraints and focusing not on the dose plot, but rather on the DVH.

This algorithm was originally developed as a stochastic approach to shaping dose distributions. The idea was to simulate the distribution of the dose in the patient with a random variable, and constrain the tail of the dose histogram to match or outperform the tail of the random variable distribution. The simple model we use implements a cumulative normal distribution, given by

$$\text{cdf}_{\mu,\sigma}(x) = \frac{1}{2} \left(1 + \text{erf} \left(\frac{x - \mu}{\sigma\sqrt{2}} \right) \right)$$

As described in Chapter 1, we will deal with the patient on two levels of granularity. The most coarse granularity is the division of the patient into separate *volumes*, based on different requirements for their dose levels. The normal tissue is not the target of our dose, but clearly it is preferable not to highly dose such tissue. The PTV is the cancerous volume which we are seeking to dose as evenly and as close to the prescribed levels as possible; the OAR is the volume or set of volumes within the patient that would potentially be compromised by small amounts of radiation and must be avoided when

planning the dose delivery. The next level of granularity is the *voxel*, which is a small cubic volume of a fixed size, and is used to discretize the larger volume so that the dose levels can be more easily measured.

Due to the physical characteristics of our particular radiation delivery system (3D-CRT, as described in Chapter 1), our solution will determine a number of fixed angle beams to use and the intensity (or *weight*, as we shall refer to it henceforth) at which to deliver radiation from these angles. The beam weights will be denoted w_a , where a is the angle number, and the dose delivered to the voxel is denoted $sumDose(i)$, where i is the index of the voxel in question.

We wish to note at this point that for those readers also consulting [3], our equations will differ from those presented in that paper due to the fact that our cumulative representations are reversed. Where our y value denotes the percent of voxels receiving *at least* the corresponding level of dose, the source paper's y value denotes the percent of voxels receiving *up to* the corresponding level of dose. This does not affect the meaning or analysis of the constraints in any way.

3.1.1 Algorithm fundamentals

The algorithm makes use of several analogous components, which differ in notation based on which tail (overdose or underdose) we are constraining. In this section, we will primarily speak about the underdose tail, while mentioning the analogous components for the overdose tail. Our tool currently employs the underdose constraints on the PTV only, and the overdose constraints on the OAR only, but these are simply limitations that we have chosen for our implementation, not requirements of the algorithm itself.

First, we wish to consider the proportion of voxels in the volume which are receiving a dose under a threshold value, t , which we treat as the distribution function of a random variable $sumDose(i)$ corresponding to the dose received by a randomly selected voxel i in the PTV. This is denoted

$$G(t) = \frac{\text{card}(i \in \text{PTV} : sumDose(i) \leq t)}{|\text{PTV}|}. \quad (3.1)$$

The overdose constraint is denoted $F(t)$, and can be applied to any of the volumes in an effort to reduce hotspot tendencies. Applying both $F(t)$ and $G(t)$ to the PTV simultaneously may in fact have a similar effect to the constraints of EUD as enumerated above - a more vertical histogram curve corresponds directly to a less varied (and therefore more uniform) dose over the volume in question.

Given this random variable distribution, the algorithm then requires that at least a fraction β of the volume Y should receive doses exceeding t , written

$$G(t) = P\{sumDose(i) \leq t\} \leq 1 - \beta.$$

More specifically, the probability that a voxel receives a dose less than t is less than or equal to $1 - \beta$. The analogous overdose constraint specifies that no more than a fraction α receive dose exceeding t , and is written $F(t) = P\{sumDose(i) \geq t\} \geq 1 - \alpha$. The unfortunate difficulty that arises when using such constraints is the non-convexity of the feasible region defined by $G(t)$ in (3.1). The present algorithm's solution to this problem

is to recognize that this is a subset of a broader class of problems; that is, the value which β takes on can be specified by a particular random variable distribution to which we wish to fit our dose distribution curves:

$$G(t) \leq \Psi(t) \quad (3.2)$$

where $\Psi(t)$ has the property $\Psi(t) = 1 - \beta$ for some underdose threshold value t .

The distribution $\Psi(t)$ (and analogously $\Phi(t)$ for the OAR) are given target distributions, chosen by the modeler. Thus for any given value of t , the values of these distributions can be explicitly calculated. The original article [3] suggests beginning with a normal distribution; as discussed in Section 2.4.2 these prove too cumbersome for our implementation and we abandoned the normal distribution in favor of the highly customizable piecewise linear distribution curve.

The values denoted t in this section are, as noted earlier, the threshold dose values over or under which we wish to constrain the distribution of dose levels. In our model, the threshold value for the PTV is denoted T , and the value for the OAR is denoted R . There is, however, no restriction on the values of t , which allows the modeler to choose any value at all at which to begin his definition of “tail” - indeed, in many of the OAR demonstrations in Chapter 2, we constrain nearly three quarters of the x values achieved as the overdose tail. Most of our PTV constraints are, however, consigned to only the very small underdose tail, as the initial solution is already quite close to the desired configuration. We expect the user will quickly develop an intuition as to appropriate threshold levels while learning to use the tool.

These constraint types - constraints directly on the distribution functions rather than on dose areas in the body - are called *first order stochastic dominance constraints*, and as mentioned above, generally describe a non-convex feasible region. This algorithm, however, creates a convex approximation of this feasible region using linear constraints that constrain not the individual values, but rather integrals under the relevant curves.

$$\int_R^\infty F(t)dt \leq \int_R^\infty \Phi(t)dt \quad (3.3)$$

$$\int_0^T (1 - G(t))dt \leq \int_0^T (1 - \Psi(t))dt \quad (3.4)$$

Note that the right hand sides of (3.3) and (3.4) can be evaluated explicitly over Φ and Ψ , which as mentioned above are distributions selected by the modeler. For further discussion, see Section 2.4.2.

However, we employ the following identities to calculate the integrals of the distributions $F(t)$ and $G(t)$, noting that as described in (3.1), these functions are the percent of voxels violating the threshold constraint. Their integrals, then, are the average magnitude of violation over the entire volume:

$$\int_R^\infty F(t)dt = \frac{1}{|\text{OAR}|} \sum_{i \in \text{OAR}} \max(0, R - \text{sumDose}(i)) \quad (3.5)$$

$$\int_0^T (1 - G(t))dt = \frac{1}{|\text{PTV}|} \sum_{i \in \text{PTV}} \max(0, \text{sumDose}(i) - T) \quad (3.6)$$

Substituting these identities into (3.3) and (3.4), we begin to construct the constraints for the final model.

Before we describe our final model, however, we note that the max function is not a linear constraint. In order to maintain status as a linear program, the model instead removes the max function completely from (3.5) and (3.6), substituting instead

$$\sum_{i \in B^l} (sumDose(i) - T_{PTV}^l)$$

$$\sum_{i \in A^l} (R_{OAR}^l - sumDose(i))$$

Where A and B are subsets of the OAR and PTV respectively, containing the voxels whose current dose level ($sumDose(i)$) violates the threshold constraints R and T .

To prevent a large number of negative values from masking the violating set, the sums are no longer performed over the entirety of the PTV or OAR, but rather on a subset of voxels determined between each iteration to be the currently violating set (see Section 3.2.3 for more details).

The cutting-plane method

Denoting the right hand sides of (3.3) and (3.4) by $u(R)$ and $v(T)$ respectively (since we replace $\Phi(t)$ and $\Psi(t)$ with our chosen distributions and can now explicitly calculate their values for R and T), we now present the algorithm described in [3] implemented by our tool. Initial conditions are set as described in Section 3.2.3, using a combination of user-provided data and values generated from the patient data. The value k is used to denote the current iteration.

The specific advantage of this method is that it can be added on to an existing model, linear or nonlinear. The objective function $f(x)$ and any other constraints $x \in X$ not directly related to our algorithm are intentionally unspecified (see Appendix A.2 for our implementation) so as to allow for this flexibility.

Each iteration of this algorithm is centered around the creation and refinement of the sets A and B as described above. At each iteration l , a new constraint is added for the PTV and/or OAR corresponding to a threshold point R^l or T^l where the integral of the PTV or OAR distribution violates the goal distribution most. Based on this threshold value and the doses delivered to each voxel for iteration l 's solution, the algorithm then selects the subset of PTV or OAR voxels which have dose levels below or above the threshold value respectively, and these subsets become B^l and A^l .

Step 1 Solve the GAMS model which now includes the following constraints:

$$\min f(x) \text{ subject to } x \in X$$

$$\frac{1}{|OAR|} \sum_{i \in A^l} (R_{OAR}^l - sumDose(i)) \leq u(R^l)$$

$$\frac{1}{|PTV|} \sum_{i \in B^l} (sumDose(i) - T_{PTV}^l) \leq v(T^l)$$

where $l = 1, \dots, k$.

Step 2 Using the values calculated in this model, find the largest violation:

$$\delta_O = \max_{t \geq R} \left\{ \frac{1}{|\text{OAR}|} \sum_{i \in \text{OAR}} \max(0, t - \text{sumDose}(i)) - u(t) \right\} \quad (3.7)$$

$$\delta_P = \max_{0 \leq t \leq T} \left\{ \frac{1}{|\text{PTV}|} \sum_{i \in \text{PTV}} \max(0, \text{sumDose}(i) - t) - v(t) \right\} \quad (3.8)$$

If δ_O and δ_P are less than our tolerance level, we stop execution and report a constraint-satisfying solution; otherwise we continue.

Step 3 Increase k by 1, set R^k to be the maximizer of (3.7) if greater than our tolerance; set T^k to be the maximizer of (3.8) if greater than our tolerance. Calculate the violating sets A and B to contain those voxels violating these thresholds.

$$A^k = \{i \in X : \text{sumDose}(i) \geq R^k\}$$

$$B^k = \{i \in Y : \text{sumDose}(i) \leq T^k\}$$

Step 4 Repeat from Step 1.

3.1.2 Creating constraining distributions

Our initial solution to the problem of choosing an appropriate distribution was to employ standard normal distributions. Given values for t and β that we derived from the user-provided points, we then fit a normal distribution using a linear regression to satisfy the requirement that

$$t - \mu_Y = \sigma_Y z_{1-\beta}$$

where μ_Y and σ_Y are the mean and standard deviation, respectively, of the distribution, and $z_{1-\beta}$ is the $(1 - \beta)$ -quantile of the standard normal distribution, which we calculate using the inverse of the erf function:

$$z_{1-\beta} = \sqrt{2} (\text{erf}^{-1}(\beta))$$

We fit this equation using a linear regression. Our model used all user-generated points if the user provided more than one, and an existing point in the actual initial distribution if he did not (for OAR, the point with the highest dose value; for PTV, the point with dose value halfway between the current point and the highest dose value).

However, as described in Chapter 2, we discovered that doing this sort of fitting to user specified points was, for lack of a better word, clunky. It was extremely difficult to exert any sort of fine-grained control over the shape of the distribution, and in particular for the OAR distribution, the characteristics of a normal distribution did not necessarily match the physically achievable characteristics of the dose distribution. Most importantly, as shown in figure 2.4(a), a true cumulative normal distribution which matches the points provided by a user may not intersect the point (0,1) - which, in the DVH, implies that some percent of voxels are not even receiving zero dose, and thus makes very little sense.

Other random variable distributions are possible which follow the characteristics of the dose distributions more closely - for example, a geometric random variable. However,

we note that what is important in the model is not the fact that our ideal distribution is that of a random variable. Instead, all the model is concerned with is the integral of the distribution between zero and the threshold value (or, for overdose constraints, the integral between the threshold value and infinity). Thus we have chosen to implement a piecewise linear constraint curve rather than attempting to fit a random distribution curve, which has a number of advantages.

The first advantage of a piecewise linear curve is intuitiveness. When placing two points on the graph, we can predict exactly where the piecewise linear curve will connect the two - an important characteristic, considering the constraints are relative to the curve as well as the user-placed points. There is no guesswork and very little trial-and-error involved in directing a curve exactly where the user desires it to go.

Another advantage of the piecewise linear curve is its flexibility. Because it affords a high amount of local control, the user can create a piecewise linear approximation of ANY desired random number distribution, and even design distributions which do not correspond to any existing mathematical expression. By moving to a piecewise linear distribution, we in fact do not lose any of the functionality afforded by random variable distributions, and gain quite a bit more.

The final advantage of moving to piecewise linear distributions is the ease of calculating their integrals. Where a normal distribution involves calculus, infinite limits and a number of dynamic calculations throughout the course of the algorithm, our piecewise linear integrals can be calculated statically at the beginning of the solution using simple geometric area formulas: the areas described by a piecewise linear curve are in fact just the sums of a small number of rectangles and triangles. This helps particularly to increase the ease of interpretation and solution.

3.1.3 Initial conditions

This algorithm includes specific instructions regarding several initial conditions relevant to our implementation. First, the initial violating voxel sets A and B are to be the entire OAR and PTV, respectively; second, the initial threshold points R (on the OAR) and T (on the PTV) should be set to be the lowest and highest dose values, respectively, at which we are constraining.

The sets A^i and B^i we call the “violation” sets for each iteration i , and are held constant over subsequent iterations. That is, the set A^1 will always be the entire OAR, just as the set B^1 will always be the entire PTV, and both will correspond to constraints using our initial conditions. At the next iteration, another set of constraints will be added, constraining subsets A^2 and B^2 of the OAR and PTV to be subject to more restrictive constraints. These subsets are chosen automatically at each iteration k , and consist of

$$A^k = \{i \in X : \text{sumDose}(i) \geq R^k\}$$

$$B^k = \{i \in Y : \text{sumDose}(i) \leq T^k\}$$

where R^k and T^k are the aforementioned threshold values. Note that A^k and B^k depend on the dose delivered at iteration k of the algorithm.

The threshold value R is used in the constraints on the OAR for shaping the overdose distribution. Its initial value is set to be the lowest dose level at which we wish to begin

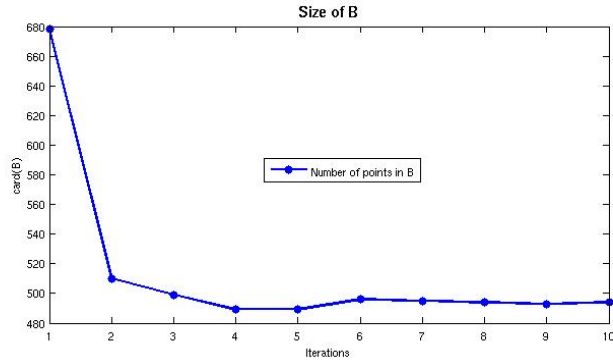


Figure 3.1: The size of the violating set of voxels (labeled B) over 10 iterations.

shaping. Similarly, T is used for the PTV in shaping the underdose distribution, and initially represents the largest dose level at which we wish to terminate our constraint. Over the course of the iterations of the model, R will increase toward the highest dose at which we constrain, while T will decrease toward the lowest dose.

Due to our current setup with a piecewise linear distribution, we can predefine several possible values for R and T as the points defining our piecewise linear curve, and calculate the values statically before beginning the optimization.

3.1.4 Drawbacks of the algorithm

In order to maintain the characteristics of the algorithm as a convex linear program, we make sacrifices as far as the direct constraining of doses to voxels. Where models that make direct constraints on the characteristics of the dose within the body may have more direct control over the dose profiles, we place greater importance on a simple, fast solution.

The first issue that arises is the potential for very small but very radical dose outliers. As mentioned above, A and B are the violating voxel sets per iteration; as the solution makes progress, their sizes should evince an overall downward trend, though this decrease is not required and may not in fact appear at every iteration (see for example figure 3.1). Given the threshold R or T , our constraints are concerned with the number of violating voxels above that particular dose level, and not with their individual dose levels. The unintended consequence of this is that outside of the threshold dose level, some voxels may in fact experience an increase (or decrease) in their dose level while the overall number of overdosed (or underdosed) voxels decreases, satisfying the constraint that the probability of choosing an overdosed voxel fits the given distribution, but perhaps resulting in health issues.

Anchor points are provided in our implementation to avoid issues like this in practice, but as these anchor points are only a feature of our implementation and not of the algorithm, this is a potential hazard that may result in very small areas of very high or very low dose. While sufficiently small areas of OAR areas may be tolerant to high doses, leaving small areas of the PTV underdosed is a potential health risk and could result in later regrowth of the tumor.



Figure 3.2: Examples of constraint satisfaction in the OAR with point-wise violations that are not considered in the area-based constraints.

The other primary issue with this algorithm that we have not been able to account for in our implementation - and is therefore an issue of which the users of this program must be aware (see Section 2.4.5) - is a function of the choices made in order to design a linear, convex model. The constraints used, as described above, are concerned only with the areas described by the constraining distribution and the PTV and OAR curves. That is, constraint satisfaction is merely a matter of area comparison, where the dose distribution’s tail should be less than or equal to the constraining distribution’s integral, and pointwise satisfaction of constraints is not at all considered (see Figure 3.2).

$$\frac{1}{|\text{PTV}|} \sum_{i \in \text{PTV}} \max(0, T - D_{\text{PTV}}(i)) \leq \int_0^T (1 - \Phi(t)) dt$$

$$\frac{1}{|\text{OAR}|} \sum_{i \in \text{OAR}} \max(0, D_{\text{OAR}}(i) - R) \leq \int_R^\infty \Psi(t) dt$$

Effectively, we constrain the *average* underdose or overdose, rather than individual voxel dose levels. However, the issue now becomes that rather than producing small numbers of extreme outliers (and compensating for them with many voxels which gain enough to produce a net negative effect on the constraint), we reduce (or increase) the dose level on all voxels with only very few outliers in either direction.

Why is this an issue? While we do wish to reduce the dose level on all OAR voxels or move all PTV voxels toward their prescribed dose level, the issue arises not in the practical biological result, but rather in the result as presented in the graph upon which we place our constraints.

Shown in figure 3.2 are two situations wherein the solve terminated before the iteration limit, which means that GAMS considered all constraints satisfied to within the acceptable tolerance. However, at first glance the user may think that no, in fact their constraints have not been satisfied, as the curve of the OAR is still above the point they had placed. What the model is concerned with, however, is the area comparison between the constraint and the OAR - closer inspection reveals that the areas contained between the violating OAR and the constraint curve are nearly equal to if not less than the areas contained between the non-violating OAR and the constraint curve. That is, by moving far below the constraint curve at some points, the distribution is allowed to move above it at others.

As discussed in Section 2.4.5, the best way for a user to model around this potential issue is to add additional “anchor” points to the piecewise linear distribution. When constraining the PTV, add a point on the line $y = 1$ at the lowest allowable dose level

before placing other constraints. Similarly in the OAR, add a point on the x -axis at the maximum allowable dose level, and shape the tail of the piecewise linear distribution in a more detailed manner.

3.2 Interfacing with GAMS

Technically speaking our primary issue was interfacing with the GAMS system. In order to increase the speed of the solver, we used several options within the model itself. First, since our algorithm involves multiple calls to the GAMS system, we used the option `model.solvelink = 2`, which causes the solver subsystem to be invoked using the spawn method rather than saving and restarting the subsystem at each iteration. Also, in order to ensure that all solves complete fully rather than hanging up the system, we increased the number of allowable iterations to 500,000.

During our iteration loop, we ensure that not only are we outside of our tolerance range and within the iteration limit specified either by the user or by default, we also ensure that the previous solution terminated successfully (`model.modelstat` is less than or equal to 1). If the previous solution was infeasible, it makes little sense to continue attempting to solve a model which cannot be feasible, since subsequent iterations only add constraints; constraints are never removed.

Due to the fact that GAMS is a third-party program, our tool interfaces with GAMS using text-based model files (with suffixes `.inc` and `.gms`) and system calls through the operating system. Most of the individual solution parameters are provided through files; the files `loop.inc` and `fullmodel.inc` in particular hold the majority of the configuration data. However, we do provide three pieces of information via system calls - the number of iterations and the weights to the PTV and OAR are given as arguments to the system call that begins the GAMS solve.

We also receive our output from GAMS entirely via text-based files; as mentioned in Chapter 2, each solve terminates with the output of the files `weights.txt` and `update_results.txt`. The weights file is intended for researchers and contains a list of the weights of each beam, ordered by angle number. The results file is intended only as a communicative device between GAMS and the tool, and contains the dose levels as calculated for each voxel in the PTV and OAR.

While we have chosen GAMS for our optimization solution system due to its ready availability within our department, ease of use via the GAMS modeling language, and our own prior knowledge of the system, we acknowledge that the proprietary nature of the system poses a significant deterrent to the portability of our tool. However, we wish to emphasize the fact that the same solutions can be achieved using non-proprietary software and internal code, rendering this an entirely self-contained tool.

3.3 Future work

We believe that an important extension of this model and program would accommodate multiple OAR and PTV volumes. The data underlying the models which generated the images for this paper are derived from a prostate cancer patient, with a single tumor

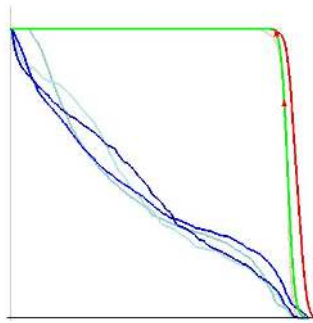


Figure 3.3: An early version of the tool which displays multiple OAR regions (in dark blue) and their initial dose profiles (in light blue).

volume in the prostate, but OAR areas comprising both the bladder and the rectum. Because these areas have different anatomical characteristics, the user may be aware of different tolerance levels for each area, and place more stringent constraints on one than the other, or a higher priority on satisfying constraints for one volume than the other. Alternately, were we to be dealing with a patient with multiple PTVs, the prescribed dose level for one tumor may be much different than for the other, necessitating the multiple PTV representation and constraints.

Inclusion of multiple OAR or PTV regions was actually achieved in an earlier version of this tool, but discarded at the time due to the lack of flexibility inherent in that particular hard-coded representation. Because of the unpredictability of individual situations (multiple tumors, varying numbers of sensitive organs, etc.), any implementation of multiple volume representation must in its very nature be a highly configurable tool, which interacts with the provided data.

Appendix A

Implementation Details

This appendix will discuss the basic structure of the source code, as well as methods for using FLTK, compiling the code, and some references for FLTK. The method of compiling FLTK currently used necessitates that all C++ source code be contained in a single file, resulting in the current source file length of 1474 lines.

A.1 C++ Source

For ease of discussion and further modification, the source code file `fulltrial-pwl.cpp` has been delineated by comment lines into nine sections. All code between these comment lines will be hereafter referred to as a *section*. We will refer to all subroutines as *methods*. To compile FLTK code, one can use the command `/s/fltk/bin/fltk-config --use-gl --compile filename.cpp`; however, for convenience, we have created a Makefile which is invoked in the standard method as simply `make`.

Section 1 - Header Files and Global Variables

As this program uses FLTK and OpenGL, the first required header files are `FL/gl.h` and `GL/glu.h`, for basic functionality. All other headers are basic C++ or FLTK libraries, however the additional file `tgasave.h` is required for the image saving capabilities.

The global variables present here exist for simplicity:

- `PTV`, `OAR`: vectors which store the current solution curves
- `PTVdose`, `OARdose`: vectors which store the current solution values
- `oldPTV`, `oldOAR`: vectors which store the previous solution curves
- `points`: vector of `PTVup` (index 0) and `OARdn` (index 1) points shown
- `Normal`: vector of points describing the piecewise linear constraint for `PTV` (index 0) and `OAR` (index 1)
- `numPTV`, `numOAR`: number of voxels in `PTV` and `OAR`
- `mousedownx`, `mousedowny`: last coordinates of mouse click

- *maxval*: maximum dose to any voxel in PTV or OAR
- *selected*, *special*: store ID of current point to move and current crucial point, respectively
- *dragging*, *placing*, *choosing*: state indicators
- *O*, *P*: indicators for OAR vs. PTV
- *illegalx*, *illegaly*, *illegal2*, *illegal4*: values for bounds checking, used in FLTK
- *pi*: 3.141592653589793

Section 2 - FLTK Workings

This section is what drives the graphical portion of the program. It creates our own instance of an FLTK window and includes an OpenGL screen, as well as methods to manipulate the output to that screen. The *drawpoint* method renders the PTVup and OARdn points to the screen, while *draw* takes care of all of the other rendering issues, such as the OAR, PTV and piecewise linear curves and dragging and placing PTVup and OARdn points.

Section 3 - GAMS Result Reader

The primary method in this section, *readResults*, takes raw data from a newline-delineated text file formatted so that the first line is the number of voxels in the PTV (stored in *numPTV*), the second line is the number of voxels in the OAR (stored in *numOAR*), followed by the PTV dose data (*PTVdose*) and finally the OAR dose data (*OARdose*). Any lines left in the file after *numPTV+numOAR* lines have been read are ignored. The method then creates a cumulative dose volume histogram and stores the data in *PTV* and *OAR*, and finally redraws the window.

The other method here, *getPoints*, runs only at the beginning of the program and serves to check whether the appropriate results file exists and call the initial GAMS model for a run if it does not.

Section 4 - ErfInv Functions

These are utility functions, included because C++ libraries contain an *erf* function but not its inverse. *Erf* is defined as:

$$erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$$

and is the standard error function, as well as being easily convertible to determine the value of a cumulative normal distribution. The functions here were verified against Matlab's built-in *erfinv* function, and provide *z* given *erf(z)*. This section of code is not used in the current implementation, but rather remains from the earlier cumulative normal constraints. This section has been left in the source in the interest of extensibility.

The *erf* function described here is the standard *erf*, however we note that this particular function is not available in GAMS, where one must instead use *errorf*. The relationship between the two functions is as follows:

$$\begin{aligned} erf(x) &= 2 * errorf(x * \sqrt{2}) - 1 \\ errorf(x) &= \frac{1}{2} \left(erf\left(\frac{x}{\sqrt{2}}\right) + 1 \right) \end{aligned}$$

Section 5 - Image Saving

Unmodified code courtesy of Rob Bateman, available publicly on the Internet. Stores the current state of the graph screen (without cursor) as a targa named *image.tga*. If users wish to save multiple images in one run, the output images should be renamed between saves to avoid overwriting.

Section 6 - Integral Calculations

This section contains an analytical method for calculating

$$\frac{1}{2} \int_0^T erf\left(\frac{x}{\sqrt{2}}\right) + 1 dx$$

(the integral for constraining the PTV) though previous versions of the program also included Simpson's method, the trapezoidal method, and Simpson's method of approximating

$$\int_T^\infty \left[1 - \frac{1}{2} \left(erf\left(\frac{x}{\sqrt{2}}\right) + 1 \right) \right] dx,$$

(the integral for constraining the OAR) which uses a large multiple of T to approximate ∞ . These methods were used only once, in calculating the initial conditions for the Ruszczyński GAMS model, and all subsequent calculations were performed within GAMS. Again, this section remains only in the interest of extensibility.

In another previous implementation, this section also contained a method for calculating

$$\int_T^\infty e^{bx} dx = \frac{1}{b} (e^{b\infty} - e^{bT})$$

as an experiment in using exponential distributions which we later abandoned in favor of the flexibility of piecewise linear distributions. We note that generally b is negative, so this became $\frac{1}{b} - e^{bT}$, since with a negative exponent, e^x is 0 at infinity.

Section 7 - Normal Distributions

The first method in this section is the utility method, taking the set of either PTVup or OARdn points and finding a suitable cumulative normal distribution that reflects their placement. This method makes use of the external GAMS model *linreg3.gms* to determine the μ and σ that best fit the distribution of points on the graph screen, and

returns these values as a `std::pair`, as (μ, σ) . Again, this is a legacy implementation for use with normal constraint curves; our current piecewise linear curves do not use this method.

The second method is used for visualizing this normal curve if need be; it takes a given μ and σ and uses *normal.gms* to generate a set of 8000 points conforming to this normal distribution. The set is then read in and analyzed in a manner similar to that of *readResults* above, and saved to a vector in the *Normal* global vector. The rendering displays either normal curve as a bright green line. This is also for use only with normal constraint curves, not piecewise linear.

In a previous implementation, this section contained two additional methods for fitting and drawing an exponential curve to the OAR points. The fitting method calls an external GAMS model *linregE.gms* to determine the best exponent b when forced to observe a coefficient of 1, and the drawing method calculates all points on the curve directly using the function $y = e^{bx}$.

Section 8 - FLTK Button Functions

All button functions as described in section 1.2 are implemented as *callbacks* in this section. The format of a callback is specified by FLTK as `void callback (Fl_Widget* w, void* v)`. As seen in FLTK documentation (available online), these can be used to provide arguments to the callbacks, but we avoid that in the interest of simplicity.

The callback methods contained here are as follows:

- *cb_addPup*, *cb_addOdn*: enter placing state for adding PTVup or OARdn points
- *cb_rmPup*, *cb_rmOdn*: remove all PTVup or OARdn points
- *cb_chooseP*, *cb_chooseO*: select a crucial point
- *cb_restart*: run the Ruszczyński algorithm using points on screen
- *cb_normalP*, *cb_normalO*: sort user-defined points by x coordinate and draw the corresponding piecewise-linear curves (method names remain from previous implementations using normal distributions)
- *cb_save*: save current state of graph screen as *image.tga*
- *cb_refresh*: clear all normal curves and restore initial data from startup
- *cb_quit*: restore initial data and exit program

All included files in the cutting plane algorithm are written dynamically by the method *cb_restart*. Any modifications to the algorithm loop, initial data, or constraints should be made in this callback method; any modifications made to the files themselves will be **overwritten** on the next run of the program. This allows the model to reflect the user's input more accurately.

Section 9 - Main

This method uses *getPoints_i* to get the initial solve data, initializes the global variables, builds the FLTK window, and lastly runs the program using the all-important `F1::run()`.

A.2 GAMS Source

Much of the critical GAMS source code is contained in *loop.inc*, which is written by the above callback method *cb_restart*. The bare bones of the code are contained in *fullmodel.gms*, which is also run by *cb_restart*.

File 1 - *fullmodel.gms*

This file contains the objective function (*objective*), as well as the voxel-specific dose calculator (*cons(I,J,K)*) and a pair of constraints to enforce an allowable range for the PTV dose (*penUn(I,J,K)* and *penOv(I,J,K)*). The allowable range is T_u to T_o , and the objective function currently minimizes the number of voxels in the PTV receiving dose outside of this range.

The model then includes the relevant constraints for PTV and/or OAR, as written by the GUI program and based on the PTVup and OARdn points on the screen. If no points are present for one of PTV or OAR, no constraint on that volume is included.

The `rucz.solveLink = 2` option controls the behavior of the solve statement later in the model. The **2** option starts the solution by spawning a new instance of GAMS instead of instructing GAMS to save its state, restart, solve, and reload itself upon completion (as is the default). The end effect of this option is to significantly speed the raw time it takes to solve the GAMS model.

The remainder of the model includes the solution loop, displays the final δ values after *card(iter)* iterations, and prints the voxel dose weights to the file *results.txt* for the GUI program to read.

File 2 - *loop.inc*

This file contains the primary functionality of the Ruszczyński algorithm. Every iteration, it performs the following tasks:

- (1) Alters the dynamic iteration sets to include the current iteration
- (2) Determines the sets A and/or B of violating voxels for the current iteration
- (3) Solves the model with all current information
- (4) Checks to see if the model has solved successfully, exits if infeasible or unbounded
- (5) Determines the maximum δ value for PTV and/or OAR
- (6) Retrieves the new *rhs* value for PTV and/or OAR

Upon exiting the loop, the model displays the maximum remaining δ values, the points at which it enforced the constraints, and the *rhs* values for all volumes used.

List of Scalars

- *littleT*: a variable which takes on values in the range of the integration for updating δ after solving

- *it*: number of intervals over which to calculate δ
- *int(it)*: the integer value of the iteration (1-20)
- *result*: a variable for calculating the integral of the distribution
- *up, lo*: the upper and lower bounds for integrating the distribution
- *sigmaP, sigmaO*: the standard deviation values for the distributions of PTV and OAR, respectively
- *muP, muO*: the mean values for the distributions of PTV and OAR, respectively
- *maxdeltP, maxdeltO*: the δ values for PTV and OAR, respectively
- *coeffP, coeffO*: the coefficient for the analytical calculation of the integral, given by $(\sigma\sqrt{2}/2)$
- *big*: a variable for calculating the integral of the OAR distribution, effectively the value of the function at infinity
- *v*: a variable for choosing an appropriate approximation for infinity in the OAR integral
- *s1,s2,h*: variables for calculating the Simpson's approximation for infinity, deprecated
- *d*: number of intervals for calculating Simpson's approximation, deprecated
- *evens(d), odds(d)*: integer values of the *d* set, deprecated
- *modstatus*: determines the state of the solution so that we can exit the loop if the model has not solved properly

A.2.1 GAMS Models

The following code is the linear GAMS model our implementation uses to create an initial solution for the tool to improve upon, contained in *initial.gms*. Any model, linear or nonlinear, can be substituted here; if quality of initial solution is unimportant, random weights could be assigned and all optimization could be performed within the tool itself.

```

$title LP Radiation Treatment for Visualization

option limrow = 0, limcol = 0, solprint = off;

* get dose matrix and sets
$include initdata.gms

* threshold values
scalar To          'overdose threshold'    /1.001/;
scalar Tu          'underdose threshold'   /0.999/;

```

```

variable obj;
positive variables w(nAngle), sumDose(I,J,K), delta(I,J,K);
equations defobj, cons1, penU, pen0;

* basic LP model
defobj..
    obj =e= (1/card(PTV))*sum(PTV(I,J,K), delta(I,J,K));

* linear dose definition
cons1(I,J,K)$(PTV(I,J,K))..
    sumDose(I,J,K) =e=
        sum((nAngle, nWedge), Dose(I,J,K,nAngle,nWedge)*w(nAngle));

* underdose penalty
penU(PTV(I,J,K))..
    delta(I,J,K) =g= Tu-sumDose(I,J,K);

* overdose penalty
pen0(PTV(I,J,K))..
    delta(I,J,K) =g= sumDose(I,J,K)-To;

model basic /defobj, cons1, penU, pen0/;
solve basic using lp minimizing obj;

```

In this initial model our objective function deals only with the PTV, simply for expediency. Our objective function for the cutting plane algorithm incorporates not only the PTV but also two weighted slack variables corresponding to unsatisfied PTV or OAR constraints.

A.2.2 Objective functions

Finally we list here a series of objective functions, linear and nonlinear, which we have used in the course of the research, and which are all of potential use with this algorithm:

```

obj =e= (1/card(PTV))*sum(PTV(I,J,K), delta(I,J,K))
    + sum(oL, \%oarweight\%*slack0(oL))
    + sum(pL, \%ptvweight\%*slackP(pL));

```

This is the objective function which we currently use. If further computation is not an issue, the `delta` values (magnitude of violation outside of a small boundary around the prescription) could be calculated for the OAR and also considered in this function.

```

obj =e= sum(TISSUE, sumDose(TISSUE));

```

This crude objective function simply looks to minimize the dose to the entire body (TISSUE being composed of the PTV, OAR, and Normal tissue in this instance) while paying no attention to the prescription levels for the PTV.

```
obj =e= sum((nAngle,nWedge), Adose(nAngle,nWedge)*w(nAngle));
```

`Adose` is in this case a parameter corresponding to the sum of all values in the dose matrix for each angle and wedge (a *wedge* being a device used for altering the flow of radiation; our models included a variable to allow for these but never utilized them). The only variable here was then the weight for each angle, resulting again in a linear objective.

```
obj =e= (1/(1+(EUDzero('PTV')/eud('PTV'))**a('PTV')))*
(1/(1+(eud('OAR')/EUDzero('OAR'))**a('OAR')))*
(1/(1+(eud('Normal')/EUDzero('Normal'))**a('Normal')));
```

As described in Section 1.2.3, the equivalent uniform dose nonlinear model (EUD) takes several forms; this was the first we used. For others, see [11] and subsequent work. (Recall that in GAMS notation, the `**` symbol represents an exponent, analogous to \wedge in C and other similar languages.)

These four objective functions represent only a small subset of the possible objectives for use in radiotherapy treatment planning, and we wish to stress that any objective and model may be combined with the algorithm in our tool, simply by modifying our model file *fullmodel.gms*.

Integral Implementation

In the interest of extensibility, we also include here a record of our implementation of the integral calculations for use with random variable distributions, though they are not currently used in our piecewise-linear implementation.

$$u(t) = \int_T^\infty (1 - \Phi(t))dt, U_X \leq T \leq \infty$$

Define $\text{coeff0} = \frac{\sqrt{2}\sigma_O}{2}$, $\text{lo} = \frac{t-\mu_O}{\sigma_O\sqrt{2}}$, and $\text{up} = \frac{\text{maxdoselev}-\mu_O}{\sigma_O\sqrt{2}}$. The integral is calculated as follows:

$$u(t) = \frac{\text{maxdoselev} - t}{2} + \text{coeff0} * \left(\text{lo} \left(2\text{errorf}(\text{lo}\sqrt{2}) - 1 \right) + \frac{e^{-\text{lo}^2}}{\sqrt{\pi}} - \text{up} \left(2\text{errorf}(\text{up}\sqrt{2}) - 1 \right) - \frac{e^{-\text{up}^2}}{\sqrt{\pi}} \right),$$

$$v(t) = \int_0^T \Psi(t)dt, 0 \leq T \leq L_Y$$

Define $\text{coeffP} = \frac{\sqrt{2}\sigma_P}{2}$, $\text{lo} = \frac{-\mu_P}{\sigma_P\sqrt{2}}$, and $\text{up} = \frac{t-\mu_P}{\sigma_P\sqrt{2}}$. The integral is calculated as follows:

$$v(t) = \frac{t}{2} + \text{coeffP} \left(\text{up} \left(2\text{errorf}(\text{up}\sqrt{2}) - 1 \right) + \frac{e^{-\text{up}^2}}{\sqrt{\pi}} - \text{lo} \left(2\text{errorf}(\text{lo}\sqrt{2}) - 1 \right) - \frac{e^{-\text{lo}^2}}{\sqrt{\pi}} \right)$$

Bibliography

- [1] Beong Choi and Joseph O. Deasy. The generalized equivalent uniform dose function as a basis for intensity-modulated treatment planning. *Physics in Medicine and Biology*, 47:3579–3589, 2002.
- [2] Joseph O. Deasy. Computational Environment for Radiotherapy Research. Website, March 2007. <http://radium.wustl.edu/CERR/about.php>.
- [3] Darinka Dentcheva, Michael C. Ferris, and Andrzej Ruszczynski. Stochastic programming approaches for radiation therapy planning. Proposal of LP algorithm for radiation therapy planning, 2008.
- [4] Michael C. Ferris, Rikhardur Einarsson, Ziping Jiang, and David Shepard. Sampling issues for optimization in radiotherapy. *Annals of Operations Research*, 148(1):95–115, November 2006.
- [5] Michael C. Ferris, Jinho Lim, and David M. Shepard. An optimization approach for radiosurgery treatment planning. *SIAM Journal of Optimization*, 13(3):921–937, 2003.
- [6] Michael C. Ferris, Jinho Lim, and David M. Shepard. Optimization tools for radiation treatment planning in matlab. May 2003.
- [7] Terrence S. Furey, Nello Cristianini, Nigel Duffy, David W. Bednarski, Michel Schummer, and David Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 16(10):906–914, 2000.
- [8] Jinho Lim, Michael C. Ferris, Stephen J. Wright, D.M. Shepard, and M.A. Earl. An optimization framework for conformal radiation treatment planning. *Informs Journal on Computing*, 19(3):366–380, 2002.
- [9] T. R. Mackie, T. Holmes, S. Swerdloff, P. Reckwerdt, J. O. Deasy, J. Yang, B. Paliwal, and T. Kinsella. Tomotherapy: a new concept for the delivery of dynamic conformal radiotherapy. *Medical Physics*, 20(6):1709–1719, 1993.
- [10] T. R. Mackie, J. Kapatoes, K. Ruchala, W. Lu, C. Wu, G. Olivera, L. Forrest, W. Tome, J. Welsh, R. Jeraj, P. Harari, P. Reckwerdt, B. Paliwal, M. Ritter, H. Keller, J. Fowler, and M. Mehta. Image guidance for precise conformal radiotherapy. *International Journal of Radiation Oncology, Biology and Physics*, 56:89–105, 2003.

- [11] Andrzej Niemierko. Reporting and analyzing dose distributions: A concept of equivalent uniform dose. *Medical Physics*, 24(1):103–110, January 1997.
- [12] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2006.
- [13] Arinbjorn Olafsson, R. Jeraj, and Stephen J. Wright. Optimization of intensity-modulated radiation therapy with biological objectives. *Physics in Medicine and Biology*, 50:5357–5379, 2005.
- [14] Arinbjorn Olafsson and Stephen J. Wright. Linear programming formulations and algorithms for radiotherapy treatment planning. *Optimization Methods & Software*, 21(2):201–231, April 2006.
- [15] World Health Organization. Global cancer rates could increase by 50% to 15 million by 2020. Website, April 2003. <http://www.who.int/mediacentre/news/releases/2003/pr27/en/>.
- [16] Annette Martin Quinn. CyberKnife: A robotic radiosurgery system. *Clinical Journal of Oncology Nursing*, 6(3):149–150, 2001.
- [17] H. Edwin Romeijn, Ravindra K. Ahuja, and James F. Dempsey. A new linear programming approach to radiation therapy treatment planning problems. *Operations Research*, 54(2):201–216, 2006.
- [18] D.M. Shepard, M.A. Earl, X.A. Li, S. Naqvi, and C. Yu. Direct aperture optimization: A turnkey solution for step-and-shoot imrt. *Medical Physics*, 29(6):1007–1018, June 2002.
- [19] J. S. Tobias. Clinical practice of radiotherapy. *Lancet*, 339(8786):159–163, 1992.
- [20] Qiuwen Wu, Radhe Mohan, Andrzej Niemierko, and Rupert Schmidt-Ullrich. Optimization of intensity-modulated radiation therapy plans based on the equivalent uniform dose. *International Journal of Radiation Oncology*Biology*Physics*, 52(1):224–235, 2002.