An Application of the Extended Cutting Angle Method in Radiation Therapy

by

Valentin Koch

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Bachelor of Science Honours

in

The I. K. Barber School of Arts & Sciences

(Computer Science)

The University Of British Columbia

April, 2008

Abstract

Global optimization of continuous, non-linear functions are very hard problems, especially when the functions are multivariate and when analytical information is not available. Heuristic methods like simulated annealing provide good results. However, if time is a critical factor, those methods may deliver suboptimal solutions and give little information about the quality of the solutions.

Methods of Lipschitz optimization allow one to find and confirm the global minimum of multivariate Lipschitz functions using a finite number of function evaluations. The Extended Cutting Angle Method (ECAM), proposed by Gleb Beliakov, is a fast method to optimize a Lipschitz function over multiple dimensions.

The first objective was to fully implement the proposed algorithm and to test it on a family of classic global optimization problems. A second objective was to apply the algorithm to the problem of optimizing the radiation treatment for cancer patients. In radiotherapy, several x-ray beams are delivered to the tumor from different angles around the patient. The ECAM was tested against a simulated annealing algorithm to find the optimal angles of the beams in order to deliver the prescribed radiation dose to the tumor and to minimize the damage to healthy tissue.

Acknowledgements

First of all, I would like to thank my supervisor Dr. Yves Lucet for his guidance and coordination towards this thesis. Without his ideas and support, this work would not have been possible.

Also, I express a special thank you to Dr. Gleb Beliakov for his infinite patience in answering all of my questions and for the help he provided to implement his algorithm.

This is not to forget Shane Marrs and Meghann Dick for giving some of their valuable time in a busy period of an ending semester to review this paper. My acknowledgements also go to the other students who provided me with their valuable feedback, corrections and encouragements.

And last but not least, I would like to thank my wife and son for their continuous support and for enduring those long days and nights without "Papi".

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Global Optimization

Optimization is a very important field in applied mathematics. These problems are solved daily in finances, engineering, medicine, and statistics. Often those problems can be formulated using linear functions or non-linear convex functions, and can be solved by efficient algorithms in linear programming or convex analysis. Those specific areas of optimization are very well understood. If a problem contains multiple minimas, however, the situation changes drastically. This is where global optimization comes into play.

Global optimization is a mathematical problem that is considered to be very difficult. Several global optimization problems have been shown to be NP-complete. While it is sometimes difficult to prove that a global optimization problem is NP-complete, solving such problems is always a difficult task [11]. Global optimization problems are encountered in nearly all areas of natural sciences and engineering. Whether it is a traveling salesman problem in computer science, a protein folding problem in chemistry, or a particular shape optimization problem in engineering, the problems pose significant challenges for most researchers in each area.

Global optimization can be divided roughly into two subfields.

Probabilistic Methods are also called heuristic methods. The word heuristic is derived from the greek "heuriskein" (to find), famously used by Archimedes in his "Eureka!, Eureka!". Heuristic methods make use of probability distributions. They are usually algorithms with infinite iterations that converge to the global minimum with probability 1 as the number of iterations approaches infinity. Probabilistic methods include:

- Simulated Annealing
- Genetic Algorithms
- Neural Networks
- Taboo Search
- Single Linkage Clustering Methods

Deterministic Methods make use of structural information about the problem. Those methods may directly use the gradient of a function if the function is differentiable. They may also use other information such as Lipschitz continuity. Deterministic methods include:

- Interval Analysis
- Branch and Bound
- Two-Phase Methods
- Stochastic Adaptive Search
- Dynamical Search
- Algorithms based on local optimization.

In this paper we consider a recently proposed algorithm in the family of Branch and Bound algorithms. It belongs to the subfield of Lipschitz optimization. The first goal of this thesis is to fully implement the algorithm and test it on a family of classic global optimization problems. A second goal is to apply the algorithm to the difficult problem of radiation therapy optimization.

1.2 Radiation Therapy

Radiosurgery is a form of radiotherapy treatment, where external photon beams are delivered to the cancer tumor of a patient. An important and complicated task in the radiosurgery treatment of a cancer patient is the design of a treatment plan. Radiation simulation algorithms are used to calculate the dose distribution in human tissues. The data gathered with these algorithms can then be used to make decisions on planning variables (i.e. beam angles, beam intensity, beam shape, etc.).

Due to the complexity of the treatments and the difficulty of finding an optimal treatment plan, much of the research in the field of radiation therapy is aimed at the treatment plan optimization through efficient algorithms. A major problem in *Intensity Modulated Radiation Therapy* (IMRT) consists of optimizing the combination of beam intensity, beam segmentation and the different beam angles in three dimensions. The beam intensity and the beam segments are usually optimized either with *Direct Apperture Optimization* (DAO) or with *Fluence Map Optimization* (FMO). The former is often based on heuristic approaches, while the latter is mostly solved by Linear Programming. Recent Linear Programming models also incorporate the coplanar angles of a preselected set of beams [13].

In this paper, a deterministic global optimization approach is presented to optimize the beam angles. Unless a preselected set of beams is optimized, as in [13], the angle optimization problem is non-convex. In [3], a random search algorithm is used to solve the beam angles in combination with DAO. We propose a bilevel deterministic optimization approach to solve the beam angles directly. The advantage of the Extended Cutting Angle Method in the radiation therapy treatment planning is that the algorithm provides a lower bound on the global minimum. This is important for time consuming treatment in order to achieve the best results for the treatment time. An example of time consuming treatments are treatments with non-coplanar beams. The experimental results with coplanar beams show that the Extended Cutting Angle Method provides good results and could be an efficient tool for the optimization of non-coplanar beams.

Chapter 2

The Extended Cutting Angle Method

2.1 Lipschitz Continuity and the Pijavskii-Shubert Method

In deterministic global optimization, one makes use of the structural information about the objective function, in order to find the global extremum. Lipschitz continuity is a structural property that restricts the rate of change of a continuous function. A function in $\mathbb{R}$ is *Lipschitz-continuous* over the interval $[a, b]$, if it satisfies the inequality

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2| \tag{2.1}$$

for any arbitrary pair of points $x_1, x_2 \in [a, b]$ where $L \in (0, \infty)$ is a fixed constant. Many practical problems can be formulated with Lipschitz functions. In [9], it is shown that a Lipschitz continuous function on a bounded set I is bounded on I. The Lipschitz properties can therefore be used to construct an estimate for the global minimum and maximum of the function. Consider the following sample points of an arbitrary Lipschitz function $f(x)$ in $\mathbb{R}$ with Lipschitz constant L

$$a = x_0 \leq x_1 \leq \ldots \leq x_k = b \tag{2.2}$$

The following functions $h(x)$ are affine underestimates of $f(x)$ at the given sample points

$$h_i(x) = f(x_i) - L|x - x_i|, x \in [a, b], 1 \leq i \leq k. \tag{2.3}$$

If we maximize the set of all $h_i(x)$, we get a function formed by the intersections of all $h_i(x)$, namely

$$H^k(x) = \max\{h_i(x) : 1 \leq i \leq k\}, x \in [a, b], \tag{2.4}$$

the piecewise linear *underestimate* for $f(x)$. Since $\forall x, H^k(x) \leq f(x)$ we know that every local minimizer in $H^k(x)$, is also below f. Hence the global minimizer of the piecewise linear support function $H^k(x)$ must lie below the global minimum x^* of $f(x)$. The global minimizer

$$z^* = \min_{a \leq x \leq b} H^k(x) \tag{2.5}$$

of the piecewise linear support function is therefore a lower bound of $f(x)$ with a tolerance $\Delta_k = f(x^*) - z^*$. Figure 2.1 shows the piecewise linear support function $H^k(x)$ for the above example with $k = 4$.

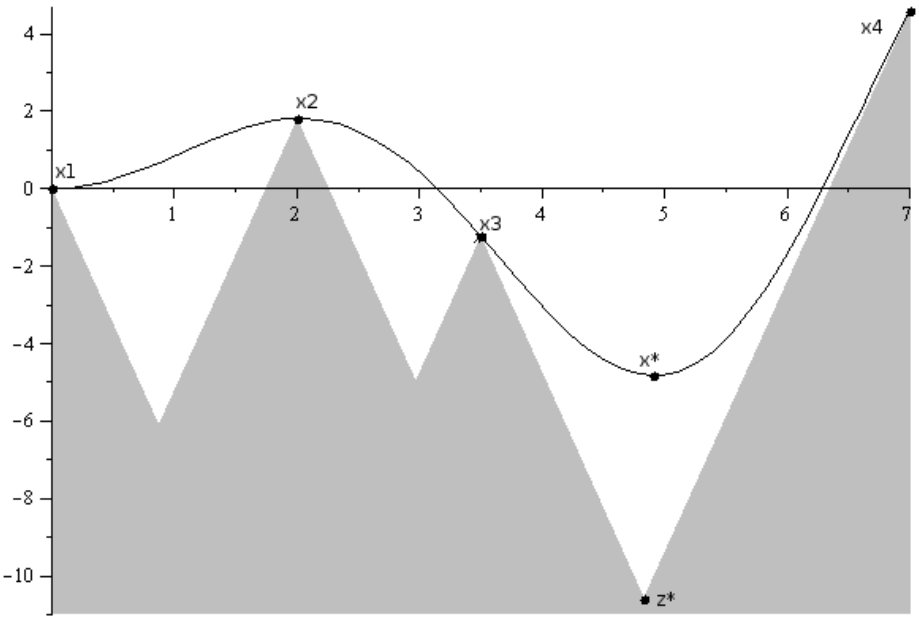


Figure 2.1: The saw-tooth underestimate for the Lipschitzian function $f(x)$ with 4 sample points.

From the above explanation, we can see that Δ_k is small when k is big. This observation suggests an iterative approach for building the saw-tooth underestimate H^k. Pijavskii and Shubert introduced a sequential method to approximate the global minimum of a Lipschitzian function [15, 21].

Consider a function f in $\mathbb{R}$ with Lipschitz constant L. We want to minimize the function by solving

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad x \in [\alpha, \beta]. \end{aligned} \tag{2.6}$$

The Pijavskii-Shubert method starts with the evaluation of the domain boundaries α, β. For each point, we can now construct an affine underestimate. The intersection of the two support vectors is solved as a simple linear system. The function f is then evaluated again at the new intersection point and two new support vectors are constructed that result in two new intersection points. At each following iteration, the minimum of all intersection points that have not yet been evaluated forms the candidate for a function evaluation. The procedure is continued until the underestimate model converges to the global minimum of f within a tolerance ϵ or until the maximum number of iterations k_{max} is reached. The convergence of the algorithm is shown in [15, 21]. The detailed steps are outlined in Algorithm 2.1

Algorithm 2.1: Pijavskii-Shubert Method

Input: Function $f(x)$, starting points α, β, Lipschitz constant L
Output: Global minimum f_{best}
begin
  $k \leftarrow 0$
  $d_{best} \leftarrow -\infty$
  $f_{best} \leftarrow \min\{f(\alpha), f(\beta)\}$
  Solve $d = -Lx^* + f(\alpha) = Lx^* + f(\beta)$
  $x_k \leftarrow x^*$ and add (d, x_k) to H^k
  repeat
    $k \leftarrow k + 1$
    $(d, x_k) \leftarrow \min_d H^{k-1}$
    $H^k \leftarrow H^{k-1} \setminus (d, x_k)$
    $d_{best} \leftarrow d$
    if $f(x_k) < f_{best}$ **then**
      $f_{best} \leftarrow f(x_k)$
    end
    $x_L \leftarrow LeftNeighbour(x_k)$
    Solve $d = -Lx^* + f(x_L) = Lx^* + f(x_k)$
    $x_k \leftarrow x^*$ and add (d, x_k) to H^k
    $x_R \leftarrow RightNeighbour(x_k)$
    Solve $d = -Lx^* + f(x_k) = Lx^* + f(x_R)$
    $x_k \leftarrow x^*$ and add (d, x_k) to H^k
  until $k \geq k_{max}$ *or* $f_{best} - d_{best} \leq \epsilon$
end

In the above method, an iteration creates exactly two new local minima in H^k. Hence, the algorithm has a linear running time and is therefore very efficient for one-dimensional problems. However, in the extension of the Pijavskii-Shubert method to $\mathbb{R}^n$, the minimization of H^k becomes an NP-hard problem, since the number of hypercone intersections grows exponentially with the dimension. Furthermore, the algorithm involves the computationally difficult problem of solving n-dimensional hypercone intersections [14].

In [2], based on the results of *abstract convexity* [17], Andramonov et al. propose a generalized version of the Cutting Plane Method from convex analysis for the optimization of non-convex functions. They introduce a special case of the Generalized Cutting Plane Method, the *Cutting Angle Method* (CAM). The CAM minimizes *Increasing Positively Homogeneous* (IPH) functions of degree one by creating a saw-tooth underestimate similar to the Pijavskii-Shubert method. Beliakov and Batten introduce a fast algorithm for the CAM [7] that significantly reduces the minimization of H^k. The key part of the algorithm is the formulation of the minimization of H^k as a combinatorial problem. With the *Extended Cutting Angle Method* (ECAM), Beliakov introduces a version of the CAM that uses a different class of support functions. As a result, the ECAM is more accurate and faster than the CAM

[6]. The Pijavskii-Shubert algorithm presented previously is a special case of the ECAM in one dimension.

In section 2.2 we explain the Cutting Plane Method. In Section 2.3 we explain the generalization of the Cutting Plane Method in abstract convex analysis and in Section 2.4 the CAM. Section 2.5 deals with the extended version of the CAM, the ECAM.

2.2 Convexity and the Cutting Plane Method

The Cutting Plane Method from convex analysis is an algorithm introduced by Kelley [12] to optimize convex functions. In order to understand the algorithm, we will introduce briefly the notions of convexity and the subdifferential. Since we are interested primarily by functions that are Lipschitz continuous, we will only consider functions that are finite. We start with the introduction of convex sets.

Definition 1. *A set $X \subset \mathbb{R}^n$ is called convex if $\forall x_1 \in X$ and $\forall x_2 \in X$ it contains all points*

$$\alpha x_1 + (1-\alpha)x_2, \qquad 0 < \alpha < 1$$

Now consider a finite function f.

Definition 2. *The epigraph of a function f is defined by*

$$\text{epi} f = \{(x, v) \in \mathbb{R}^n \times \mathbb{R} : v \geq f(x)\}.$$

One can imagine the epigraph of a one dimensional function as being the surface that lies above the function. Combining Definition (1) and (2), we can define a convex function.

Definition 3. *A function f is called convex if epi f is a convex set.*

An analytical formulation of the above definition is given in the following Lemma.

Lemma 1 ([18])**.** *A function f is convex if and only if for all x_1 and x_2 and for all $0 \leq \alpha \leq 1$ we have*

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1-\alpha) f(x_2).$$

The proof is omitted here. We know that if a function is not differentiable at a point, we do not have a gradient at that point. However, in convex analysis, a generalized version of the gradient is introduced. It is called the *subgradient*. The subgradient of a convex function f at some point x is defined as follows.

Definition 4. *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function and let $x \in \text{dom} f$. A vector $g \in \mathbb{R}^n$ such that*

$$f(y) \geq f(x) + \langle g, y - x \rangle, \forall y \in \mathbb{R}^n$$

is called the subgradient of f at x.

Definition 5. *The set of all subgradients of f at x is called the subdifferential of f at x, denoted by $\partial f(x)$.*

Now consider the subgradient $g \in \partial f(x)$. In geometrical terms, the inequality in (4) means that the affine function $h(x) = f(x) + \langle g, y - x\rangle$ lies below the function $f(x)$. Hence, $h(x)$ is an *affine underestimate* of f at x. Notice that in [18] it is shown that for a convex function, $\partial f(x)$ is a nonempty, convex, closed and bounded set. Moreover, if the function f is differentiable at x, then $\partial f(x)$ contains only one element which is the gradient of f.

With the above definitions, we can now introduce Kelley's Cutting Plane Method. Consider the problem

$$\begin{aligned} &\min f(x) \\ &\text{s.t.} \quad x \in D \end{aligned} \tag{2.7}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and D is a convex and compact set. The Cutting Plane Method starts with taking the subgradient of f at some point x_1. An affine underestimate of f at x_1 is then constructed with

$$h^1(x) = f(x_1) + \langle g_1, x - x_1\rangle.$$

The underestimate $h^1(x)$ is then minimized over the domain D. The resulting minimizer x^* is taken as the starting point for the next iteration. At $x_2 = x^*$, a new affine underestimate is created by using the subgradient of f at x_2. Hence we have

$$h^2(x) = f(x_2) + \langle g_2, x - x_2\rangle.$$

The two underestimates are then combined into a piecewise linear underestimate

$$H^2(x) = \max\{h^1(x), h^2(x)\}.$$

Minimizing the piecewise linear underestimate results in the point $x^* = \min_{x\in D} H^2(x)$ which is the starting point for the next iteration. The procedure continues iteratively. At k iterations, we therefore have a piecewise lower approximation to the function f of the form

$$H^k(x) = \max_{1\le i\le k}\{f(x_i) + \langle g_i, x - x_i\rangle\}.$$

And hence

$$x_{k+1} = \min_{x\in D} H^k(x)$$

is the location of the next function evaluation. Since $H^k(x)$ is a piecewise linear function, the minimum is easily obtained by using linear programming techniques. The algorithm stops when $f(x_{k+1}) = h^k(x_{k+1})$ or $f(x_{k+1}) - h^k(x_{k+1}) \le \epsilon$ for some given value ϵ. The convergence of the algorithm is shown in [18].

2.3 Abstract Convexity and the Generalized Cutting Plane Method

In 2.2 we showed that for the optimization of a convex function, one can build a piecewise affine convex function and find the global minimum by solving a linear program. The algorithm is based on the conclusion from convex analysis that a convex function forms the upper envelope of its affine minorants. The Generalized Cutting Plane Method takes a similar approach for non-convex functions. The method is based on the results from *abstract convex* analysis that an abstract convex function is the upper envelope of its *generalized affine minorants* (sufficiently simple minorants other than affine) [17]. In the case of a Lipschitz function, the support functions are the saw-tooth or min-max type functions. The restriction of certain types of abstract convex functions allows for a specific choice of support functions.

In [17], an abstract convex function is defined as follows.

Definition 6. *Given a set of real-valued functions H defined on a set $X \subset \mathbb{R}^n$, a function $f : X \to \mathbb{R}$ is abstract convex with respect to H (or H-convex) if there exists a set $U \subset H$ such that $\forall x \in X$*

$$f(x) = \sup\{h(x) : h \in U\}.$$

Now all $h(x) \in H$ that are generalized affine minorants of f are called the support set of f with respect to the set of functions $H : \text{supp}(f, H) = \{h \in H, h \leq f\}$. Analogous to convex analysis, we have the following definition.

Definition 7. *The H-subdifferential of f at point x_0 is defined by*

$$\partial_H f(x_0) = \{h \in H : h(x) - h(x_0) \leq f(x) - f(x_0), \forall x \in X\}$$

Now consider an *Increasing Positively Homogeneous* function (IPH) of degree one

$$\text{IPH} = \{f : \forall x, y \in \mathbb{R}^n_+ : x \geq y \Rightarrow f(x) \geq f(y); \forall x \in \mathbb{R}^n_+, \lambda \in \mathbb{R}_{++} : f(\lambda x) = \lambda f(x)\} \quad (2.8)$$

where $\mathbb{R}^n_+$ denotes the cone of vectors with non-negative components and $\mathbb{R}_{++}$ denotes the set $(0, \infty]$. Andramonov et al.[2] show that a function $f : \mathbb{R}^n_+ \to \mathbb{R}$ is H-convex if and only if f is IPH of degree one. Any Lipschitz function restricted to a set $X \subset \mathbb{R}^n_+$ can be extended to an IPH by adding a constant $C \geq 2L$. The addition of a constant does not change the minimum of the function.

Let f be an IPH function of degree one on a compact convex subset $X \in \mathbb{R}^n$. In order to minimize f, the generalized Cutting Plane algorithm, analogous to the Cutting Plane algorithm, iteratively creates a piecewise function $H^k(x)$ of H-minorants of f with the restriction that $x \in X$. At each iteration k, the algorithm solves the following problem

$$\begin{aligned} &\min H^k(x) \\ &\text{s.t.} \quad x \in D. \end{aligned} \quad (2.9)$$

As in the Cutting Plane Method, the solution of (2.9) serves as the starting point for the next iteration. Andramonov et al. show that if one uses the class of support functions of type

$$h(x) = \min_{i=1,\dots,n} l_i x_i, \qquad l_i \in \mathbb{R}^n_+, x \in S, \tag{2.10}$$

then the algorithm will converge to the global minimum of f. The algorithm as outlined in [17] is presented below (Algorithm 2.2).

Algorithm 2.2: Generalized Cutting Plane Method

Input: Function $f(x)$, starting point $x^* \in X \subseteq \mathbb{R}^n$
Output: Global minimum f_{best}
begin
  $k \leftarrow 1$
  $x^k \leftarrow x^*$
  $f_{best} \leftarrow f(x^k)$
  repeat
    Calculate $h^k \in \partial_H f(x^k)$ such that $h(x^k) = f(x^k)$ (H-subgradient of f at x^k)
    $H^k(x) \leftarrow \max_{i=1,\dots,k} h^i(x), \forall x \in X$
    $x^* \leftarrow \min H^k(x)$
    $k \leftarrow k + 1$
    $x^k \leftarrow x^*$
    $f_{best} \leftarrow \min\{f(x^k), f_{best}\}$
  until $k \geq k_{max}$ *or* $f_{best} - H^k(x^*) < \epsilon$
end

2.4 The Cutting Angle Method

A special case of the generalized Cutting Plane Method is the Cutting Angle Method (CAM). In the CAM, the restricting set X of a function f in $\mathbb{R}^n$ is the unit simplex $S \in \mathbb{R}^n$. The CAM calculates a function $h \in H$ at a point x^* (H-subgradient) with the following formula

$$h^k(x) = \left(\frac{f(x^k)}{x^k}\right) = \left(\frac{f(x^k)}{x^k_1}, \frac{f(x^k)}{x^k_2}, \dots, \frac{f(x^k)}{x^k_n}\right), \quad x_i \neq 0, i = 1, \dots, n. \tag{2.11}$$

Functions like (2.11) are called *support vectors* in the CAM. The CAM uses the support vectors (2.11) to build the lower estimate of f on the unit simplex S similar to the Cutting Plane Method of Section 2.2. The original CAM presented in [2] solved the relaxed problem (2.9) with a standard integer programming approach. This made it possible to deal with about 100 support vectors [4]. In [7], Beliakov and Batten transform the relaxed problem into a combinatorial problem (as explained in Section 2.5) which enables the CAM to process hundred of thousands of support vectors. The saw-tooth underestimate of the CAM using

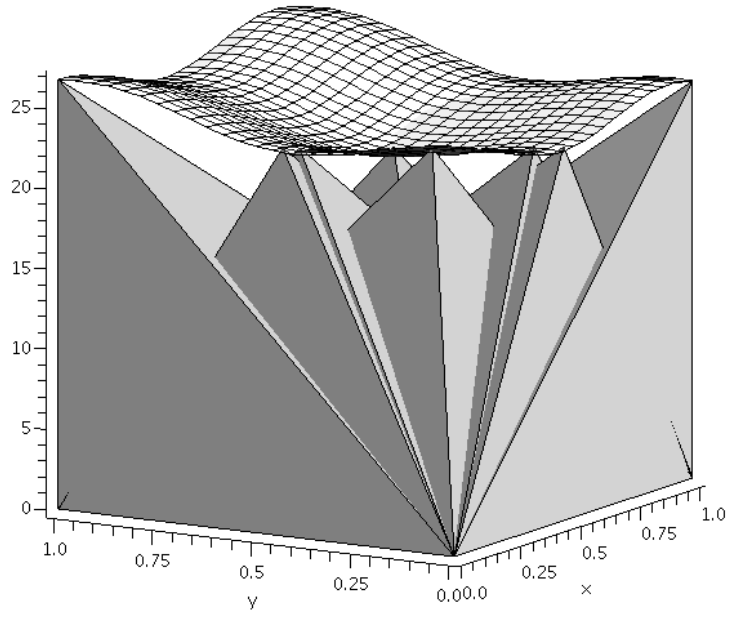


Figure 2.2: Part of a saw-tooth underestimate with support functions of type (2.11) for an objective function of two variables in the CAM.

support functions of type (2.11) in $\mathbb{R}^3$ is shown in Figure 2.2.

Beliakov points out that there are some disadvantages of the CAM when support functions of type (2.10) are used [6]. One disadvantage is that the Lipschitz constants are different for every $h(x)$ in H^k and can become very large along the rays close to the boundary of the unit simplex S (see Figure 2.2). Also, the transformation of a Lipschitz function to IPH may result in a loss of accuracy. Beliakov proposes a new type of support function

$$h(x) = \min_x (Cx + b), \qquad C > 0, \quad x, b \in \mathbb{R}^n : \sum_{i=1}^{n} x_i = 1 \tag{2.12}$$

where C is a constant greater or equal to the Lipschitz constant L of f. Specifically, at iteration k with $b = f(x^k) - Cx^k$, the above can be rewritten as

$$h^k(x) = \min_x (f(x^k) - C(x^k - x)) \tag{2.13}$$

In the one-dimensional case, the set $H^k = \{h_1(x), \ldots, h_k(x)\}$ is the saw-tooth underestimate of the Pijavskii-Shubert method. However, the generalized version of the Pijavskii-Shubert

underestimate, namely,

$$H^k(x) = \max_{i=1,\dots,k} (f(x^i) - C \parallel x^i - x \parallel) \tag{2.14}$$

is different from what Beliakov proposes. He replaces the norm $\parallel \cdot \parallel$ with a polyhedral distance function d_P so that (2.14) becomes

$$H^k(x) = \max_{i=1,\dots,k} (f(x^i) - CdP(x, x^i)), \qquad \sum_{i=1}^{n} x_i = 1. \tag{2.15}$$

Beliakov shows that classes of functions which are abstract convex with respect to (2.15) are the same classes of functions which are abstract convex with respect to (2.14). The use of the polyhedral distance makes it possible to solve the relaxed problem very fast by enumerating all local minima of H^k. This was already done in [5] for the fast implementation of the CAM. Beliakov also proves that all Lipschitz functions are abstract convex with respect to (2.13)[6]. This allows an extension of the CAM by applying (2.15) as an underestimate for Lipschitz functions.

2.5 The Extended Cutting Angle Algorithm

2.5.1 Polyhedral Distance and Support Functions

The definition of a polyhedral distance is given in [6]. Beliakov points out that polyhedral distances are special cases of Minokowski gauges and exhibit therefore the following property

$$A \parallel x - y \parallel \leq d_p(x, y) \leq B \parallel x - y \parallel, \qquad \forall x \in \mathbb{R}^m.$$

This means that for any Lipschitz function, the polyhedral distance is bounded by the Lipschitz property. Beliakov then proves that, with the help of an auxillary variable, support functions used in (2.15) are equivalent to support functions in (2.14). In other words, in the case of a Lipschitz underestimate, one can replace the norm by the polyhedral distance. For the details of the proof, we refer to [6]. The auxillary equation in the case of a function in $\mathbb{R}^m$ is defined as $x_{m+1} = 1 - \sum_{i=1}^{m} x_i$. This results in a constraint for the variables of the form

$$\sum_{i=1}^{n} x_i = 1 \tag{2.16}$$

where $n = m + 1$. The additional variable and the constraint extends the function domain and restricts it to the hyperplane that is spanned by the unit simplex. Due to the equivalence of (2.15) and (2.14), it is now possible to use (2.13) as a support function to build the saw-tooth underestimate. Beliakov shows that Lipschitz functions in $\mathbb{R}^m$ are abstract convex with respect to (2.13) on the extended domain $\mathbb{R}^n$ with $n = m + 1$, subject to (2.16)[6].

As a result, the extension of the CAM with the support functions of type (2.13) allows the minimization of a Lipschitz function on a compact set D. The proof of convergence for the *Extended Cutting Angle Method* (ECAM) is omitted here. We refer to [6] for the details.

2.5.2 Saw-tooth and Calculation of Local Minima

As mentioned above, due to the use of a polyhedral distance d_p instead of the norm, we build the saw-tooth underestimate with functions of type (2.13). If we expand (2.13), we get

$$h^k(x) = \min_x (f(x^k) - Cx^k + Cx).$$

We reformulate the equation as

$$h^k(x) = \min_x \left(\frac{f(x^k)}{C} - x^k + x \right).$$

This leads to the formulation of the support vector

$$l_i^k = \frac{f(x^k)}{C} - x_i^k \tag{2.17}$$

and the final formulation of the support function

$$h^k(x) = \min_x C(l^k + x). \tag{2.18}$$

To minimize the saw-tooth underestimate, we first have to enumerate all minimizers that result from the intersections of the support functions (2.18). The final form of the support functions allows us to formulate the enumeration of the minimizers as a combinatorial problem which is the foundation of the fast implementation of the CAM and ECAM.

A minimizer in the relative interior riD needs to fulfill the necessary and sufficient conditions listed in [10] applied to the support function (2.18). This results in the following proposition.

Proposition 1 ([6]). *A necessary and sufficient condition for a point $x^* \in riD$ to be a local minimizer of $H^K = \max_{k=1,\ldots,K} h^k$ and h^k given by (2.18), is that there exists an index set $J = \{k_1, k_2, \ldots, k_n\}$ of cardinality n, such that*

$$d = H^K(x^*) = C(l_1^{k_1} + x_1^*) = C(l_2^{k_2} + x_2^*) = \ldots = C(l_n^{k_n} + x_n^*),$$

and $\forall i, i = 1, \ldots, n, C(l_i^{k_i} + x_i^*) < C(l_j^{k_i} + x_j^*), j \neq i$.

Again, for the proof of the above proposition, we refer to [6]. In geometrical terms, the above proposition can be explained as follows. The point x^* represents a point of intersection of n polyhedral cones in the saw-tooth underestimate H^k. A cone P^k is defined by n hyperplanes with slope C. Therefore x^* is the intersection of n edges of those hyperplanes, each with slope C in the direction of x_i. Thus, the i-th edge of a cone is denoted by $C(l_i + x_i^*)$.

In order to find the point of intersection of n cones, according to Proposition 1, we list the n cones as an ordered index set J with indices $\{k_1, k_2, \ldots, k_n\}$, one index per cone. We

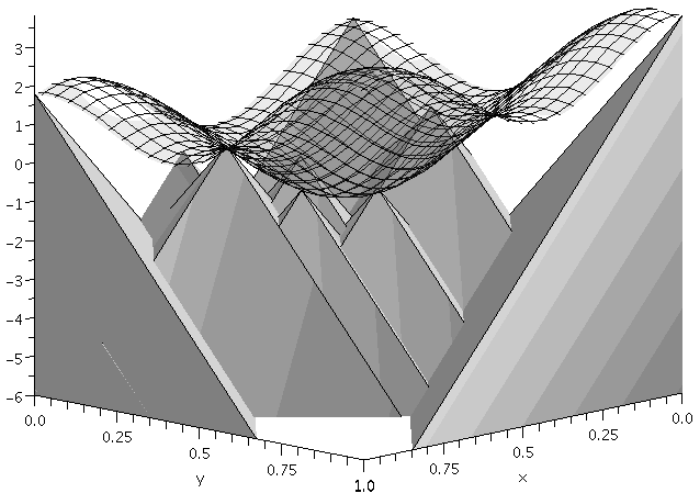


Figure 2.3: Part of a saw-tooth underestimate with support functions of type (2.13) for an objective function of two variables in the ECAM.

list each cone as a set of n support vectors, representing the n edges of the cone. We write this combination of vectors in matrix form

$$L = \begin{bmatrix} l_1^{k_1} & l_2^{k_1} & \dots & l_n^{k_1} \\ l_1^{k_2} & l_2^{k_2} & \dots & l_n^{k_2} \\ \vdots & \vdots & \ddots & \vdots \\ l_1^{k_n} & l_2^{k_n} & \dots & l_n^{k_n} \end{bmatrix}. \tag{2.19}$$

From Proposition 1 we can solve now for x^* with

$$x_1^* = \frac{d}{C} - l_1^{k_1}, x_2^* = \frac{d}{C} - l_2^{k_2}, \dots, x_n^* = \frac{d}{C} - l_n^{k_n}. \tag{2.20}$$

Because of constraint (2.16), if we add the individual $x_i, i = 1, \dots, n$ together, we can set the sum equal to 1. It follows that

$$\sum_{i=1}^{n} x_i^* = \sum_{i=1}^{n} \left(\frac{d}{C} - l_i^{k_i} \right) = d \sum_{i=1}^{n} \frac{1}{C} - \sum_{i=1}^{n} l_i^{k_i} = d \sum_{i=1}^{n} \frac{1}{C} - Trace(L) = 1.$$

Solving for d, we get

$$d = \frac{Trace(L) + 1}{\sum_{i=1}^{n} \frac{1}{C}}. \tag{2.21}$$

Once we have d, we can calculate the position of the minimizer with (2.20). The last part of Proposition 1 also requires that $d = C(l_i^{k_i} + x_i^*) = C(l_j^{k_j} + x_j^*) < C(l_j^{k_i} + x_j^*), j \neq i$ which results in the constraint

$$l_i^{k_i} < l_i^{k_j}. \tag{2.22}$$

Constraint (2.22) means that the diagonal entries of L must be *dominated* by the entries in their column. In geometrical terms, this means that the minimizer cannot lie below an intersection of other edges of the same cones. The last constraint for the minimizer x^* says that any edge of a cone that does not belong to the intersection L should not be above the minimizer. Hence

$$\forall r \notin L, \exists i : L_{ii} \geq l_i^{k_i}. \tag{2.23}$$

Beliakov groups constraints (2.22),(2.23) and equations (2.21), (2.20) in the following Theorem.

Theorem 1 ([6])**.** *Let the support vectors $l^k, k = 1, \ldots, K$ be defined by (2.17). Let x^* be a local minimizer of H^K in riD and $d = H^K(x^*)$. Then the matrix L defined by (2.19) and corresponding to x^* enjoys the following properties:*

(1) $l_i^{k_i} < l_i^{k_j}, i = 1, \ldots, n, j = 1, \ldots, n, i \neq j$;

(2) $\forall r \notin L, \exists i : L_{ii} \geq l_i^r, i = 1, \ldots, n$;

(3) $d = \frac{Trace(L)+1}{\sum_{i=1}^n \frac{1}{C}}$;

(4) $x_i^* = \frac{d}{C} - l_i^{k_i}$.

Theorem 1 allows us to identify a minimizer with a combination of n support vectors. To find all the minima of H^K at iteration K, one could test all possible permutations of n support vectors from the set of K vectors $\{l^1, l^2, \ldots, l^K\}$. This amounts to a running time complexity of $\mathcal{O}\left(\binom{K}{n}\right)$ which is a polynomial running time.

The next section presents another approach that allows the enumeration of the local minima in a running time complexity of $\mathcal{O}(\log q)$ where q is the number of local minima in H^K.

2.5.3 Fast Enumeration of Local Minima

In the fast implementation of the CAM, Beliakov showed how to build combinations of L incrementally, by starting with an initial set $L_0 = \{l^1, l^2, \ldots, l^n\}$ [7]. The idea is similar to the top down approach in *Dynamic Programming*. A newly created support vector l^k is added and the new underestimate is calculated by

$$H^k(x) = \max\{H^{k-1}(x), h^k(x)\}, \qquad k = n+1, \ldots, K. \tag{2.24}$$

Suppose we store all the minimizers in a tree structure T. If we start with L_0 as a root node, we calculate the intersection point x^* from L_0 using condition (4) in (1). We then add

a new cone $h^{n+1}(x^*)$. With (2.17), we compute the new support vector l^{n+1}. Now since we want to calculate the intersection of the new cones with the initial n cones in L_0, we create n permutations of the root node L_0 with l^{n+1} which represents the n new intersections of H^n with the new cone.

For the next iteration, since the new cone $h^{n+1}(x^*)$ lies above the intersection L_0, it is clear that L_0 is not part of H^{n+1} anymore. Hence, H^{n+1} consists only of the leaves of T. We denote $V^n = Leaves(T^n)$ as the n leaves of T^n. Analogous to (2.9) we find the lowest minimizer L^* in V^{n+1} with the help of equation (4) in Theorem 1. As before, a new cone $h^{n+2}(x^*)$ is formed and the new intersections are calculated through permutations of l^{n+2} with all the leaves L and verification of condition (1) and (2) of those permutations.

Assume, however, that at iteration k we permute l^k with all the leaves V^k. The running time complexity becomes $\mathcal{O}\left(V^k\right)$. Furthermore, some of those permutations would also lie below H^k. Fortunately, we do not need to permute l^k with all of the leaves V^k. Beliakov showed that it is only necessary to permute a vector l^k with leaves that fail condition (2) in Theorem 1[7]. Indeed, looking at Theorem 1, the only new combinations of leaves with l^k that could satisfy condition (2) are the ones whose parent node fails condition (2) because of l^k. This means we do not have to check leaves of parents that satisfy condition (2).

Since we store all the nodes in a tree structure, starting with the root node L_0, we can perform a depth-first search and check for each node L if it satisfies condition (2) with $l^r = l^k$. The root node will always fail this condition, since otherwise we would not have any leaves. However, as soon as we encounter a node that satisfies condition (2), we can discard it and all of its children. We can prune this branch of the tree. This reduces the running time complexity from $\mathcal{O}\left(V^k\right)$ to $\mathcal{O}\left(\log(V^k)\right)$. Also, testing for condition (2) can be done in $\mathcal{O}\left(1\right)$ operations since we need to compare only the vector at position i in L where the parent failed against l^r.

The detailed steps of the above procedure are outlined in Procedure `UpdateTree`.

Procedure `UpdateTree`(L, l^k, T^{k-1}, V^{k-1})

Input: Node L, new support vector l^k, tree T^{k-1} with leaves V^{k-1}
Output: Tree T^k with leaves V^k
begin
  if $L_{ii} < l_i^r$ **then**
    if L is not a leaf **then**
      for $i \leftarrow 1$ **to** n **do**
        $child \leftarrow$`Child(`L,i`)`
        `UpdateTree(`$child$, l^k, T^{k-1}, V^{k-1}`)`
    else
      for $i \leftarrow 1$ **to** n **do**
        $child \leftarrow L$
        $child_i \leftarrow l^k$
        if $child$ satisfies condition (1) **then**
          $d \leftarrow \frac{Trace(child)+1}{\sum_{i=1}^{n} \frac{1}{C}}$
          if $d < f_{best}$ **then**
            Add $child$ as new leaf to L
          else
            Delete $child$
        else
          Delete $child$.
  else
    $T^k \leftarrow T^{k-1}$
    $V^k \leftarrow V^{k-1}$
end

2.5.4 Starting Points

In [6], Beliakov shows that the ECAM evaluates f only at points x inside the *search domain* A of the ECAM. The set A is defined by the combination L_0 of the first n support vectors. For $f(x) : \mathbb{R}^n \to \mathbb{R}$, A is therefore determined by the following n inequalities

$$C(x_j^* - x_j^{k_j}) < C(x_i^* - x_i^{k_i}), \qquad i = 1, \ldots, n, j = 1, \ldots, n, i \neq j. \tag{2.25}$$

For solving the problem (2.9), we want the ECAM to find the minimizers in the compact set D, where D is a polytope. Hence D needs to lie inside A such that $D \subset A$. However, we want the distance d_p from any point x on the boundary ∂A of A to any point $y \in D$ to be greater than the distance from x to the closest vertex defining D. In [6], this is defined as

$$\forall x \in \partial A : \quad \max_{k=1,\ldots,n} f(x^k) - Cd_p(x, x^k) \geq \max_{y \in D} f(y) - Cd_p(x, y). \tag{2.26}$$

In practice, this is achieved by taking the first n points $x^1, x^2, \ldots, x^n$ sufficiently far apart. In general, we choose $x^k, k = 1, \ldots, n$ such that x^k is roughly a distance of $\frac{1}{4}$ of the length of the corresponding axis in D apart from the next vertex defining D.

2.5.5 Constraints

In section 2.5.2 and 2.5.3 we showed, based on Proposition 1, how to find and enumerate the local minimizers. However, Proposition 1 applies only to minimizers in the relative interior riD of the compact set D. But for the ECAM to be able to converge, we also need to enumerate the minimizers on the boundary ∂D of D.

In procedure `UpdateTree` in section 2.5.3, we do not check if a newly generated intersection lies inside or outside D. Hence, if at some iteration k we are calculating the position $x^* \in A(L^*)$ of the lowest minimizer L^* in V^k, we either have $x^* \in D$ or $x^* \notin D$. In the latter case, we want to see if there is a minimizer x^{**} on the boundary ∂D in the direction of $x^* \in A(L^*) \setminus D$. Note that here $A(L^*)$ is a polytope that is a partition of $A(L_0)$. To find x^{**} we solve the following constrained problem

$$\begin{aligned} &\min \max_{i=1,\ldots,n} C(x_i + l_i^{k_i}) \\ &\text{s.t.} \quad x \in D \cap A(L^*). \end{aligned} \tag{2.27}$$

Again, $D \cap A(L^*)$ is a polytope and with the addition of a helper variable, we can transform (2.27) into a Linear Programming problem and solve for x^{**}

$$\begin{aligned} &\min v \\ \text{s.t.} \quad &x \in D \cap A(L^*), \\ &C(x_i + l_i^{k_i}) \leq v, \qquad i = 1, \ldots, n. \end{aligned} \tag{2.28}$$

When there are large numbers of support vectors, solving the above linear program in each iteration becomes very inefficient. Therefore, in practice, we use a projection as an approximation of the minimizer on ∂D. If $x^* \notin D$, x^* is projected onto ∂D to get x^{**}. In the ECAM algorithm, we then evaluate $f(x^{**})$ and if $f(x^{**}) < f_{best}$, we record x^{**} and let $f_{best} = f(x^{**})$. The function value $f(x^{**})$ and the point x^{**} are then used to generate the new support vector and $\{x^{**}, d = \infty\}$ is moved back into V^k.

2.5.6 The ECAM Algorithm

We are now able to explain the full ECAM algorithm. In section 2.5.3, we introduced the tree T of all nodes and the set of all leaves V of this tree. In the implementation, the tree T^{k-1} and T^k occupy the same memory space. The tree T^k is in fact the same tree as T^{k-1} with possible children added to the leaves. The set V is implemented as a priority queue that allows for removal of its elements. The full ECAM algorithm is outlined in Algorithm 2.4

Algorithm 2.4: Extended Cutting Angle Method

Input: Function $f(x)$, Lipschitz constant C, k_{max}, domain $D \subseteq \mathbb{R}^n$
Output: Global minimum f_{best}
begin
  Choose $x^k, k = 1, \ldots, n$ according to (2.26)
  for $i \leftarrow 1$ **to** n **do**
    construct l^i according to (2.17)
    $L_i \leftarrow l^i$
  end
  sort vectors in L in order to satisfy condition (2) of Theorem 1
  $L_0 \leftarrow L$
  $T^n \leftarrow L_0$
  $V^n \leftarrow L_0$
  $k \leftarrow n$
  $f_{best} = \min_{k=1,\ldots,n} f(x^k)$
  repeat
    $L^* \leftarrow \min V^k$
    $x^* \leftarrow x^*(L^*)$ using condition (4) of Theorem 1
    if $x^* \notin D$ **then**
      Solve problem (2.28)
    end
    $f^* \leftarrow f(x^*)$
    $f_{best} \leftarrow \min\{f(x^k), f_{best}\}$
    $k \leftarrow k + 1$
    Form l^k using $l_i^k = \frac{f(x^k)}{C} - x_i^k$
    $T^k, V^k \leftarrow$ `UpdateTree`$(L^*, l^k, T^{k-1}, V^{k-1})$
  until $k \geq k_{max}$ *or* $f_{best} - d(L^*) < \epsilon$
end

Chapter 3

Radiation Therapy and the Beam Angle Optimization

3.1 External Beam Radiation Therapy

Despite scientific achievements in cell biology and biochemistry, cancer remains one of the biggest challenges for modern Health Care departments and organizatinos. In Canada alone, an estimated 159,900 new cases of cancer occured in 2007 [1]. Radiation therapy, along with surgery and chemotherapy, is one of the most important treatment methods for cancer. More than 50% of all cancer patients will receive radiation therapy at some stage during their treatment.

A common method of radiation treatment is the external beam radiation therapy. In most cases of external beam radiation therapy, a photon or electron beam is formed outside the patient body and aimed at the tumor. As the beam passes through the tumor tissue, it creates an ionizing radiation that either directly destroys the DNA structure of the tumor cells or indirectly damages the cell structures through a reaction with other molecules. In either case, the damage results eventually in the death of the cells. Since the beam also traverses the tissue that surrounds the tumor, the healthy cells that lie in the way of the beam are damaged too. One advantage of radiation therapy is that healthy tissue cells are more likely to recover from the damage than cancerous cells. A correctly prescribed radiation dose is therefore able to destroy the tumor cells while at the same time allowing healthy tissue to recover. However, the healthy tissue damage has important implications on side effects, especially for cells that belong to critical organs. In external beam radiation therapy, it is therefore desirable to minimize the damage to the *Organs At Risk* (OAR) and *Other Healthy Tissue* (OHT) that occurs during the delivery of the prescribed radiation dose to the tumor.

A simple approach to minimize the radiation dose on the OHT and the OAR is the use of multiple beams from different angles. The beams are all aimed at the *Planning Target Volume* (PTV). The intersection of the beams at the PTV creates a higher dose in the tumor cells than in the healthy cells that lie in the rays of the beam. Figure 3.2 shows the intersection of four beams and the corresponding dose intensity. As one can see from Figure 3.2 that one gains more flexibility to shape the intersection of the beams if more beams are available. Most patients receive radiation treatment on a so called medical linear accelerator. The patient lies on a table, and the head of the linear accelerator is turned around the patient for each beam position. This means that the head of the linear accelerator needs to be turned for each beam angle. Hence, the disadvantage of a large number of beams is that

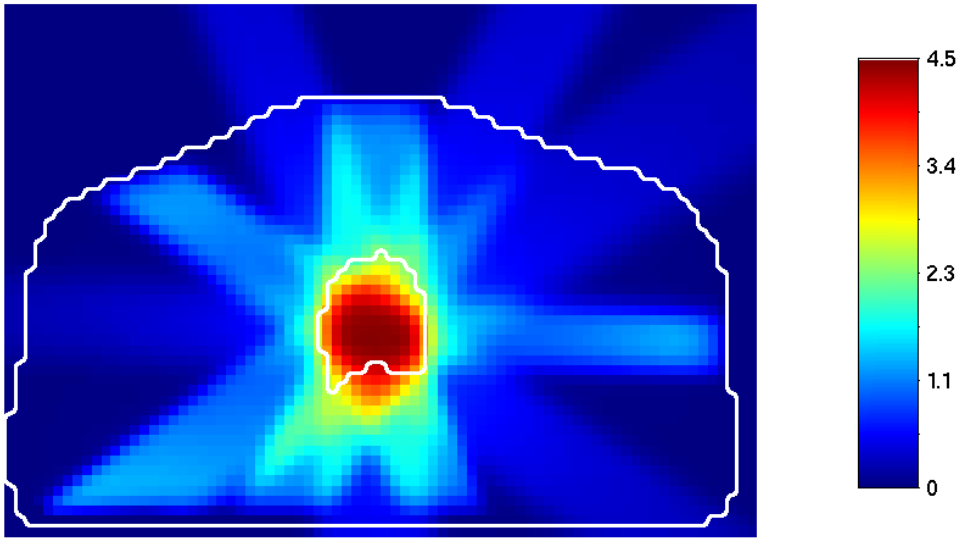


Figure 3.1: A cross-section of the pelvis with the dose distribution matrix of six beams at different angles in a radiation treatment for prostate cancer. In this example, no beam intensity modulation is performed and each beam has the same weight.

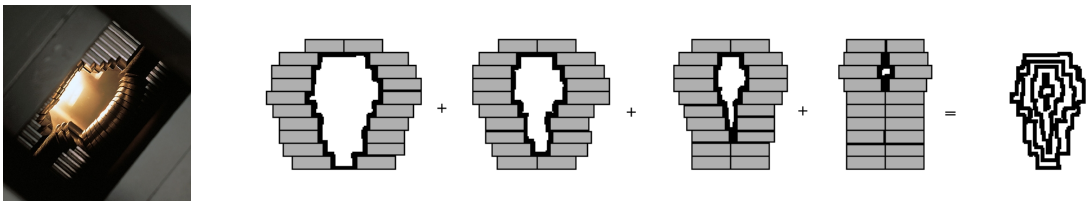
the treatment time increases significantly.

3.2 Intensity Modulated Radiation Therapy

In *Intensity Modulated Radiation Therapy* (IMRT), the beam is shaped in the direction of the *Beam's Eye View* (BEV) and the intensity is modulated in order to achieve a dose distribution that closely follows the shape of the tumor and avoids the *OAR*. In order to form the beam in the two mentioned directions, either custom made solid compensators are attached to the head of the linear accelerator or a *Multileaf Collimator* (MLC) is used.

An MLC is a device that is attached to the head of the linear accelerator and usually consists of a rectangular opening. A set of metal leaves are attached at two opposing sides of the opening controlled by a computer, these leaves can be inserted into the opening to block parts of the beam. For a picture of an MLC, see Figure 3.2a[1]. In an IMRT treatment with MLC, the leaves are usually inserted until the BEV takes the shape of the tumor. The beam is then fired a first time for a specified amount of time which is called a *Monitor Unit*. The leaves configuration is then changed in order to fire the beam a second time. The leaves configuration should now block the parts of the tumor that are rather thin. The reason is,

[1]Figure 3.2a with permission from Elekta AB. ©Copyright 2008 Elekta AB (publ). All rights reserved.



(a) (b)

Figure 3.2: A multileaf collimator and its use in the segmentation of a dose distribution.

that those parts usually receive enough radiation with the first monitor unit and a second shot would only damage the healthy tissue in the line of the beam. A leaf configuration together with a monitor unit is called a *Segment* or an *Aperture*. Several segments are superimposed to generate a custom shape of the beam. Figure 3.2b shows the addition of the different segments and the contour plot of the resulting dose.

3.3 Optimization Approaches

A complete problem formulation for radiation treatment optimization involves hundreds of variables. Traditionally, the problem is reduced to a subset of variables in order to use a mathematical programming approach that is suitable for the selected subset. Optimality in radiation treatment is hard to quantify. Radiation oncologists use dose computation algorithms to calculate a cumulative *Dose Volume Histogram* (DVH) to decide on the quality of the suggested treatment plan. The DVH shows the curves for each type of tissue, PTV, OHT, OAR, that are the fraction of the volume of each tissue as a function of the relative dose. The dose computation algorithms make use of the density information available from *Computer Tomography* (CT) scans. Dose computation algorithms range from simple fitting methods that correct reference dose values to the tissue density, up to very sophisticated Monte Carlo simulations that produce accurate dose data. The dose computation divides the BEV up into a grid of rectangles. Each rectangle represents a small part of the beam. This partial beam is also called a *pencil beam* or *beamlet*. We denote a beamlet with p.

To compute the total dose matrix D, we divide the area of interest into cross sections or *slices*. These slices correspond usually to medical image slices from the CT scan. The cross sections are then divided into a grid of tissue cubes, also called *voxels*. For every pencil beam p with a monitor unit weight w_p, the dose computation algorithm calculates the dose value D_{ij}^p that is delivered to each voxel

$$D_{ij} = \sum_{p=1}^{n} w_p D_{ij}^p, \tag{3.1}$$

where n is the number of pencil beams and i and j indicate the position of the voxel in the dose grid.

A traditional optimization approach for IMRT is the so called *Fluence Map Optimization* or simply fluence optimization. In fluence optimization, the weights of the pencil beams are optimized. The fluence map is the BEV grid representing all the beamlets, where each rectangle has a color (value) assigned that represents the weight. Once the optimal weight for each beam is known, a leaf sequence algorithm is run to translate the fluence map into deliverable sequence configurations. To reduce the complexity, a fixed set of equispaced beam angles is considered.

A fast way to solve for the fluence map is to use linear programming. Shepard [20] formulates a typical linear program as follows

$$\begin{aligned} \min_{w} \quad & \theta_{\mathcal{T}} \sum_{(k,l)\in\mathcal{T}} D_{kl} + \theta_{\mathcal{R}} \sum_{(k,l)\in\mathcal{R}} D_{kl} + \theta_{\mathcal{H}} \sum_{(k,l)\in\mathcal{H}} D_{kl} && \\ \text{s.t.} \quad & D_{ij} = \textstyle\sum_{p=1}^{n} w_p D_{ij}^p && \forall (i,j), \\ & \gamma \le D_{kl} && \forall (k,l) \in \mathcal{T}, \\ & w_p \ge 0, && \end{aligned} \tag{3.2}$$

where $\mathcal{T}$ is the PTV, $\mathcal{R}$ is the OAR, and $\mathcal{H}$ is the OHT. The number θ represents the weighted factor for the corresponding tissue type and γ represents a lower bound on the tumor voxel. Problem 3.2 is especially well suited for configurations with small number of beam angles. The reason is, that there is no upper bound on the tumor which translates to a high flexibility for the dose distribution. However, this leads also to areas of normal tissues with high dosage (hot spots). A solution that enforces more dose uniformity is the following model, also proposed by Shepard [20]

$$\begin{aligned} \min_{w} \quad & \theta_{\mathcal{R}} \sum_{(k,l)\in\mathcal{R}} D_{kl} + \theta_{\mathcal{H}} \sum_{(k,l)\in\mathcal{H}} D_{kl} && \\ \text{s.t.} \quad & D_{ij} = \textstyle\sum_{p=1}^{n} w_p D_{ij}^p && \forall (i,j), \\ & \gamma_L \le D_{kl} \le \gamma_U && \forall (k,l) \in \mathcal{T}, \\ & w_m \le \tfrac{\alpha}{n} \textstyle\sum_{p=1}^{n} w_p, && m = 1, 2, \ldots, n, \\ & w_p \ge 0. && \end{aligned} \tag{3.3}$$

Here, in addition to the lower bound γ_L, there is also an upper bound γ_U on the tumor voxels. This guarantees that the optimizer will not choose a beam configuration that results in a very high dose concentration at some tumor location and consequently, at the surrounding healthy tissue. There is also a bound on the pencil beam weight. Namely, a single beamlet weight cannot exceed α times the mean beam weight. This results in a more evenly distributed dose.

Shepard mentions that the disadvantage of the linear models is that only linear constraints can be imposed to the model. However, especially regarding the DVH, non-linear constraints are often desirable. He therefore proposes a non-linear weighted least squares model

$$\min_{w \ge 0} \theta_{\mathcal{T}} \sum_{(i,j)\in\mathcal{T}} (D_{ij}(w) - \delta_{ij})^2 + \theta_{\mathcal{R}} \sum_{(i,j)\in\mathcal{R}} (D_{ij}(w) - \delta_{ij})^2 + \theta_{\mathcal{H}} \sum_{(i,j)\in\mathcal{H}} (D_{ij}(w) - \delta_{ij})^2. \tag{3.4}$$

In the weighted least squares model, the tissue weights θ and the pencil beam weights w remain the same as in the linear model. However, in this model, the square of the difference of the dose D_{ij}, calculated from (3.1), and the prescribed dose δ is minimized. Typically, δ is taken to be zero for the OAR and the OHT. Notice that this problem formulation is convex and bound-constrained.

The weighted least square model can be used in traditional fluence map optimization. It is also used in more recent optimization approaches like *Direct Aperture Optimization* (DAO) [19]. In DAO, one optimizes directly the segment configurations rather than an intensity map that is later translated into a segment configuration. The advantage over fluence map optimization is that fewer segments are required to deliver the prescribed dose. In a typical fluence map optimization the number of segments required for an optimized intensity map with N different intensities is approximately $2N$ to $3N$. This creates large number of segments resulting in a large monitor unit to dose ratio. The large number of segments also creates uncertainities in leakage, scatter radiation, and other negative effects [19]. In DAO, however, one can specify the number of segments as a constraint. DAO does not directly optimize the pencil beam weights but rather the leaf configuration. Hence, in DAO, (3.4) becomes simply

$$\min_{w \geq 0} \theta_{\mathcal{T}} \sum_{(i,j) \in \mathcal{T}} (D_{ij} - \delta_{ij})^2 + \theta_{\mathcal{R}} \sum_{(i,j) \in \mathcal{R}} (D_{ij} - \delta_{ij})^2 + \theta_{\mathcal{H}} \sum_{(i,j) \in \mathcal{H}} (D_{ij} - \delta_{ij})^2 \tag{3.5}$$

where D_{ij} does not depend on the pencil beam weight w but rather on the entire segment configuration (leaf configuration and segment weight). However, pencil beams are still used in DAO to accelerate the optimization process. Instead of recomputing the whole dose after each leaf movement, the optimizer verifies which pencil beam is affected by the movement and adds or substracts the pencil beam dose to the total dose.

Due to the high complexity and the non-convexity of the DAO problem, simulated annealing is used to optimize the segment configurations. In DAO, as in fluence map optimization, a predefined set of beam angles is selected. The final number of beams is formulated as a constraint and the optimizer solves for the optimal beam angles and the corresponding segments.

3.4 Problem formulation

One objective of this thesis is to investigate the behaviour of the ECAM algorithm on the optimization of the beam angles. Rather than computing a subset of optimal angles from a preselected set of beams, the ECAM treats the angles as a continuous variables. In a typical IMRT treatment, the angles are either all coplanar, i.e. the linear accelerator turns around the patient who lies on a fixed table, or the angles are non-coplanar, in which case the table can be turned in order to introduce a spherical coordinate system for the beam angles. The advantage of coplanar beams is that the treatment time is shorter. Sometimes, however, a non-coplanar beam configuration is needed to reach complicated forms of tumors. In the coplanar case, one variable per beam is required where as the non-coplanar case requires two

variables per beam. In the scope of this paper, we treat only the case of coplanar beams. We then look at the behaviour of the ECAM in the coplanar case and discuss potential solutions for the non-coplanar case.

3.4.1 Bilevel Approach

Due to the high complexity of the full problem, a separation of variables is necessary. The ECAM is able to handle up to 10 variables fairly well. This suggests that the ECAM should be used to optimize the beam angles since in a typical treatment, usually 3 to 7 beams are used. A common MLC can have between 40 to 100 leafs. The segment optimization requires therefore several hundred variables, depending on the number of segments. This is impractical for the ECAM. Hence, we treat the problem as a bilevel optimization problem. The upper level optimization involves the beam angles and is solved by the ECAM. At each iteration of the ECAM, the lower level optimization is invoked. The lower level optimization solves for the segment configuration. The general formulation of the upper level problem is therefore

$$\min_{x \in \mathbb{R}^n} f(x) \tag{3.6}$$

where n is the number of beams and $x_i, i = 1, \ldots, n$ are the beam angles. The function f represents the lower level problem. Since this paper focuses more on the behaviour of the ECAM towards the upper level problem, we use a simplified approach to solve the lower level problem. In fact, we introduce a very simple mixed approach of DAO and fluence optimization. We divide the collimator up into s rectangular openings, similar to the fluence map approach. Contrary to the fluence map approach, we use rather large openings in order to keep s small. We treat those openings as separate segments. Hence, one can now look at the optimization problem as a sort of DAO approach, since we use fixed segments that are directly deliverable. To provide a fast solution for the lower level, we then use linear programming to determine the weight of each segment. In our problem formulation, we use a slight modification of (3.2) with the beam weight constraint from (3.3)

$$\begin{aligned} f(x) = \min_{w} \quad & \theta_{\mathcal{T}} \sum_{(k,l)\in\mathcal{T}} D_{kl}(w) + \theta_{\mathcal{R}} \sum_{(k,l)\in\mathcal{R}} D_{kl}(w) + \theta_{\mathcal{H}} \sum_{(k,l)\in\mathcal{H}} D_{kl}(w) && (3.7)\\ \text{s.t.} \quad & D_{ij} = \textstyle\sum_{s=1}^{n} w_s D_{ij}^s && \forall (i,j),\\ & \gamma \leq D_{kl} && \forall (k,l) \in \mathcal{T},\\ & w_m \leq \tfrac{\alpha}{n} \textstyle\sum_{p=1}^{n} w_s, && m = 1, 2, \ldots, n,\\ & w_s \geq 0. \end{aligned}$$

Notice that in the above formulation, we replaced w_p by w_s. This means that the weight w is not associated to a pencil beam anymore, but rather to a segment that belongs to a beam angle x from the upper level problem. We also remove the upper bound on the tumor. This allows us to test the ECAM with small numbers of beams without rendering the problem infeasible. To achieve some form of uniform dose distribution, we use constraints on the beam intensities similar to (3.3).

There are, of course, more sophisticated models to solve the above optimization problems. Partial volume constraints and biological models of cell response are just two examples of many that can be incorporated into a more advanced problem formulation. Those models, however, are outside the scope of this thesis.

Chapter 4

Numerical Results

4.1 Generic Test Problems

In the numerical experiments, we first tested ECAM on a family of classic global optimization functions. Most functions were taken from [6] to test the behaviour of the algorithm. The multivariate functions range from two to four dimensions.

4.1.1 Problem Descriptions

Problem 1 (One)

$$f(x) = 1$$
$$L = 0.5, 0 \leq x_i \leq 2, i = 1, \ldots, n.$$

Problem 2 (Convex)

$$f(x) = \sum_{i=1}^{n} x_i^2$$
$$L = 5.7, -2 \leq x_i \leq 2, i = 1, \ldots, n.$$

Problem 3 (Sum of sine)

$$f(x) = \sum_{i=0}^{n} \sin(x_i)$$
$$L = 4.5, 0 \leq x_i \leq 4, i = 1, \ldots, n.$$

Problem 4 (Six hump camel back)

$$f(x) = \left(4 - 2.1x_1^2 + \frac{x_1^2}{3}\right) x_1^2 + x_1 x_2 + 4\left(x_2^2 - 1\right) x_2^2$$
$$L = 100, -2 \leq x_i \leq 2, i = 1, 2.$$

Problem 5 (Product of sine)

$$f(x) = \sin(x_1)\sin(x_1 x_2)\ldots\sin(x_1 \cdots x_n)$$
$$L = 100, 0 \leq x_i \leq 4, i = 1, \ldots, n.$$

Problem 6 (Griewanks)

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
$$L = 10, -50 \leq x_i \leq 50, i = 1, \ldots, n.$$

4.1.2 Results

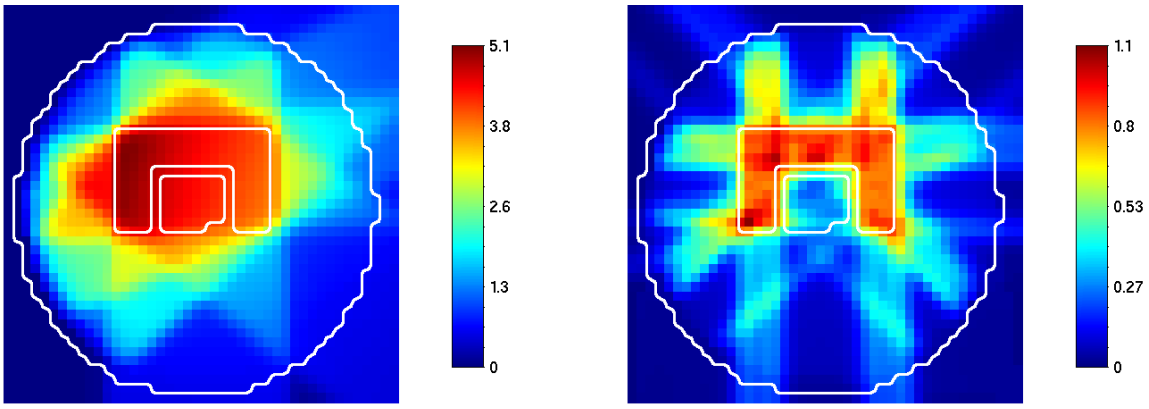
The generic test function experiments were done with Linux on a Macbook Pro laptop computer with an Intel®Core™2 Duo processor with 2.4 GHz and 4 GB of memory.

Problem	n	iterations	CPU (sec)	upper bound f_{best}	lower bound h	minimum
1	2	250	0.01	1.0	0.95	1
2	2	4000	0.09	0.0	-0.004999	0
3	2	250	0.0	-1.9996	-2.0492	-2
4	2	10000	0.12	-1.0316	-1.08015	-1.03162
5	2	50000	0.61	-0.9999	-1.2317	-1.0
6	2	100000	1.37	0.000002	-1.199989	0.0
1	3	10000	0.82	1.0	0.9596	1
2	3	4000	0.28	0.004325	-0.1256	0.0
3	3	50000	16.65	-2.9990	-8.6650	-3
5	3	50000	8.64	-0.9893	-7.5563	-1.0
6	3	50000	11.92	0.0023	-31.5504	0.0
5	4	10000	7.69	-0.8597	-33.1997	-1.0
6	4	60000	142.45	0.0334	-13.9190	0.0

Table 4.1: Generic Test Functions.

The results show a fast convergence towards the minimum. The algorithm was not run until convergence because a large number of iterations is required to obtain convergence to a tolerance similar to those used in local search algorithms. In fact the available memory on the test machine would not suffice to accommodate the number of support vectors. The fixed number of iterations is therefore taken as stopping condition. The real minimum in the right most column allows to assess the quality of the solution. The lower bound is the minimum of the saw tooth underestimate at termination.

One can observe that there are differences of the previous results with the results presented in [6]. One reason of these differences is the choice of the starting points. In our experiments, the starting points are chosen manually where as in [6] the choice of the starting points is performed with calculations and function sampling. Also, if a minimizer is projected onto the boundary ∂D, we evaluate it immediately in our implementation whereas in [6], the model value of the projected minimizer is calculated and the minimizer is moved



(a) 5 beams with no optimization.

(b) 5 beams after bilevel optimization.

Figure 4.1: The IMRT problem using a cylindrical water phantom with a U-shaped target surrounding the organ at risk.

back into V^k without direct evaluation. Beliakov incorporated his implementation of the ECAM in the GANSO package, `http://www.ganso.com.au`. The GANSO package also uses transformations of the original search domain.

4.2 IMRT Problem

4.2.1 Problem Descriptions

In the numerical experiments, we compared the ECAM with an implementation of Simulated Annealing (SA). For the Simulated Annealing algorithm, Everett (Skip) Carter's public available package was used [8]. The cooling schedule used in the package is the following.

$$T_k = \frac{T_0}{1.0 + k\alpha}$$

where T_k is the temperature at the current annealing iteration k, T_0 is the initial temperature depending on the problem configuration, and α is a scaling factor. The implementation of the Simulated Annealing method does not consider domain constraints. We therefore impose the box constraints at each function evaluation by penalizing the function with infinity if the evaluation location is outside the domain.

Instead of real patient data, a cylindrical water phantom is used for the IMRT problem. The PTV consists of a U-shaped structure around a small cube that represents the OAR.

This is a standard IMRT problem that is also used in [20]. The challenge for the optimizer is to cross the straight line beams in order to fully irradiate the U-shaped PTV while avoiding the OAR cube in the middle. A visualization of the problem is shown in Figure 4.1.

Problem 1 uses formula 3.7 with 3 coplanar beams. One beam is fixed at 180 °C and the other two move freely between [0°,119°] and [240°,359°].

Problem 2 uses formula 3.7 with 5 coplanar beams. However, since the 5 beams provide a more uniform dose distribution, the tumor upper bound from 3.3 is added as a constraint. Note that in a typical treatment with coplanar beams, around 5 to 7 beams are used. 7 beams provide usually more accurate results but incur a longer treatment time.

4.2.2 Results

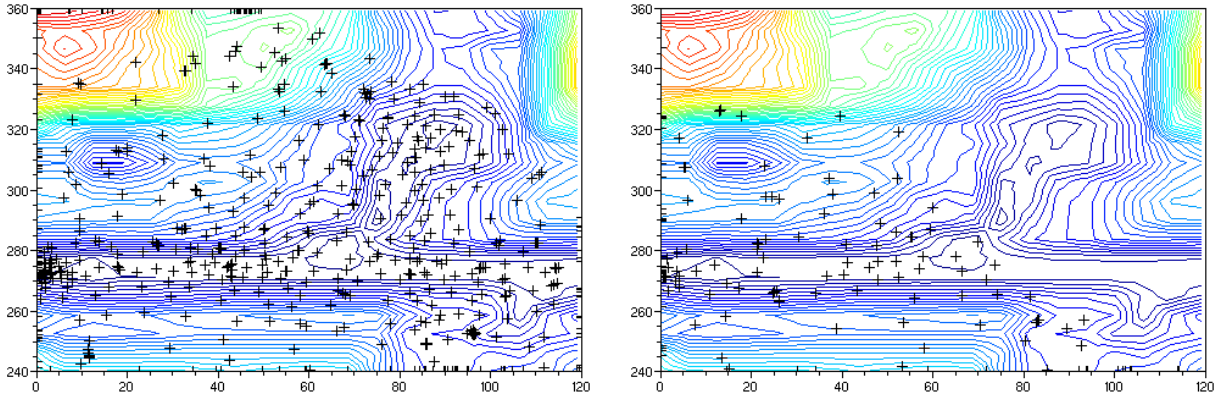
The IMRT experiments were done on a Linux desktop computer with an Intel®Core™2 Quad processor with 2.4 GHz and 4 GB of memory. For the dose computation, the radiotherapy treatment planning software PLanUNC 6.6.10 from the University of North Carolina was used. The dose computation for each segment was parallelized using the *Message Passing Interface* (MPI).

Problem	Iterations	CPU (sec)	upper bound f_{best}	lower bound h	Figure
1 (ECAM $L = 200$)	826	986	7978.99	7978.49	4.2a
1 (ECAM $L = 150$)	272	329	7973.91	7973.41	4.2b
1 (SA $T_0 = 1000$)	1000	548	8000.12		4.2c
1 (SA $T_0 = 2000$)	1000	408	8007.85		4.2d

Table 4.2: IMRT Problem 1.

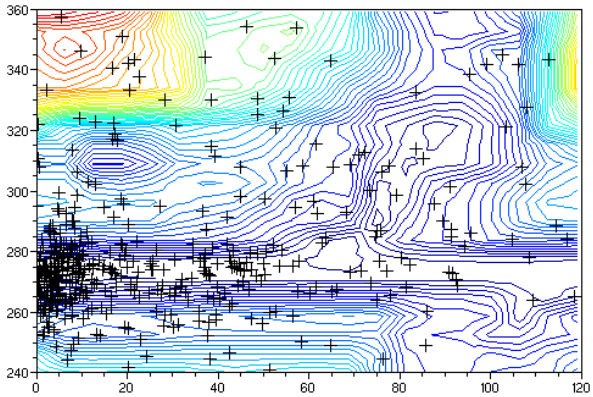
The behaviour of the algorithms on Problem 1 is shown in Figure 4.2. Figure 4.2a and 4.2b show the ECAM with different Lipschitz constants. In Figure 4.2a, one can see how the ECAM clusters around the local minima and eventually focuses the function evaluations to the global minimum. Figure 4.2c and 4.2d show the SA algorithm with different starting temperatures. In Figure 4.2c, the algorithm chooses a more "greedy" strategy and ends up in the global minimum. In Figure 4.2d, the algorithm accepts more risky steps and ends up in a local minimum.

Problem 1 shows clearly the advantages and disadvantages of each algorithm. The SA algorithm is faster at large numbers of iterations than the ECAM. This is expected since SA does not need to maintain a large tree structure. Also, the behaviour of SA changes with the starting temperature. The choice of the right starting temperature is probably as tricky as the choice of the correct Lipschitz constant for the ECAM. In both cases, presampling of the function might be of some help. The advantage of the ECAM is the guaranteed convergence

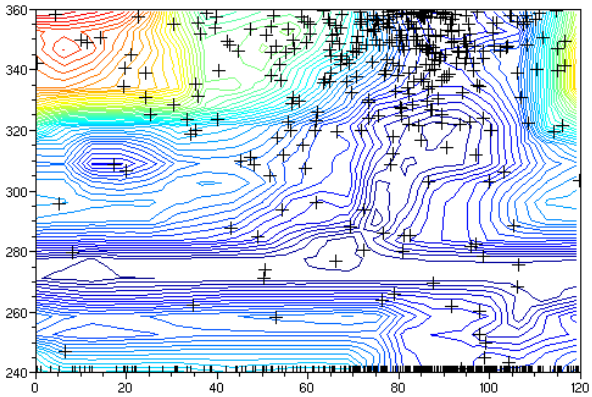


(a) ECAM, 826 iterations ($L = 200$).

(b) ECAM, 272 iterations ($L = 150$)



(c) SA, 1000 iterations ($T_0 = 1000$).



(d) SA, 1000 iterations ($T_0 = 2000$)

Figure 4.2: ECAM and Simulated Annealing (SA) for a problem with two variables. The problem involves one fixed beam at 180° and two beams with angles in [0°,119°] and [240°,359°].

to the global minimum. Furthermore, if the Lipschitz constant is low, the algorithm does find the minimum faster than SA.

In the experiments with Problem 2, the ECAM implementation from the GANSO package was used to solve the problem. The results are shown in Table 4.2.2. In Table 4.2.2, the experiments are grouped by the number of iterations.

In this experiment we can see that SA performs better than ECAM. However, one must take into consideration the structure of the problem. In Problem 2, the five beams are all coplanar and restricted to a relatively small range of [0°,72°]. This results in very few local minima and this configuration is therefore not an optimal problem to show the efficiency of the ECAM over SA. However, if one extends Problem 2 with the use of non-coplanar beams,

Problem	Iterations	CPU (sec)	upper bound f_{best}	lower bound h
2 (ECAM $L = 150$)	1000	1930	7930.89	5060.87
2 (SA $T_0 = 400$)	1000	719	7821.31	
2 (ECAM $L = 150$)	2000	3844	7939.31	5622.69
2 (SA $T_0 = 400$)	2000	1788	7801.77	

Table 4.3: IMRT Problem 2.

the search domain will contain significantly more local minima and will extend to $\mathbb{R}^{10}$. In fact, Pugachev et al. show that non-coplanar beams result in significant improvements for nasopharyngeal cases and paraspinal cases [16]. Also, Wang et al. show in [22] that the use of 5 non-coplanar beams provides better results than the use of 9 coplanar beams in the case of paranasal sinus carcinoma.

The focus of this thesis was to implement and apply the ECAM to the IMRT problem. The restriction to coplanar problems was intentional due to the scope and the time constraint of the thesis. For future work, we suggest to investigate the ECAM on several problems with non-coplanar beams. The data from Problem 1 suggest that the ECAM will provide better results in cases with multiple local minimas.

Chapter 5

Conclusion

In the first part of this paper we discussed Lipschitz continuity and the difficulties that arise if one extends the one dimensional Pijavskii-Shubert to multiple dimensions. We then introduced the notions of convexity, abstract convexity and the General Cutting Plane method. We then explained the Cutting Angle Method and the Extended Cutting Angle Method.

In the second part of the thesis, we introduced the field of radiation therapy planning. We discussed several optimization approaches and we proposed a new bilevel approach to solve the beam angle and the beam segments simultaneously. In the last part of the thesis, we first tested the implementation of the ECAM on several test functions. We then tested the bilevel approach with the Extended Cutting Angle Method and Simulated Annealing.

The Extended Cutting Angle method provides accurate results of the global minimum for each function tested. Problem 1 showed the advantages of ECAM over Simulated Annealing. In cases where Simulated Annealing does not find the global minimum, ECAM does find it or gives us at least a lower bound on the minimum. This is of significant importance for time consuming treatments where good results are crucial to justify the treatment time. This is especially the case for non-coplanar beams. In this thesis we did not have enough time to investigate the behaviour of ECAM vs. SA on non-coplanar beams. However, the early data from Problem 1 indicates that ECAM behaves better when multiple local minima are present. Problem 2 used a large number of beams with restricted range of movement. This leads to fewer local minima and is therefore not the best test case to show the performance of ECAM. Future work should therefore focus on the optimization of non-coplanar beams. Due to the proposed bilevel approach, several lower level optimization approaches like DAO, LP, or projection methods can be tested interchangeably without affecting the upper level optimization.

Bibliography

[1] *Canadian cancer statistics 2007*, tech. rep., Canadian Cancer Society/National Cancer Institute of Canada, Toronto, Canada, April 2007.

[2] M. Andramonov and A. M. Rubinov, *Cutting angle methods in global optimization*, Applied Mathematics Letters, 12 (1999), pp. 95–100.

[3] J. L. Bedford and S. Webb, *Direct-aperture optimization applied to selection of beam orientation in intensity-modulated radiation therapy*, Physics in Medicine and Biology, (2007).

[4] G. Beliakov, *Geometry and combinatorics of the cutting angle method*, Optimization, 52 (2003), pp. 379–394.

[5] ——, *The cutting angle method - a tool for constrained global optimization*, Optimization Methods and Software, 19 (2004), pp. 137–151.

[6] ——, *Extended cutting angle method of global optimization*, Pacific Journal of Optimization, 4 (2008), pp. 153–176.

[7] G. Beliakov and L. M. Batten, *Fast algorithm for the cutting angle method of global minimization*, Journal of Global Optimization, 24 (2002), pp. 149–161.

[8] E. S. Carter. http://www.taygeta.com/annealing/simanneal.html.

[9] D. J. Estep, *Practical Analysis in One Variable*, Springer, 2002, ch. 8.6, pp. 93–94.

[10] C. A. Floudas, *Deterministic Global Optimization: Theory, Methods and Applications*, vol. 37 of Nonconvex Optimization and its Applications, Springer, 1999.

[11] R. J. V. Iwaarden, *An Improved Unconstrained Global Optimization Algorithm*, PhD thesis, University of Colorado, Denver, 1996.

[12] J. E. Kelley, *The cutting-plane method for solving convex programs*, Journal of the Society for Industrial and Applied Mathematics, 8 (1960), pp. 703–712.

[13] E. K. Lee, T. Fox, and I. Crocker, *Simultaneous beam geometry and intensity map optimization in intensity-modulated radiation therapy*, International Journal of Radiation Oncology, Biology, Physics, 64 (2006), pp. 301–320.

[14] R. H. Mladineo, *An algorithm for finding the global maximum of a multimodal, multivariate function*, Mathematical Programming, 34 (1986), pp. 188–200.

[15] S. A. Pijavskii, *An algorithm for finding the absolute extremum of a function*, USSR Computational Mathematics and Mathematical Physics, (1972), pp. 57–67.

[16] A. Pugachev, J. G. Li, A. L. Boyer, and L. Xing, *Role of non-coplanar beams in imrt*, Engineering in Medicine and Biology Society, 2000. Proceedings of the 22nd Annual International Conference of the IEEE., 1 (2000), pp. 456–459.

[17] A. M. Rubinov, *Abstract Convexity and Global Optimization*, Nonconvex Optimization and its Applications, Kluwer Academic Publishers, P.O. Box 17, 3300 AA Dordrecht, The Netherlands, 2000.

[18] A. P. Ruszczynski, *Nonlinear Optimization*, Princeton University Press, 41 Wiliam Street, Princeton, New Jersey 08540, 2006.

[19] 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 (2002), pp. 1007–1018.

[20] D. M. Shepard, M. C. Ferris, G. H. Olivera, and T. R. Mackie, *Optimizing the delivery of radiation therapy to cancer patients*, SIAM Review, 41 (1999), pp. 721–744.

[21] B. O. Shubert, *A sequential method seeking the global maximum of a function*, SIAM Journal on Numerical Analysis, 9 (1972), pp. 379–388.

[22] X. Wang, X. Zhang, L. Dong, H. Liu, M. Gillin, A. Ahamad, K. Ang, and R. Mohan, *Effectiveness of noncoplanar imrt planning using a parallelized multiresolution beam angle optimization method for paranasal sinus carcinoma.*, International Journal of Radiation Oncology, Biology, Physics, 1 (2005), pp. 594–601.