Multiobjective optimization in radiosurgery

How to approximate and navigate on the Pareto surface

J E N N I S V E N S S O N

Master of Science Thesis
Stockholm, Sweden 2014

Multiobjective optimization in radiosurgery

How to approximate and navigate on the Pareto surface

J E N N I S V E N S S O N

Master's Thesis in Optimization and Systems Theory (30 ECTS credits)
Master Programme in Mathematics (120 credits)
Royal Institute of Technology year 2014
Supervisor at Elekta AB was Jens Sjölund
Supervisor at KTH was Krister Svanberg
Examiner was Krister Svanberg

TRITA-MAT-E 2014:25
ISRN-KTH/MAT/E--14/25--SE

Royal Institute of Technology
School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Abstract

Cancer is a common cause of death worldwide and radiotherapy is one of the treatments used. Since treatment planning is a time consuming matter for the radiation therapist, a way to decrease the time spent finding the plan would be an improvement. This can be achieved by pre-calculating a number of optimal plans and then choosing among these in real-time.

In this thesis a dual algorithm for approximation of the Pareto optimal plans suggested by Bokrantz and Forsgren, was adapted to the parameters of the Leksell Gamma Knife®. A Graphical User Interface was also created, based on the navigation tool described by Monz et al to enable choosing among the pre-calculated dose plans.

The computational time of the algorithm was investigated and the dimensionality of the solutions and Pareto optimal points were looked at to see if it might be possible to reduce the number of dimensions to speed up computations.

Although no certain conclusions can be drawn about dimensionality reduction, I found no reason to rule that possibility out. It was also confirmed that there is reason to keep the number of objectives low to get a better approximation.

Referat

Svenska

Cancer är en allt vanligare dödsorsak i världen och strålterapi är en vanlig behandlingsmetod. Att ta fram en strålbehandlingsplan är en tidskrävande process för den ansvarige sjukhusfysikern eller läkaren.

Ett sätt att korta ner denna tid är att förberäkna ett antal optimala behandlingsplaner och sedan välja mellan kombinationer av dessa i realtid. Planerna som tas fram med hjälp av en dual algoritm föreslagen av Bokrantz och Forsgren, är anpassade till Leksell Gamma Knife®. Ett grafiskt verktyg har skapats för att navigera mellan de förberäknade planerna, baserat på navigeringsverktyget beskrivet av Monz et al.

Beräkningstiden för att ta fram planerna har studerats, tillsammans med olika faktorer som påverkar den. I anknytning till detta gjordes en enklare dimensionsanalys av lösningarna och de Pareto-optimala punkterna för att se om det är möjligt att reducera antalet dimensioner för att snabba upp beräkningstiden.

Inga långtgående slutsatser kan dras angående detta, men möjligheten går inte att utesluta. Slutsatsen att försöka hålla antalet målfunktioner lågt för att få en så bra approximation som möjligt bekräftades.

Contents

List of Figures

Preface

This master's thesis has been done at Elekta, a company providing medical equipment for treatment of cancer and brain disorders. The project was supervised by the division of Optimization and Systems Theory at The Royal Institute of Technology (KTH) in Stockholm. I would like to thank my supervisors Jens Sjölund, at Elekta, and Krister Svanberg, at KTH, for their valuable time and guidance.

1 Introduction

Cancer is one of the most common causes of death worldwide. Treatment forms involve surgery, chemotherapy and radiotherapy, often combined. A major advantage of radiotherapy is that it is non-invasive, avoiding a lot of the risks following surgical procedures.

The goal of this thesis has been to look at an alternative way of producing treatment plans for radiosurgery of brain disorders. Today, the treatment plan is largely done manually, by a radiation therapist. The plan is found by adjusting parameters and letting the computer redetermine a solution for each new set of parameters until a satisfactory plan is found. To reduce planning time a number of treatment plans could be pre-computed and a visual tool be provided for finding a satisfactory solution among these.

Often there are several objectives to consider, apart from getting enough radiation to the tumour, sparing risk organs close to the tumour and keeping delivery time short. If the different objectives are not in conflict the problem can be solved quite easily. However, there rarely exists one solution which is optimal in all aspects, but rather several solutions, corresponding to different ways of prioritizing the conflicting goals. The difficulty lies in beforehand knowing how to weigh the importance of the different objectives to achieve a certain dose distribution. This is why a number of plans which are optimal for different priorities can be pre-computed. By making trade-offs between the different goals, the planner plays an important role in choosing a treatment plan among these different optimal plans.

Immediate visual feedback in terms of dose distributions illustrate how different priorities affect the treatment plan and helps the planner to choose according to his/her preferences. A part of the project has therefore been to create such a graphical tool.

This thesis starts with a description of radiotherapy and how Elekta's Leksell Gamma Knife® works, continuing with explaining multiobjective optimization and a description of the implemented algorithm used to approximate the optimal solutions. The process of navigating between the optimal treatment plans and how the visual tool works is described and further improvements are discussed. The algorithms are also analyzed with respect to computational times and the dimensionality of the bundle of computed plans and corresponding solutions are examined.

2 Background

2.1 Radiotherapy

Radiotherapy uses high energy radiation to break the DNA of cancer cells, which results in slower reproduction and eventually killing these cells.

Different forms of radiotherapy may be used; a distinction is made between external beam therapy, internal radiotherapy (brachytherapy) and systemic radioisotope therapy. In external beam therapy the source of radiation, emitting either photons or particles with high energy, is located outside the body directing beams at the treatment area. Brachytherapy uses small radioactive sources that are placed within or next to areas to treat and in systemic radioisotope therapy radioisotopes are injected into the bloodstream or ingested. External beam therapy is the method most commonly used.

In external beam therapy, healthy tissue gets a considerable part of the radiation, which is why doses have to be kept low and delivered at several treatment occasions, letting healthy tissue recover in between. This is called fractioning and works due to the fact that tumour cells are not as good at recovering from radiation damage as healthy cells.

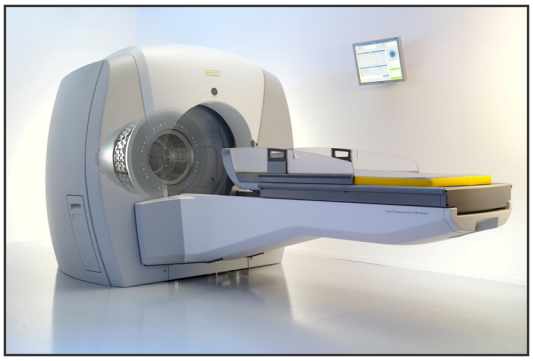
Radiosurgery, which this thesis concerns, is a form of external beam therapy that usually only requires one treatment occasion, hence the reference to surgery in the name.

2.2 Radiosurgery and the Leksell Gamma Knife®

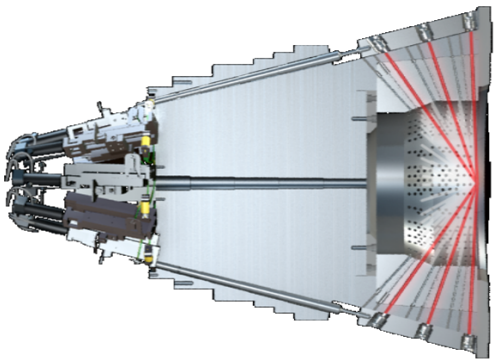
Stereotactic radiosurgery is a form of radiotherapy, where the high energy beams are focused to deliver a higher dose to the tumour area, letting less radiation damage healthy tissue around the tumour. This means it is often possible to deliver the treatment at only one occasion. Due to the high doses delivered, high accuracy of the patient's position is critical.

This thesis will consider properties associated with Elekta's Leksell Gamma Knife® (LGK) Perfexion. The Leksell Gamma Knife® (LGK) consists of a platform, which can move in three dimensions, on which the patient is fixated, and a stationary part containing the collimator body that focuses the beams. In the LGK the brain position is kept still by attaching a stereotactic frame to the patient's head using

small screws fastened to the skull. This frame is then attached to the LGK, giving the coordinate system used in the treatment planning.



(a) The Leksell Gamma Knife®



(b) The collimator body which focuses the beams.

Figure 2.1: The Leksell Gamma Knife® and a close up of the collimator body that surrounds the head during treatment.

2.3 Treatment plan

In order to make a treatment plan medical images of the area of the patient to be treated are necessary. MRI (Magnetic Resonance Imaging) images are most common because of their soft tissue contrast, but CT-scans (Computed Tomography), which give 3D X-ray information may also be used as a complement. PET (Positron Emission Tomography) is another technique that may be used to give further information. Using a number of provided 2D images, the targets are manually located and delineated along with the Organs At Risk (OARs).

Today treatment plans are found by so called inverse planning, where the desired dose distribution is known and the optimization aims to find the machine parameters to achieve this distribution. The planning time may be long due to an iterative process involving the planner, who has to adjust the objective function used and rerun the time-consuming optimization until a satisfactory plan is generated.

Since the preferences of the planner are not explicitly known beforehand, the goal in this thesis is instead to approximate the set of all optimal treatment plans so that the planner can navigate on this set in real-time, using a visualization tool, until a favourable plan is found.

2.4 Variables in the Leksell Gamma Knife®

The variables in this thesis relate to the machine parameters of the LGK, which are briefly described in this section.

The collimator body delivers a set of cone shaped beams through eight different sectors, resulting in a focused spot where they intersect, creating a so called shot. Each sector can deliver beams of three different sizes or it may be switched off, resulting in shots of different sizes and shapes. The patient is moved in between shots to change the focus point, referred to hereinafter as different shot positions.

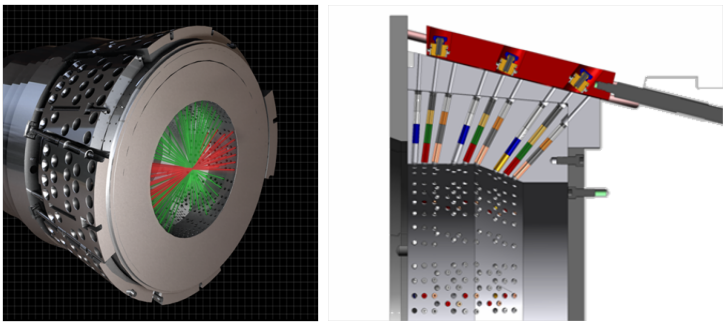


Figure 2.2: The collimator body contains 8 individually controlled sectors, with 3 possible beam sizes each.

The number of shots N and their positions have been determined beforehand using a packing algorithm [1] that fills up the target volumes with shots. For each shot there are eight possible sectors to use, each with three different collimator sizes. This results in $N \cdot 8 \cdot 3$ degrees of freedom of our optimization variable, each component corresponding to the time which the beam is on for that collimator, sector and shot.

2.5 Mathematical background

The algorithms used in this thesis require mathematical background knowledge on some concepts that will be explained in this section. The mathematical definitions and theorems on optimization are taken from [7] if not stated otherwise. Minor modifications have been made to suit this project.

2.5.1 Multiobjective optimization

Multiobjective optimization is a type of optimization dealing with several objective functions, $f_1, f_2, \ldots, f_n$, which all should be minimized. All Multiobjective Optimization Problems (MOPs), can be written on the general form

$$(\text{MOP}) \begin{cases} \underset{\boldsymbol{x}}{\text{minimize}} & \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\} \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases}$$

where $\boldsymbol{x}$ is a solution vector and S is the set of all feasible solution vectors, which can be expressed as constraint functions and bounds on the variables. Only considering

minimization is not a limitation since one can easily translate a maximization problem into a problem of minimization ($\max f(\boldsymbol{x}) \Longleftrightarrow \min -f(\boldsymbol{x})$).

The goal in solving an MOP is to minimize all of the objective functions simultaneously, but since it is not generally the case that all objectives achieve an optimal value simultaneously a new definition of optimality is needed.

Let $Z = \boldsymbol{f}(S)$ be the feasible objective region and $\boldsymbol{z} = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x})\} \in Z$, such that $\boldsymbol{x} \in S$, be a feasible point in objective space.

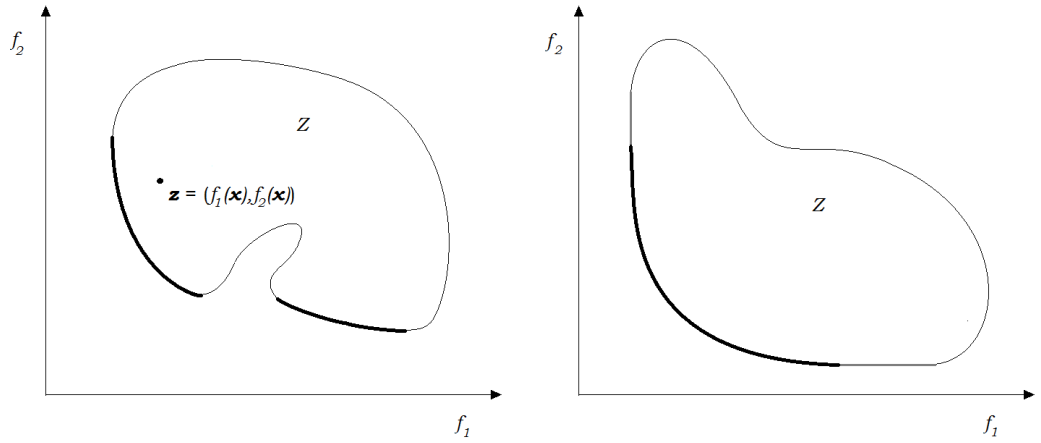
Definition 2.1. A decision vector $\boldsymbol{x}^* \in S$ is *weakly Pareto optimal* if there does not exist another decision vector $\boldsymbol{x} \in S$ such that $f_i(\boldsymbol{x}) < f_i(\boldsymbol{x}^*)$ for all $i = 1, \ldots, n$.

An objective vector $\boldsymbol{z}^* \in Z$ is weakly Pareto optimal if there does not exist another objective vector $\boldsymbol{z} \in Z$ such that $z_i < z_i^*$ for all $i = 1, \ldots, n$; or equivalently, $\boldsymbol{z}^*$ is weakly Pareto optimal if the decision vector corresponding to it is weakly Pareto optimal.

Definition 2.2. A decision vector $\boldsymbol{x}^* \in S$ is *Pareto optimal* if there does not exist another decision vector $\boldsymbol{x} \in S$ such that $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{x}^*)$ for all $i = 1, \ldots, n$ and $f_j(\boldsymbol{x}) < f_j(\boldsymbol{x}^*)$ for at least one index j.

An objective vector $\boldsymbol{z}^* \in Z$ is Pareto optimal if there does not exist another objective vector $\boldsymbol{z} \in Z$ such that $z_i \leq z_i^*$ for all $i = 1, \ldots, n$ and $z_j < z_j^*$ for at least one index j; or equivalently, $\boldsymbol{z}^*$ is Pareto optimal if the decision vector corresponding to it is Pareto optimal.

In other words, for a Pareto optimal solution a function value can only be decreased if another one is increased.



(a) Non-connected Pareto surface (b) Connected convex Pareto surface

Figure 2.3: Thick line shows the Pareto frontier and the grey line segments in (b) show weakly Pareto optimal points. The feasible point $\boldsymbol{z}$ in the left figure is not Pareto optimal since $f_1(\boldsymbol{x})$ can be lowered, keeping $f_2(\boldsymbol{x})$ constant and vice versa.

There usually exist an infinite number of Pareto optimal points in the objective space, forming a Pareto surface or sometimes called Pareto frontier, which may be both non-convex and non-connected, see Figure 2.3a.

However for convex MOPs the Pareto optimal points $\boldsymbol{z}^*$ are connected and form a convex set. Convexity is a crucial property in this thesis which both the approximation and navigation rest on and the concept of convexity will therefore be explained next.

Convexity

In this project only convex MOPs are considered, since the Pareto optimal set Z^* can then be approximated by a sandwich algorithm, that computes upper and lower approximations of the surface, thereby enclosing it, see Figure 2.4.

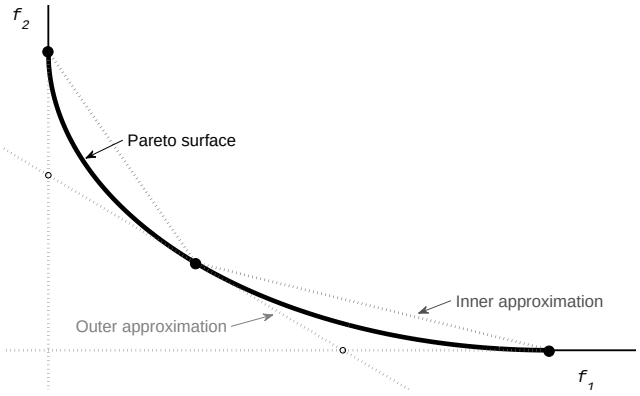


Figure 2.4: Inner and outer approximations of the sandwich algorithm for three Pareto optimal points.

Defining a convex MOP requires the definitions of convex functions and convex sets.

Definition 2.3. A function $f : \mathbb{R}^n \to \mathbb{R}$ is *convex* if for all $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \in \mathbb{R}^n$ it holds that $f(\beta\boldsymbol{x}^{(1)} + (1-\beta)\boldsymbol{x}^{(2)}) \leq \beta f(\boldsymbol{x}^{(1)}) + (1-\beta)f(\boldsymbol{x}^{(2)})$ for all $0 \leq \beta \leq 1$.

A set $S \subset \mathbb{R}^n$ is *convex* if $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \in S$ implies that $\beta\boldsymbol{x}^{(1)} + (1-\beta)\boldsymbol{x}^{(2)} \in S$ for all $0 \leq \beta \leq 1$.

In other words a function is convex if the line segment between two arbitrary points on the curve lies above the curve and a set is convex if all points on a line segment between two arbitrary points in the set lie in the set.

A convex (MOP) can now be defined.

Definition 2.4. A multiobjective optimization problem is *convex* if all the objective functions and the feasible region are convex.

The first step to enable us to use a sandwich algorithm is thus to make sure the

problem is convex. The method of finding Pareto optimal points is described in the following section.

Scalarization

The way to handle MOP is by scalarization, transforming the vector of objective functions to a scalar valued function. This makes the problem solvable as an ordinary single objective optimization problem. Scalarization can be achieved by assigning positive normalized weights $w_1, w_2, \ldots, w_n$ to each objective function and formulating the dot product, yielding:

$$\text{(WSP)} \begin{cases} \underset{\boldsymbol{x}}{\text{minimize}} & \sum_{i=1}^{n} w_i f_i(\boldsymbol{x}) \\ \text{subject to} & \boldsymbol{x} \in S, \end{cases}$$

where $w_i \geq 0$ for all $i = 1, \ldots, k$ and $\sum_{i=1}^{n} w_i = 1$.

The Weighted Sum Problem (WSP) can be solved using regular optimization methods and as it turns out the solution $\boldsymbol{x}$ is Pareto optimal if certain requirements are met.

Theorem 2.5. *The solution of (WSP) is weakly Pareto optimal.*

Theorem 2.6. *The solution of (WSP) is Pareto optimal if the weighting coefficients are positive, that is $w_i > 0$ for all $i = 1, \ldots, n$.*

Theorem 2.7. *A unique solution of the weighting problem is Pareto optimal.*

Theorem 2.8. *Let the MOP be convex. If $\boldsymbol{x}^*$ is Pareto optimal, then there exists a weighting vector $\boldsymbol{w}$ ($w_i \geq 0$, $i = 1, \ldots, n$, $\sum_{i=1}^{k} w_i = 1$) such that $\boldsymbol{x}^*$ is a solution of (WSP).*

Theorem 2.8 shows that every Pareto optimal point of a convex MOP can be found by some nonnegative weights $\boldsymbol{w}$ and Figure 2.5 shows why this may not be possible in the non-convex case. The weights constitute supporting hyperplanes to the Pareto surface. By making the functions f_i strictly convex and thus ensuring unique solutions, all non-negative weights could guarantee Pareto optimal solutions. Once positive weights have been chosen the corresponding Pareto optimal solution $\boldsymbol{x}$ can be determined using regular optimization techniques.

Ideally a huge number of Pareto optimal solutions and corresponding points in the objective space would be calculated, but since this is prohibitively time consuming it has to be approximated by a limited amount of points. The difficulty lies in choosing the weight vectors to get a set of well distributed Pareto optimal points $\boldsymbol{z}^*$.

When solving the (WSP) the objective functions are normalized so that no objective will dominate the others. Two concepts are important in this context, namely the ideal vector and the nadir vector, which give the lower and upper bounds on the functions values $\boldsymbol{z}^*$, see Figure 2.6. The ideal vector is defined as follows:

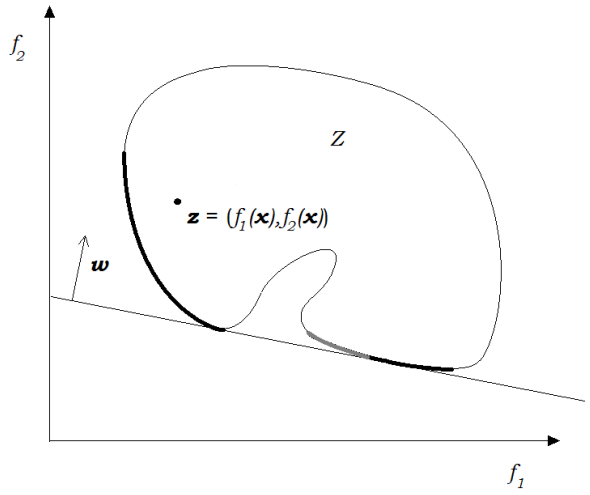


Figure 2.5: The grey curve segment represents Pareto optimal points which cannot be found by any weight.

Definition 2.9. The components z_i^* of the *ideal objective vector* $\boldsymbol{z}^* \in \mathbb{R}^n$ are obtained by minimizing each of the objective functions individually subject to the constraints, that is, by solving

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}} && f_i(\boldsymbol{x}) \\ &\text{subject to} && \boldsymbol{x} \in S, \end{aligned}$$

for all $i = 1, \ldots, k$.

If feasible the ideal point would be a unique Pareto optimal point $\boldsymbol{z}^*$.

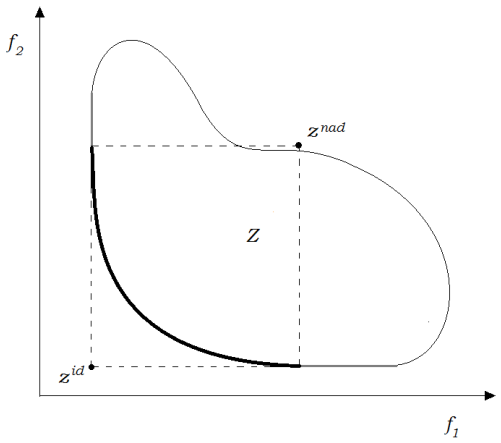


Figure 2.6: Ideal and nadir points of the Pareto surface

Similarly the nadir vector contains the worst Pareto optimal function values. However this problem is a convex maximization problem and cannot be determined, but has to be approximated. A common way to do this is by using a pay-off table. This is done by taking the solutions to the problems solved when searching for the ideal vector and evaluating the other functions for those solutions and then comparing them component-wise. The maximum value for each component $f_i(\boldsymbol{x})$ constitutes an approximation of element i of the nadir vector, see Table 2.1.

	$\boldsymbol{x}^*$	$f_1(\boldsymbol{x})$	$f_2(\boldsymbol{x})$	$\cdots$	$f_n(\boldsymbol{x})$
minimize $f_1(\boldsymbol{x})$	$\boldsymbol{x}^{*(1)}$	$f_1(\boldsymbol{x}^{*(1)})$	$f_2(\boldsymbol{x}^{*(1)})$	$\ldots$	$f_n(\boldsymbol{x}^{*(1)})$
minimize $f_2(\boldsymbol{x})$	$\boldsymbol{x}^{*(2)}$	$f_1(\boldsymbol{x}^{*(2)})$	$f_2(\boldsymbol{x}^{*(2)})$	$\ldots$	$f_n(\boldsymbol{x}^{*(2)})$
$\vdots$					
minimize $f_n(\boldsymbol{x})$	$\boldsymbol{x}^{*(n)}$	$f_1(\boldsymbol{x}^{*(n)})$	$f_2(\boldsymbol{x}^{*(n)})$	$\ldots$	$f_n(\boldsymbol{x}^{*(n)})$

Table 2.1: Pay-off table: z_i^{nad} is approximated by the maximum value in column i. The diagonal consists of z_i^{id} components.

The normalization of the objective functions, which is done prior to searching for Pareto optimal points, is performed with respect to the values of the ideal and nadir vectors:

$$\tilde{f}_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - z_i^{id}}{z_i^{nad} - z_i^{id}}, \tag{2.1}$$

which ideally would mean that the normalized function values will be in the range [0,1]. However, since the nadir point is only approximated it might be either too high or too low [5], which is important to keep in mind.

2.5.2 Dual Polytopes

The vertices of the outer approximation play an important role when finding an error bound in the dual algorithm that is used for approximating the Pareto surface, see Figure 2.4. They consist of the intersections of the supporting hyperplanes of the Pareto surface, that are defined by the weights and corresponding Pareto optimal points. To find these outer vertices an algorithm based on the concept of dual polytopes is used.

The convex hull of a set of points describes a convex polytope that can be represented by the hyperplanes enclosing it. A hyperplane can be written as $\{\boldsymbol{z} : \boldsymbol{a}^T\boldsymbol{z} = b\}$, where $\boldsymbol{a}$ is the nonzero normal of the hyperplane. If the normals of the polytope hyperplanes are oriented outwards, all points inside the polytope can be described as the set $\{\boldsymbol{z} : \boldsymbol{A}^T\boldsymbol{z} \leq b\}$, where $\boldsymbol{A}$ is a matrix containing the hyperplane normals.

In an n-dimensional polytope a 0-dimensional element is called a vertex, a 1-dimensional element an edge, an element of $n-2$ dimensions is called a ridge and elements of $n-1$ dimensions are called a facets.

By inverting the facets of a polytope in the unit sphere a set of points are created that constitute the vertices of another polytope, called the dual polytope. A dual vertex point $\mathbf{d}$ is given by $\mathbf{d} = \frac{\boldsymbol{a}}{b}$, a reciprocation in the unit sphere, where $\boldsymbol{a}$ is normalized. For clarity the original polytope is called the primal polytope. A 2D example can be seen in Figure 2.7.

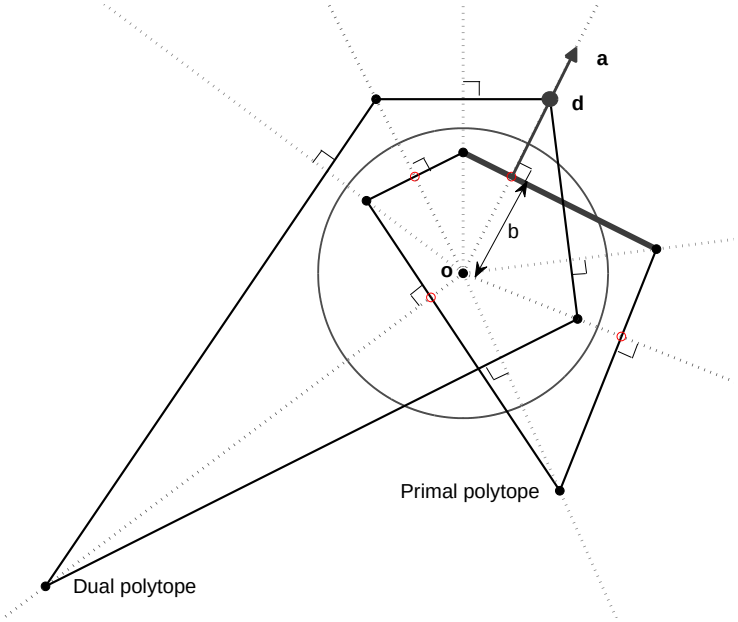


Figure 2.7: The primal and dual polytopes. The reciprocation of the thick grey facet of the primal polytope results in the emphasized dual point.

Dualizing twice gives back the original polytope. This fact is used to find the outer vertices and explained further in the Vertex enumeration section in the following Method chapter.

3 Method

As mentioned in the previous chapter, the Pareto surface for convex MOPs can be approximated by a sandwich algorithm, which finds an outer and an inner approximation of the convex surface. Sandwich algorithms have the advantage that a bound of the approximation error can be determined as the difference between the outer and the inner approximations, see Figure 3.1a. In this thesis a type of sandwich algorithm suggested in [2] is used for this purpose. This dual algorithm, based on vertex enumeration, uses the error bound throughout the algorithm and thus it is known in each step of the algorithm and may be used as a stopping criteria.

The real-time navigation on the Pareto surface and visualisation tool for the treatment plan is based on [8]. Here convex combinations of Pareto optimal points are used as possible plans as they are feasible and easily found, thereby making real-time navigation possible. Since combinations of plans are used as potential solutions, the error bound gives a measure of how large the deviation from a Pareto optimal plan can be. It also ensures that the real solution is at least as good as the one found by the navigation.

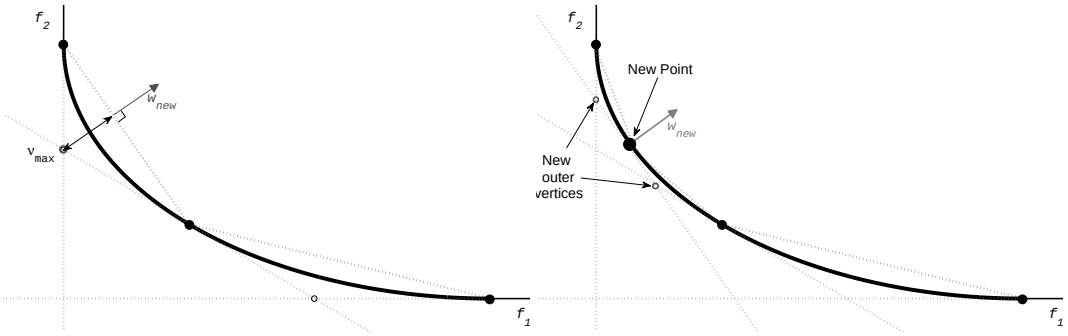
3.1 The Dual algorithm for approximation of the Pareto surface

Sandwich algorithms in general start off with a few starting points and then iteratively add one point at a time to improve the surface approximation.

In the dual algorithm improvement means lowering the bound on the approximation error, by adding a new point where the approximation error bound is the largest.

3.1.1 Initial step

Let n be the number of objectives and thus the dimension of objective space containing the Pareto optimal points. Initially $n + 1$ Pareto optimal points are found. The inner approximation consists of the lower hull of these points and the outer approximation is formed by the supporting hyperplanes of the Pareto surface. For each outer vertex, arising from the intersections of the supporting hyperplanes, the distance to the inner approximation is calculated and a new Pareto optimal



(a) The new weight found by solving (D) for $\boldsymbol{\nu}_{max}$.

(b) The new point is found by solving (WSP) for the new weight.

Figure 3.1: A new weight is found at the point where the error bound of the approximation is the largest. The new weight is used to add a new point, improving the approximation.

point is determined and added where the approximation error is the largest, see Figure 3.1.

Finding starting points

The initial step consists of finding n anchor points, where each anchor point $\boldsymbol{z}^{(i)}$ is found by minimizing the corresponding function f_i. This is equivalent to solving the (WSP) using a weight with $w_i = 1$ and $w_j = 0$ for all $j \neq i$. One additional point is used, an inner point found by weighing all functions equally, by using weight $\boldsymbol{w} = \frac{1}{n}\mathbb{1}$, where $\mathbb{1} = (1 \quad 1 \quad \cdots \quad 1)^T$. The hyperplane through point i has $\boldsymbol{w}^{(i)}$ as its normal.

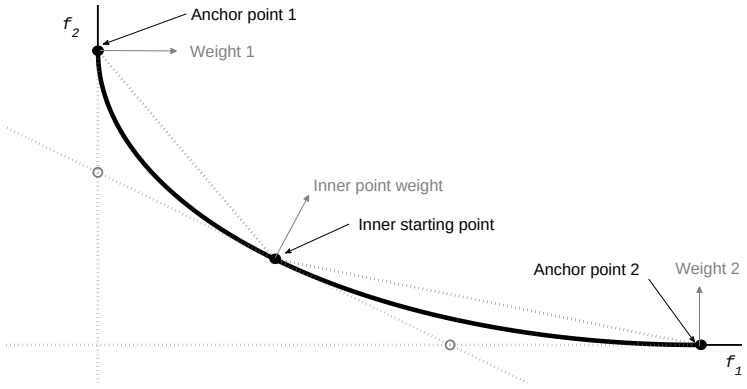


Figure 3.2: Inner and outer approximations.

The dual algorithm presented in [2], uses the outer vertices, marked with circles in Figure 3.2, when calculating the distances between the approximations, since

those are the points on the outer approximations furthest away from the inner approximation.

When the starting points are all found, the task is to find these vertices. The method of finding these outer vertices is based on duality of polytopes.

Vertex enumeration

The set of outer vertices can be seen as corners of the polytope formed by the convex hull of this set. To include the whole Pareto surface inside this polytope, the set has to be expanded with n bounds, see Figure 3.3. These bounds are only used to find the outer vertices and do not have to correspond to upper bounds of the functions used in the (WSP). However the bounds forming the polytope have to be large enough to cover values that might occur in the functions as a result of the weighted sum optimization, to be able to sort them out properly.

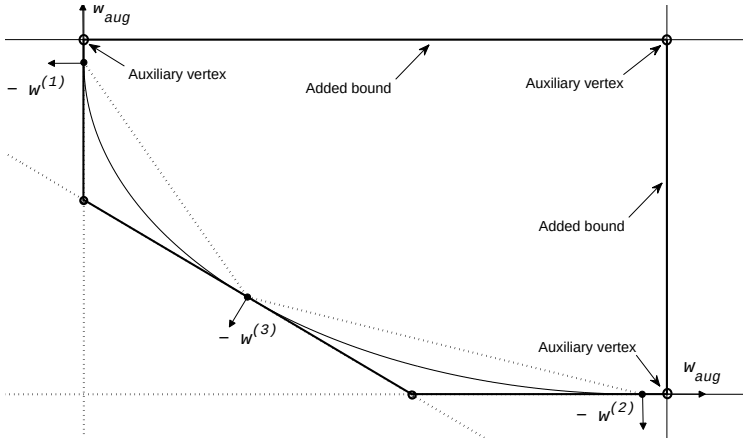
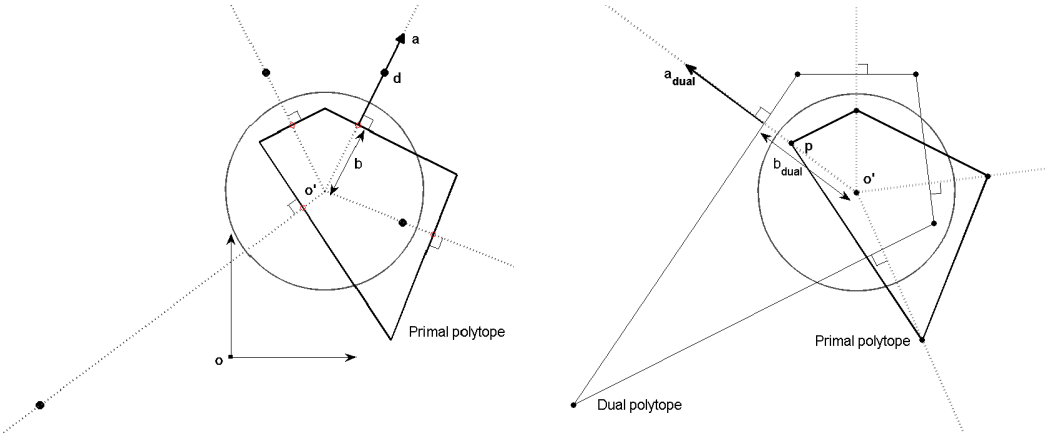


Figure 3.3: Bounds added to include the hyperplanes through the anchor points.

This expanded set of outer vertices will form our primal polytope, where the hyperplanes are known and vertices are to be found. The normals of the primal facet hyperplanes are known as the weights (with opposite signs) that gave the Pareto optimal points lying on these facets, see Figure 3.4a. To be able to dualize this polytope, it first has to be translated so that the origin is located in its interior. The new origin can be chosen as the average of the $n+1$ starting points and added vertices.

Dualization will give the vertices of a dual polytope, where each of the vertices corresponds to a facet in the primal polytope. Due to duality between the polytopes, the facets of the dual polytope will correspond to the vertices of the outer approximation, which is exactly what we are looking for, see Figure 3.4b. Remaining is to calculate the normals of the dual facets, to dualize them and to translate the resulting points back to the original coordinate system. Now the auxiliary vertices lying on the upper boundaries added at the start can be removed so that only the

true outer vertices remain. Pseudocode for the vertex enumeration algorithm can be found as Algorithm A.1 in the Appendix.



(a) Finding the dual points corresponding to the primal facets (supporting hyperplanes).

(b) Finding the primal vertices (vertices of outer approximation) corresponding to the dual facets.

Figure 3.4: Finding the primal vertices using dualization. $\boldsymbol{a}$ refers to the unit normal of the hyperplane and b is the distance from the origin to the hyperplane.

Calculating an upper limit of the approximation error

For each of the outer vertices the distance to the inner approximation needs to be determined. The distance measured in [2] is the distance which minimizes the largest component distance $d(\boldsymbol{\nu}, \boldsymbol{z}) = \max_i(z_i - \nu_i, 0)$, between an outer vertex $\boldsymbol{\nu}$ and a point $\boldsymbol{z}$. This can be written as an optimization problem in the following way:

$$\begin{cases} \text{minimize} & \eta \\ \text{subject to} & \eta \geq \max(z_i - \nu_i, 0) \end{cases}$$

$$\Updownarrow$$

$$\begin{cases} \text{minimize} & \eta \\ \text{subject to} & \eta \geq z_i - \nu_i \quad \forall i \\ & \eta \geq 0 \end{cases}$$

Now let $\boldsymbol{z} = \boldsymbol{P}^T\boldsymbol{\lambda} + \boldsymbol{I\mu}$, where $\boldsymbol{P}$ contains the Pareto optimal points found so far and $\boldsymbol{I\mu}$ allows $\boldsymbol{z}$ to lie above the inner approximation. The optimization problem

giving the sought distance can be rewritten as:

$$(\mathrm{P}) \begin{cases} \underset{\eta,\boldsymbol{\lambda},\boldsymbol{\mu}}{\text{minimize}} & \eta \\ \text{subject to} & \eta\mathbb{1} \geq \boldsymbol{P}^T\boldsymbol{\lambda} + \boldsymbol{I}\boldsymbol{\mu} - \boldsymbol{\nu} \\ & \mathbb{1}^T\boldsymbol{\lambda} = 1 \\ & \eta, \boldsymbol{\lambda}, \boldsymbol{\mu} \geq 0. \end{cases}$$

Since the problem minimizes η the point $\boldsymbol{z}$ will lie on the inner approximation and η will represent the sought distance. Thus this optimization problem is solved for each vertex of the outer approximation and the distance η is stored in association with that vertex. More on this issue in the section about updating the dual polytope.

3.1.2 Iterative algorithm

When all the distances from the outer vertices to the inner approximation are determined, the largest one is chosen and the corresponding vertex $\boldsymbol{\nu_{max}}$ is used to find the new weight, that can be used to find a new Pareto optimal point.

Adding a new Pareto optimal point

For the outer vertex $\boldsymbol{\nu_{max}}$ corresponding to the largest distance from the inner approximation, the dual to (P) is solved.

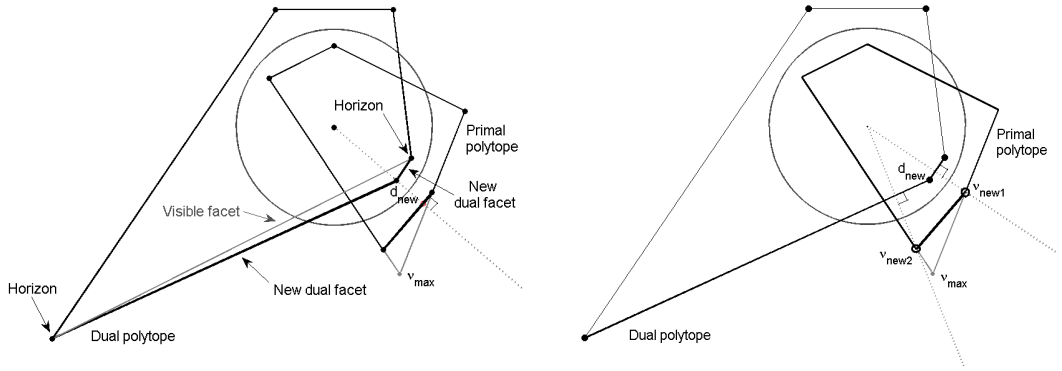
$$(\mathrm{D}) \begin{cases} \underset{\boldsymbol{\pi},\rho}{\text{maximize}} & \rho - \boldsymbol{\nu}^T\boldsymbol{\pi} \\ \text{subject to} & \boldsymbol{P}\boldsymbol{\pi} \geq \rho\mathbb{1} \\ & \mathbb{1}^T\boldsymbol{\pi} \leq 1 \\ & \boldsymbol{\pi} \geq 0. \end{cases}$$

By Proposition $A.5$ in [2], $\boldsymbol{\pi}$ will be the normal of the hyperplane that supports the inner approximation at the point $\boldsymbol{P}^T\boldsymbol{\lambda} + \boldsymbol{I}\boldsymbol{\mu}$, see Figure 3.1.

Therefore $\boldsymbol{\pi}$ is used as the new weight vector, which can be used to solve the Weighted Sum Problem (WSP) to give the next Pareto optimal point. This results in "cutting off" $\boldsymbol{\nu_{max}}$ from the former outer approximation (primal polytope) and creating at least n new outer vertices. These new vertices can be found by updating the dual polytope and dualizing its new facets, see Figure 3.5.

Updating the dual polytope

The new Pareto optimal point is dualized in the same way as before and the new dual facets yield the new outer vertices. Since the distance of the approximations only changes in the neighbourhood of the cut off vertex $\boldsymbol{\nu_{max}}$, there is no need to redetermine all of the vertex distances. Instead it would be convenient to keep track of neighbouring vertices and their distances. This can be achieved by relating outer vertices with the dual facets and to update the dual hull for each new added Pareto



(a) Dualizing the new facet (b) Finding the new outer vertices

Figure 3.5: Updating the dual polytope and finding the new outer vertices.

optimal point. This is done using a beneath and beyond approach, proposed in [2]. The suggested method checks which facets of the dual polytopes that are *visible* to the new dual point and finds the border between visible and invisible facets, called the horizon. A point $\boldsymbol{z}$ is visible to the facet described by the hyperplane $\boldsymbol{a}_{dual}^T \boldsymbol{z} = b$ if it lies in the halfspace $\boldsymbol{a}_{dual}^T \boldsymbol{z} \geq b$. From the horizon new facets are connected to the new dual point, see Figure 3.5a.

To find the horizon, saved as the ridges between visible and invisible facets, one starts with one visible facet and then using depth first search goes through its neighbours to check visibility. The adjacency of the facets can be represented by an adjacency matrix, where $\boldsymbol{A}(i,j) = 1$ if facets i and j are neighbours. The first visible facet can be found as the facet corresponding to $\boldsymbol{\nu_{max}}$. New facets are created by connecting horizon ridges with the new dual point. These facets are then dualized to find the new outer vertices, as seen in Figure 3.5b.

According to Proposition $A.6$ in [2] the distance from the new outer vertices to the inner approximation is at most as large as the largest distance from any of its neighbouring old vertices' distance to the inner approximation. This is why when creating a new facet its distance can be approximated by the largest distance of the two facets incident to the horizon ridge resulting in the new facet.

Using this fact not all vertex distances need to be recalculated. Only approximated distances larger than the largest known distance need to be determined with higher accuracy, which saves computations.

When the new outer vertices and new distances are computed the next iteration can start. As a stopping criterion, a minimum value for the distance between the approximations may be used. If the normalization of the functions is accurate, the distance corresponds to percentages of the function values, which is convenient.

The main steps of the dual algorithm can be seen in the flow chart 3.6.

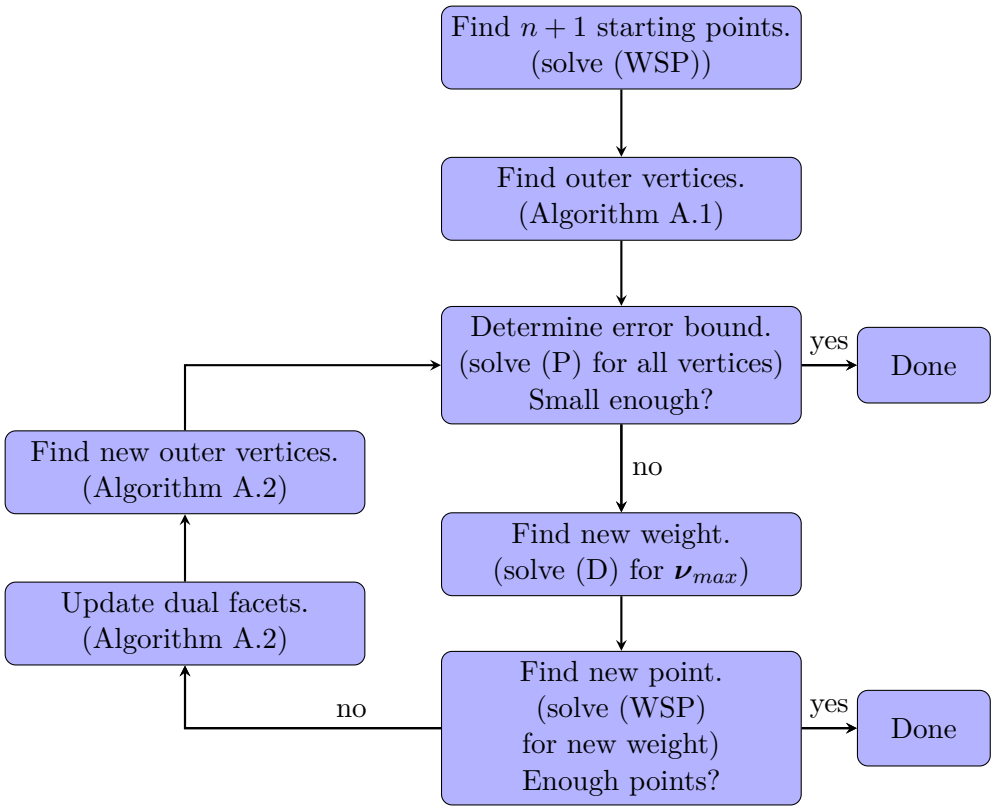


Figure 3.6: Main steps in the dual algorithm

3.2 Interpolation between Pareto optimal solutions

In our case the objectives $f_i(\boldsymbol{D}(\boldsymbol{x}))$ will be functions of the voxel doses $\boldsymbol{D}(\boldsymbol{x})$, depending on the solution $\boldsymbol{x}$ implicitly. Voxels are the 3-dimensional analogy of pixels that is a result of the discretization of the volumes of interest. The dose calculation can be performed as a matrix multiplication $\boldsymbol{D}\boldsymbol{x}$, where the matrix $\boldsymbol{D}$ has been constructed beforehand based on the fixed shot positions, found by the packing algorithm used to determine the shots.

When finding a suitable treatment plan only using the precomputed Pareto optimal plans is not enough since the number of plans is limited, so additional plans in between have to be approximated for the navigation. Since the Multiobjective Optimization Problem (MOP) is convex, linear combinations of Pareto optimal solutions $\boldsymbol{x}$ are also feasible. However, computing these are rather time consuming since the doses $\boldsymbol{D}(\boldsymbol{x})$ would first have to be calculated and used to determine the function values $\boldsymbol{f}(\boldsymbol{D}(\boldsymbol{x}))$. Instead [8] suggests to use linear combinations of Pareto optimal points, that is that the Pareto optimal solutions in the function space may be used instead. Let $\boldsymbol{X} = (\boldsymbol{x}^{(1)} \quad \boldsymbol{x}^{(2)} \quad \cdots \quad \boldsymbol{x}^{(m)})$ denote the matrix containing all solutions $\boldsymbol{x}$ and the set $\boldsymbol{X}\boldsymbol{\lambda}$, where $\mathbb{1}^T\boldsymbol{\lambda} = 1$ and $\boldsymbol{\lambda} \geq 0$, denote all convex

combinations of these solutions. Further let $\boldsymbol{f}$ be the bundle of objective functions and $\mathbf{b}$ contain the upper bounds of the function values. Then

$$\boldsymbol{f}(\boldsymbol{DX\lambda}) = \boldsymbol{f}(\sum_{i=1}^{m} \lambda_i \boldsymbol{Dx}^{(i)}) \leq \sum_{i=1}^{m} \lambda_i \underbrace{\boldsymbol{f}(\boldsymbol{Dx}^{(i)})}_{\boldsymbol{z}^{(i)}} \leq \sum_{i=1}^{m} \lambda_i \boldsymbol{b} = \boldsymbol{b},$$

where the first inequality holds due to convexity, see the Definition 2.3. This means that the convex combinations of Pareto optimal points $\boldsymbol{z}$ (middle part of inequality) are feasible and can be used for faster computations. When a satisfactory plan is found the convex combination of Pareto optimal solutions $\boldsymbol{x}$ (left hand side in the inequality) can be calculated, which gives a plan which is at least as good.

3.3 Navigating among the pre-calculated plans

The updates in the current plan in the visualization tool are made according to the method described in [8].

The navigation starts off with one plan and proceeds by letting the planner choose a function value to lower. When an objective function value is changed for one of the functions, the tool will search for a feasible solution with that chosen function value.

It is also possible to change upper bounds for objective function values, which will result in changes of the ranges of the other objective functions values.

3.3.1 Finding a new plan

When a target value τ of function j is chosen, the goal is to find a new plan $\boldsymbol{z}$, by minimizing the largest component distance to a feasible point.

$$\begin{cases} \text{minimize} & \max\limits_{i \neq j}\{z_i - z_i^{current}\} \\ \text{subject to} & \boldsymbol{z} = \boldsymbol{f}(\boldsymbol{DX\lambda}) + \boldsymbol{I}\boldsymbol{s} \\ & \boldsymbol{z} \leq \boldsymbol{b} \\ & z_j = \tau \\ & \mathbb{1}^T \boldsymbol{\lambda} = 1 \\ & \boldsymbol{\lambda}, \boldsymbol{s} \geq 0. \end{cases}$$

As suggested $\boldsymbol{f}(\boldsymbol{DX\lambda})$ is replaced by $\boldsymbol{P}^T\boldsymbol{\lambda} = \sum_{i=1}^{m} \lambda_i \boldsymbol{f}(\boldsymbol{Dx}^{(i)})$, where $\boldsymbol{P}^T = (\boldsymbol{f}(\boldsymbol{Dx}^{(1)}) \quad \boldsymbol{f}(\boldsymbol{Dx}^{(2)}) \quad \ldots \quad \boldsymbol{f}(\boldsymbol{Dx}^{(m)}))$ contains all Pareto optimal points. This is similar to calculating the distances between the approximations in (P) and is handled in the same way by:

$$\text{minimize } \max_{i \neq j}\{z_i - z_i^{current}\} \quad \Longleftrightarrow \quad \begin{cases} \text{minimize} & y \\ \text{subject to} & y \geq z_i - z_i^{current} \quad i \neq j \end{cases}$$

This results in the following problem to be solved:

$$\begin{cases} \text{minimize} & y \\ \text{subject to} & y = (\boldsymbol{P}^T\boldsymbol{\lambda})_i + s_i - z_i^{current} \quad i \neq j \\ & \boldsymbol{P}^T\boldsymbol{\lambda} \leq \boldsymbol{b} \\ & (\boldsymbol{P}^T\boldsymbol{\lambda})_j = \tau \\ & \mathbb{1}^T\boldsymbol{\lambda} = 1 \\ & \boldsymbol{\lambda}, y, \boldsymbol{s} \geq 0. \end{cases}$$

Obtained from this optimization problem are the optimal convex coefficients $\boldsymbol{\lambda}^*$, which give the new approximately Pareto optimal point by $\boldsymbol{z} = \boldsymbol{P}^T\boldsymbol{\lambda}^*$. $\boldsymbol{\lambda}^*$ may be used to calculate a better approximation from $\boldsymbol{f}(\boldsymbol{X}\boldsymbol{\lambda}^*)$.

Notice that $\boldsymbol{P}^T$ only contains points that satisfy the restrictions chosen in the navigation.

4 Model

In this thesis both the Pareto surface approximation algorithm [2] and the Graphical User Interface [8] have been implemented in MATLAB®.

Each objective function consists of a sum of two convex, piecewise quadratic functions so that the (WSP) could be solved using 'interior-point-convex' in quadprog [4, 6, 3]. When a solution is found the functions are evaluated for that solution to find coordinates in objective space. That is why the discrepancy between the found optimal value and the evaluated function value for the corresponding solution has to be small. To get the difference between the optimal objective function values $\boldsymbol{z}^*$ and the value from evaluating the functions for the optimal solution $\boldsymbol{f}(\boldsymbol{x}^*)$, to be less than 10^{-5}, all tolerances were put to 10^{-12}. $\boldsymbol{z}^*$ is the optimal value given by the solver when minimizing each function separately and $\boldsymbol{x}^*$ is the corresponding optimal solution. The tolerance values were found by solving the (WSP) for the first n anchor points.

The linear programs used for the distance calculations were solved with the default 'interior-point' in MATLAB®'s linprog [9, 6], with the tolerance 10^{-5}, since this was the accuracy of the points.

4.1 Data

The test cases used in this thesis are all based on the same patient data with the tumour and organ volumes discretized into voxels. Coordinates of each voxel are given, together with the information of which voxels constitute which organ or tumour. The dose matrix $\boldsymbol{D}$, that only considers voxels constituting the tumour and organ volumes, is given and has been determined when the shot positions were fixed prior to the dual algorithm was used.

All of the test cases contain one tumour, but differ in the number of Organs At Risk (OARs) in a way such that test case $j+1$ contains one more OAR than test case j, see Figures A.1-A.5 in Appendix. Since each of the organs in the test case is represented by one objective function, the number of functions increases for each test case as well as the size of the dose matrix $\boldsymbol{D}$ since the number of voxels increases. In addition to the objective functions corresponding to the physical organs and tumours a function corresponding to the total treatment time is used.

4.2 Optimization

The objective functions have been chosen quadratic and linear so that the (WSP)s could be expressed as $\frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} + \boldsymbol{f}^T\boldsymbol{x}$ and solved using quadprog's 'interior-point-convex' algorithm. To ensure unique solutions and thus Pareto optimal anchor points the organ functions were made strictly convex. However this was not possible for the treatment time function that was added at end of the project and hence the weighted sum of all functions is not guaranteed to be strictly convex.

4.2.1 Variables

The variables $x_{i,j,k}$ considered are the beam-on times for sector j and collimator size k of shot i and organized into a vector as follows:

$$\boldsymbol{x} = \begin{pmatrix} x_{1,1,1} \\ x_{1,1,2} \\ x_{1,1,3} \\ x_{1,2,1} \\ \vdots \\ x_{N,8,3} \end{pmatrix}$$

4.2.2 Objective functions

The functions correspond to penalizations of unwanted voxel doses, such that low doses for the target and high doses for the Organ At Risks (OARs) are punished. The total treatment time can be expressed as a convex function and can therefore be penalized directly.

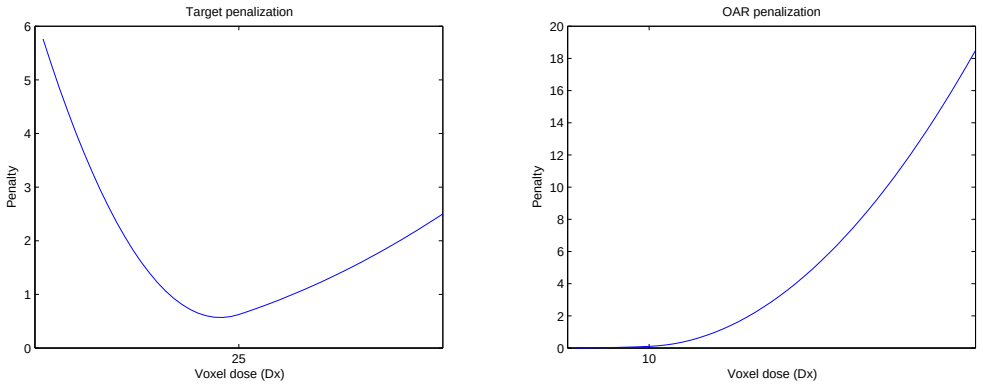
Target and organ objective functions

The organ objective functions are constructed to mainly penalize the one-sided deviation from a threshold level. For a target, mainly deviations for voxel doses below the minimum target dose l_T are penalized and for an OAR mainly deviations above the maximum OAR dose l_R are penalized, such that $l_R \leq l_T$. These penalizations are then summed up and constitute the corresponding function value. To achieve strict convexity higher doses in the target are also penalized, but much less, as well as low doses for OARs, see Figure 4.1.

The organ function expressions are presented next.

Target objective function:

$$\begin{aligned} f_t &= \alpha_t(\boldsymbol{D}_t\boldsymbol{x})^T(\boldsymbol{D}_t\boldsymbol{x}) + \beta_t(\max(l_T\mathbb{1} - \boldsymbol{D}_t\boldsymbol{x}, 0))^T(\max(l_T\mathbb{1} - \boldsymbol{D}_t\boldsymbol{x}, 0)) \\ &= \boldsymbol{x}^T(\alpha_t\boldsymbol{D}_t^T\boldsymbol{D}_t)\boldsymbol{x} + (\max(l_T\mathbb{1} - \boldsymbol{D}_t\boldsymbol{x}, 0))^T(\beta_t\boldsymbol{I})(\max(l_T\mathbb{1} - \boldsymbol{D}_t\boldsymbol{x}, 0)) \end{aligned}$$



(a) Penalization of voxel doses in target (b) Penalization of voxel doses in OAR

Figure 4.1: Voxel dose penalizations

α_t and β_t are parameters, such that $\alpha_t < \beta_t$ and $\boldsymbol{D}_t$ contains the rows of the dose matrix $\boldsymbol{D}$ corresponding to target voxels. The max function yields a vector containing the penalizations of each of the voxels in the target.

OAR objective function:

$$f_r = \boldsymbol{x}^T(\alpha_r \boldsymbol{D}_r^T \boldsymbol{D}_r)\boldsymbol{x} + (\max(\boldsymbol{D}_r\boldsymbol{x} - l_r\mathbb{1}, 0))^T(\beta_r \boldsymbol{I})(\max(\boldsymbol{D}_r\boldsymbol{x} - l_r\mathbb{1}, 0)),$$

where α_r, β_r and $\boldsymbol{D}_r$ are parameters belonging to the OARs.

In my project $\alpha_t = \alpha_r = 0.001$ and $\beta_t = \beta_r = 0.01$. Of course either α or β can be eliminated and the other be modified to reduce the number of parameters. They can also have different values for different OARs.

The max-functions are dealt with as in the prior cases, namely by minimizing the help variables $\boldsymbol{y}_i$:

$$\begin{aligned}
&\text{minimize} && \boldsymbol{y}_i \\
&\text{subject to} && \boldsymbol{y}_i \geq \begin{cases} \max(l_t\mathbb{1} - \boldsymbol{D}_t\boldsymbol{x}, 0) & \text{if } i \text{ target} \\ \max(\boldsymbol{D}_r\boldsymbol{x} - l_r\mathbb{1}, 0) & \text{if } i \text{ OAR} \end{cases} \\
&\qquad \Updownarrow \\
&\text{minimize} && \boldsymbol{y}_i \\
&\text{subject to} && \boldsymbol{y}_i \geq \begin{cases} l_t\mathbb{1} - \boldsymbol{D}_t\boldsymbol{x} & \text{if } i \text{ target} \\ \boldsymbol{D}_r\boldsymbol{x} - l_r\mathbb{1} & \text{if } i \text{ OAR} \end{cases} \\
& && \boldsymbol{y}_i \geq 0,
\end{aligned}$$

and $\boldsymbol{y}_i$ is the vector of voxel penalizations belonging to volume i. Since the objective function is minimized it can be written as

$$f_i = \boldsymbol{x}^T(\alpha_i \boldsymbol{D}_i^T \boldsymbol{D}_i)\boldsymbol{x} + \boldsymbol{y}_i^T(\beta_i \boldsymbol{I})\boldsymbol{y}_i$$

with the constraints above.

Total treatment time

The total treatment time is too penalized, but linearly, and can be expressed as the sum of the times of each shot. The time of each shot is the largest time of all of sectors' times for that shot, and the time of each sector and shot is obtained by adding the times of each collimator size for that sector and shot. The time of shot k, t_k, is thus given by $t_k = \max_j\{c_{k,j}\}$, where $j = 1, \ldots, 8$ and corresponds to the sectors and $k = 1, \ldots, N$ corresponds to the shots.

$$\begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix} = \begin{pmatrix} \max(c_{1,1}, c_{1,2}, \ldots, c_{1,8}) \\ \max(c_{2,1}, c_{2,2}, \ldots, c_{2,8}) \\ \vdots \\ \max(c_{N,1}, c_{N,2}, \ldots, c_{N,8}) \end{pmatrix}$$

All sector-shot times $c_{k,j}$ are given by a matrix multiplication:

$$\boldsymbol{F}_t^T \boldsymbol{x} = \begin{pmatrix} x_{1,1,1} + x_{1,1,2} + x_{1,1,3} \\ x_{1,2,1} + x_{1,2,2} + x_{1,2,3} \\ \vdots \\ x_{N,8,1} + x_{N,8,2} + x_{N,8,3} \end{pmatrix} = \begin{pmatrix} c_{1,1} \\ c_{1,2} \\ \vdots \\ c_{N,8} \end{pmatrix}, \longleftarrow \text{time for shot 1 in sector 2}$$

where

$$\boldsymbol{F}_t = \begin{pmatrix} \mathbb{1} & 0 & \cdots & 0 \\ 0 & \mathbb{1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbb{1} \end{pmatrix}, \quad \mathbb{1} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Thus $t_k \geq c_{k,j}$ for all j since t_k is minimized. We can then write

$$\boldsymbol{A}_t \boldsymbol{t} = \begin{pmatrix} t_1 \\ \vdots \\ t_1 \\ \vdots \\ t_N \\ \vdots \\ t_N \end{pmatrix}$$

using

$$\boldsymbol{A}_t = \begin{pmatrix} \mathbb{1} & 0 & \cdots & 0 \\ 0 & \mathbb{1} & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & \mathbb{1} \end{pmatrix}, \quad \mathbb{1} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

and the constraint becomes $\boldsymbol{A}_t \boldsymbol{t} \geq \boldsymbol{F}_t^T \boldsymbol{x}$. And the sum of these shot times again is the total treatment time, so now we have a simple expression for the treatment time.

Treatment time objective function:

$$f_T = \mathbb{1}^T \boldsymbol{t},$$

where $\boldsymbol{t}$ is a vector of all shot times.

Forming the weighted sum

The functions were scaled by a factor reflecting the number of voxels in the organ since the amount differed by a factor 1000 between the organs. This scaling factor is not vital since normalization should take care of this matter, however it might affect the numerical accuracy of the ideal and nadir vector. The weighted sum to be minimized becomes

$$\begin{aligned} \sum_{i=1}^{n} w_i \tilde{f}_i &= \sum_{i=1}^{n} w_i \frac{f_i - z_i^{id}}{z_i^{nad} - z_i^{id}} = \sum_{i=1}^{n} w_i \frac{f_i}{z_i^{nad} - z_i^{id}} - \underbrace{\sum_{i=1}^{n} w_i \frac{z_i^{id}}{z_i^{nad} - z_i^{id}}}_{\text{constant} := \text{c}} \\ &= \sum_{i=1}^{n-1} w_i \cdot \frac{1}{z_i^{nad} - z_i^{id}} \cdot \frac{1}{n_i} (\boldsymbol{x}^T (\alpha_i \boldsymbol{D}_i^T \boldsymbol{D}_i) \boldsymbol{x} + \boldsymbol{y}_i^T (\beta_i \boldsymbol{I}) \boldsymbol{y}_i) \\ &+ w_n \cdot \frac{1}{z_i^{nad} - z_i^{id}} \cdot \mathbb{1}^T \boldsymbol{t} - \text{c}, \end{aligned}$$

where n is the number of objective functions and c is a constant. This can now be written as a (WSP):

$$\text{(WSP)} \begin{cases} \underset{\boldsymbol{x},\boldsymbol{t}}{\text{minimize}} & \sum_{i=1}^{n-1} w_i \cdot \frac{1}{z_i^{nad} - z_i^{id}} \cdot \frac{1}{n_i} (\boldsymbol{x}^T \alpha_i \boldsymbol{D}_i^T \boldsymbol{D}_i \boldsymbol{x} + \boldsymbol{y}_i^T \beta_i \boldsymbol{y}_i) \\ & + w_n \cdot \frac{1}{z_n^{nad} - z_n^{id}} \cdot \mathbb{1}^T \boldsymbol{t} \\ \text{subject to} & \boldsymbol{y}_i \geq l_i \mathbb{1} - \boldsymbol{D}_i \boldsymbol{x} \quad \text{if } i \text{ target} \\ & \boldsymbol{y}_i \geq \boldsymbol{D}_i \boldsymbol{x} - l_i \mathbb{1} \quad \text{if } i \text{ OAR} \\ & \boldsymbol{y}_i \geq 0 \\ & \boldsymbol{A}_t \boldsymbol{t} \geq \boldsymbol{F}_t^T \boldsymbol{x} \\ & \boldsymbol{x}, \boldsymbol{t} \geq 0. \end{cases}$$

4.3 Graphical user interface

The goal now is to make a visualization tool for comparing the different optimal treatment plans representing the Pareto surface.

Medical images are usually used to display tumours and organs together with dose distribution. Another representation of the dose distribution that is commonly used together with the medical images is a Dose Volume Histogram (DVH), where the x-axis represents the amount of dose and the y-axis corresponds to the fraction of the organ volume that receives that amount of dose.

The objective functions and their values are represented by sliders, where the slider thumb represents the value of the current plan and the slider window corresponds to the range of the function values from the approximated points. Ideally these ranges would be [0,1], but this might not be the case due to the approximation of the nadir vector, so the values are normalized according to the pre-computed values found.

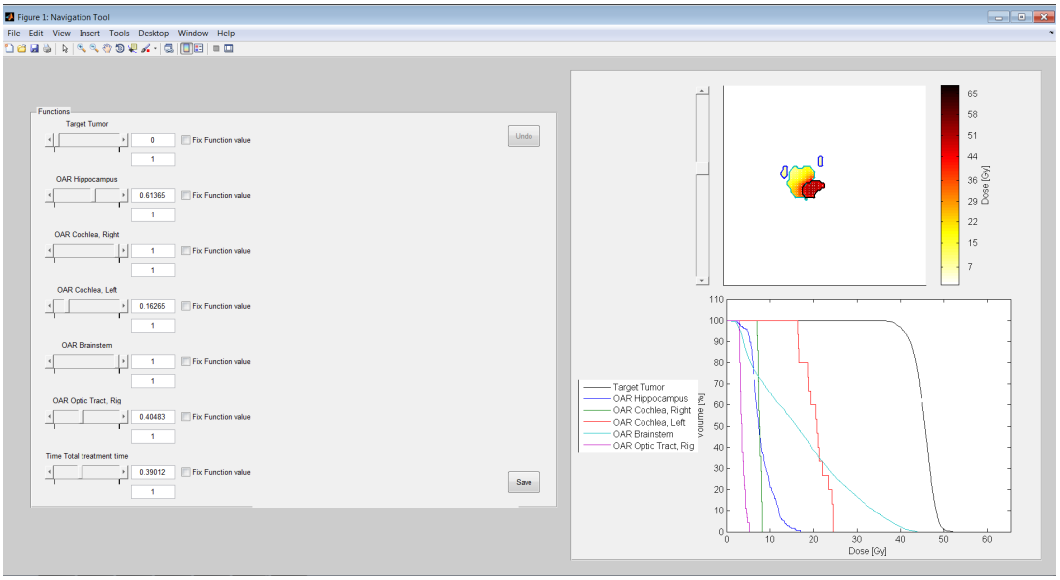


Figure 4.2: The navigation tool. On the left are the sliders corresponding to the objective functions. The upper boxes corresponds to the function value and the lower boxes the upper restrictions which are shown as tick marks below the sliders. The check boxes may be checked to keep the function value fixed. On the upper right hand side there is a figure showing the organs and the dose distribution. On the lower right hand side the DVH-curves are shown for the different organs and the tumour.

4.3.1 Dose distribution plots

The dose distribution is commonly presented by iso-dose curves in the medical images, where the area inside each level curve has received the amount of dose represented by that curve. Since the dose matrix $\boldsymbol{D}$ only contains information

about the dose of the targets and organs at risk, the dose outside those organs is unknown. Therefore the dose is illustrated by different colour intensities instead of iso-dose curves in the navigation tool of this project. A slider beside the image handles which 2D image in z-direction is shown. This provides an easy way to represent 3D information in good detail.

DVH curves are also shown for all tumours and organs considered in the optimization.

4.3.2 Function sliders

Each objective function will be represented by a slider handling restriction of the function value for that specific objective function. By moving the slider thumb the planner chooses a specific function value that results in searching for a new feasible point corresponding to that value. This leads to changes in the other sliders as well as the dose distribution and the dose-volume histogram.

Next to each slider there is a check box enabling the user to fix function values. This feature was not included in [8], but added since it seemed like a useful feature. At most $n-2$ check boxes can be checked, since there needs to be at least two free function values, one for which the value can be chosen by moving the slider and the other to be determined.

There is also a possibility to put upper bounds on the function values, and thus consider less plans in the navigation. As pointed out in [8], these bounds may have been hard to foresee, which is why they are not included in the original (WSP). Restricting upper bounds of the functions filters out Pareto optimal points with function values larger than that and may result in changes in the upper bounds of the functions. The upper bounds are represented by tick marks beneath the sliders.

Displayed function values for the organs are normalized with respect to the maximum values obtained in the pre-calculation. Since the treatment time slider is normalized as well, the treatment time is displayed separately to the right of the function slider.

5 Results

A major cause of concern in the optimization of dose plans is the computational time and so the effect of different parameters on this matter was examined. The computational time plays an important role since the longer the algorithm runs, the better the approximation gets.

All code in this project was written in MATLAB® and not optimized for speed, the computational times are mostly interesting for comparative reasons. It should be said that the concern here only regards the time for pre-computing plans, since the navigation is done in real-time.

A comparison between the dual algorithm and uniformly distributed random weights was done with respect to the error bound of the approximations and the computational time.

Since the degrees of freedom of the problem have an impact on the computational time the dimensionality of the solutions was studied to see if the degrees of freedom might be reduced to speed up computations.

The test cases used can be seen in the Appendix. Each test case has one tumour, which is the same in all cases, and one objective function corresponding to the total treatment time. Test case 1 represents one tumour and one OAR and is thus represented by three objective functions and the last test case, number 5, which contains five OARs is represented by seven functions. The objective functions are formulated in the Model chapter. The number of shots and their positions in all of the test cases were the same, since they originate from the same patient data.

5.1 Comparison between the dual algorithm and uniform weights

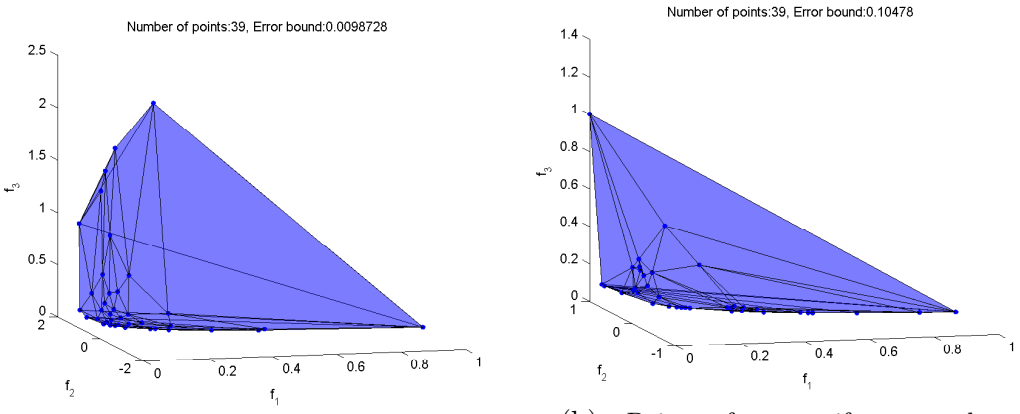
In this section the speed and error bound decrease of the dual algorithm was investigated by comparing it to an algorithm which randomly chooses weights to find the next Pareto optimal point.

The comparison was made using uniformly distributed random weights on test case 1 (one tumour, one OAR and one treatment time function, see Figure A.1), thereby skipping the time consuming calculations related to the dual polytope. Since this yields more points in the same amount of time, it would be interesting to see if this might give a comparable result in terms of the approximation error.

The uniformly distributed weights were generated from the symmetric Dirichlet distribution, giving evenly distributed normalized weights. Since the error bound is not known in the uniform weight algorithm, the dualization technique for finding outer vertices and calculating their distances was used for this purpose, but not considered in the timekeeping of the algorithm.

The dual algorithm was run until an approximation error bound of 0.01 was achieved. The goal was to run the uniform weight algorithm until it reached the same error bound, but this simply was not possible due to the long computational time, so I had to settle for a disappointing 0.1 bound.

The sets of points are plotted and compared as well as the decrease of the error bound for both algorithms. If the functions are properly normalized the error measure represents percent of the function values.



(a) Points from dual algorithm

(b) Points from uniform random weights

Figure 5.1: Pareto optimal points found in test case 1 containing 3 objective functions.

Figure 5.1 shows the first 39 Pareto optimal points found for both algorithms. As expected the dual algorithm does a better job in approximating the Pareto surface with higher accuracy. The difference in the final error bound is about a factor 10. Interestingly even for 3219 Pareto optimal plans, plotted in Figure 5.2, the uniform weights algorithm does not come close when it comes to the approximation error, still differing by about a factor 10. This follows from the fact that most of the time the uniform weights generated add points where the approximation does not need improvement.

This can also be seen in the plot of the decrease of the error bound, shown in Figure 5.3, where the blue solid curve representing the error bound of the dual algorithm decreases fast, while the black dashed curve corresponding to the random weights has a tendency to stay constant for several iterations between the sudden decreases when a 'good' weight is generated. Two different plots were made, the first corresponding to the error bound decrease per time unit. However since there

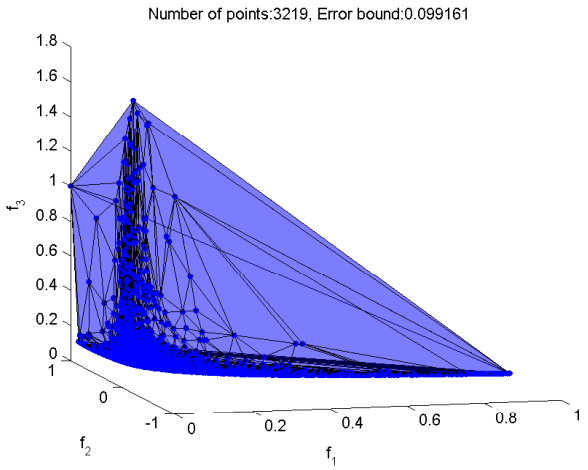
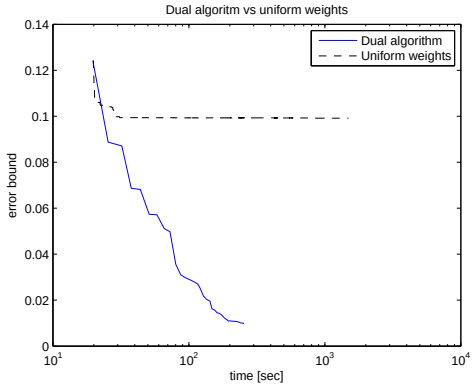
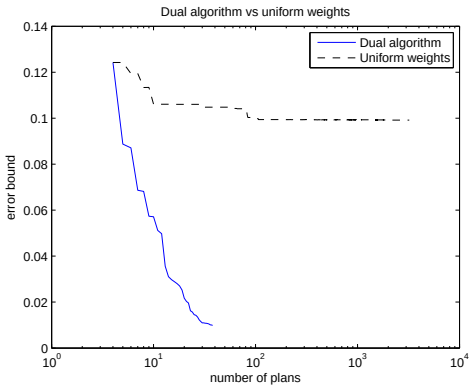


Figure 5.2: All the points found by uniform weights when reaching an error bound below 0.1.

might be room for improvement when it comes to running time of the dual code, it is relevant to see how the error bound decreases for each iteration as well.



(a) The error bound as a functions of time



(b) The error bound of each iteration

Figure 5.3: Plots of the error bound decrease for the case with 3 functions. The solid curves correspond to the error bound of the dual algorithm and the dashed curves corresponds to the uniform weight algorithm. Note that the x-axis is logarithmic to give more details in the beginning of the plot.

The conclusion is that either way the dual algorithm is superior to the uniform weights generating algorithm when it comes to lowering the approximation error.

I should point out that the result of the uniform algorithm might differ between different runs due to the randomness in generating the new weights. The huge number of points produced at the end though suggests that the result should not deviate too much from what one could expect as an average of a number of runs.

Further the comparison was only done for the smallest problem and might yield different results and lead to other conclusions for larger problems as the computational time of the dual algorithm increases, as seen in the following section.

5.2 Computational time

In this section the influence of different parameters on the computational times is investigated. The parameters that are expected to matter for the computational times was the size of the dose matrix, which in turn depends on the number of machine parameters (3 collimator sizes x 8 sectors = constant), the number of shots and the number of voxels. Since the number of shots in all of the test cases were the same, namely 19 shots, the effect of changing the number of shots has not been looked into. Thus the remaining thing to investigate was the number of voxels.

Another thing that could affect the computations is the number of functions, since this increases the dimension of the function space and thereby the number of computations relating to the dual polytope. The effect of the number of dual facets in each iteration on the computational time was also studied, as in [2], since updating the dual polytope uses a depth first search among these facets. The dual facets correspond to the number of outer vertices, which intuitively increase with the dimension on the function space.

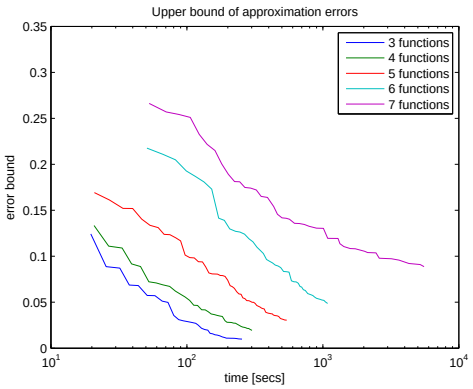
5.2.1 The number of functions

The main goal is to get a good approximation, but since this is highly connected to the computational time, the decrease of the error bound was studied for the different test cases. First the decrease per time unit in the test cases was studied as well as the decrease for each new plan added in the algorithm. The iteration tolerance for each of the problems was set to: 0.01, 0.02, 0.03, 0.05 and 0.09. The two latter had to be adjusted due to the otherwise very long computational time.

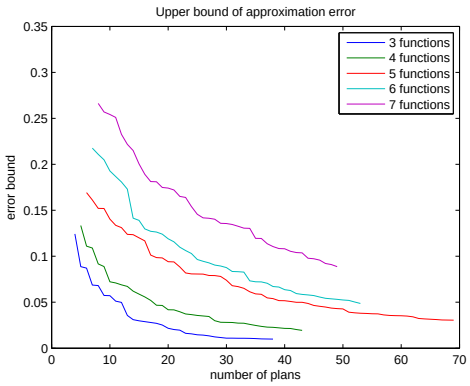
As seen in Figure 5.4 the initial bound of the approximation error is larger for higher dimensional problems. The x-axis again is log scaled to show details, but it can be seen that the decrease in error is slower per time unit for higher dimensional problems.

The time for finding a new plan in each of the cases is plotted together with the number of dual facets (outer vertices) to see how they correlate, see Figure 5.5. The plot suggests that there is some sort of correlation, however the last test case deviates from the others in the sense that the time to find the last handful of plans skyrockets without the number of facets showing the same kind of behaviour. This probably needs to be looked into further in case that there are other factors contributing to the computational time.

The results suggests that it is wise to be cautious when choosing objective functions to avoid slowing the algorithm down more than necessary.



(a) The error bound as a functions of time



(b) The error bound of each iteration

Figure 5.4: The figures show how the error bound decreases as a function of time and by each iteration for a different number of objective functions.

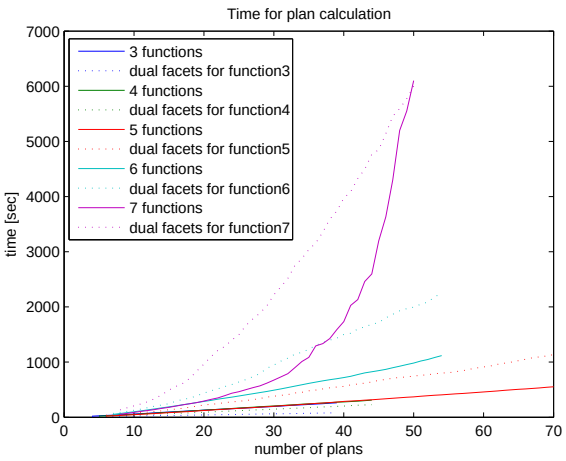


Figure 5.5: A plot showing the number of dual facets (dotted) and the computational times at each iteration.

5.2.2 The number of voxels

In this section the effect on the computational time of the total number of voxels was examined.

The brainstem, which is added as an OAR in the fourth test case (Figure A.4) contains a considerable amount of voxels, which might affect the computational time, since it impacts the size of the dose matrix $\boldsymbol{D}$ that is used in the optimization. The relation between the number of voxels and the total computational time for 39 plans is shown in Table 5.1. The computational times for solving the (WSP) for the solution with $\boldsymbol{w} = \frac{1}{n}\mathbb{1}$, is also checked. All times were found by using MATLAB®s `tic` and `toc` and some caution should be taken to the precision of the numbers. The optimization time in each test case is taken as an average over five different runs for the same weight.

# functions	# voxels	optimization time one solution [s]	total time 39 solutions [s]
3	2019	9.63	262
4	2033	9.95	272
5	2048	8.93	276
6	5473	22.2	699
7	5518	22.9	1589

Table 5.1: Table showing for each of the test cases the number of voxels, the optimization time of one solution and the total computational time for the first 39 points.

It seems that the time of solving the (WSP) for this particular solution is affected by the amount of voxels ($\Leftrightarrow$ rows of $\boldsymbol{D}$) in a linear manner and there is no reason to think this result would differ for any other weight $\boldsymbol{w}$. This indicates that the optimization time depends mostly on the size of the dose matrix $\boldsymbol{D}$. The pattern is not the same for the total time, which suggest there is something else affecting the total computational time, than only the optimization time.

Both Figure 5.5 and Table 5.1 indicate that there is something other than the number of voxels affecting the total computational times, since the large computational times should otherwise be reflected in the results of test case 4 as well.

A probable cause is the number of recursions when searching for the dual horizon. The reason for finding this dual horizon in the first place was to update the dual polytope that stores information about the vertex distances to the inner approximation, and in that way save computations and thus computational time. For cases with many objective functions however the matter of handling the adjacency of outer vertices could dominate the computations and obviously there is a point where it would be faster to re-calculate each vertex distance in the iterations to avoid the tedious dual polytope calculations.

5.3 Dimensionality analysis

If a lot of the variables are correlated they could be reduced to speed up computations and thus allow reaching higher accuracy of the approximation. Therefore a dimensionality analysis was done using Principal Component Analysis (PCA) on the solutions.

The same kind of analysis was done on the Pareto optimal points to see if they could be represented by fewer dimensions indicating that some objective functions were correlating.

5.3.1 Principal Component Analysis

PCA looks at eigenvalues of the covariance matrix of the data giving information about the directions in the data where the variance is the largest. The data point $\boldsymbol{x}$ can then be reduced by transforming it to a lower dimension, by $\boldsymbol{y}_k = \boldsymbol{W}_k\boldsymbol{x}$, only considering the k directions of largest variance and in that way keep as much information as possible. $\boldsymbol{X}$ here contains the solutions $\boldsymbol{x}$ of size d, translated to have zero mean. $\boldsymbol{W}_k$ contains the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix of $\boldsymbol{X}$, $Cov(\boldsymbol{X})$, and $\boldsymbol{Y}_k$ hence is the data with reduced dimensionality. The reconstructed solution $\boldsymbol{x}_k$ can be given by $\boldsymbol{x}_k = \boldsymbol{W}_k^T\boldsymbol{y}_k$. If the number of principal components k is equal to the number of initial dimensions d, no information is lost.

5.3.2 Dimensionality of solutions

First the dimensionality of the solutions was investigated, since a reduction would make it possible to transform the original (WSP) using $\boldsymbol{x}_k = \boldsymbol{W}_k^T\boldsymbol{y}_k$ as a new constraint. This could then be substituted into the objective functions $\boldsymbol{f}_k(\boldsymbol{D}\boldsymbol{W}_k^T\boldsymbol{y}_k) = \boldsymbol{f}_k(\boldsymbol{D}_k\boldsymbol{y}_k)$ to be minimized for the new lower dimensional variable $\boldsymbol{y}_k$.

Since the degrees of freedom are many in this case, 3 x 8 x 19 = 456, all of the solutions were used, from each of the test cases as well as the ones from the uniform weight algorithm. This yields a total of 3477 solutions.

The eigenvalues λ_i of the covariance matrix can be seen in Figure 5.6.

The dimensionality was first reduced by using the smallest k such that $\frac{\sum_{i=1}^{k}\lambda_i}{\sum\lambda} \geq 0.999$, which resulted in 218 principal components. The other fraction used was 0.99, yielding $k = 111$.

The residuals between the voxel doses $\boldsymbol{D}\boldsymbol{x}_k$ generated by these reduced solutions and the real voxel doses $\boldsymbol{D}\boldsymbol{x}$ were compared. The DVH curves of the solutions corresponding to the largest residuals for the two fractions were plotted. The x-axis corresponds to the delivered dose and the y-axis corresponds to the fraction of volume that has received that dose. Both these plots can be seen in Figure 5.7.

Figure 5.7b shows considerable deviations and the search for a proper value of k was continued manually. One thing that had to be considered was that the PCA was performed on the solutions $\boldsymbol{x}$, so the maximum residual $\|\boldsymbol{x} - \boldsymbol{x}_k\|$ decreases for

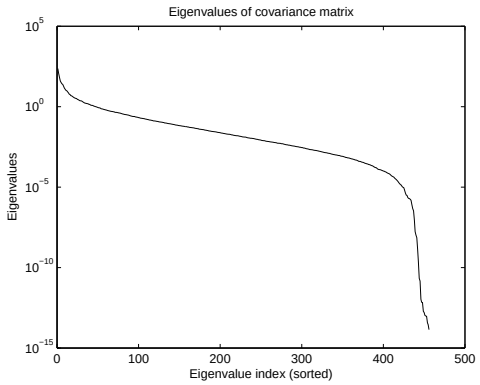
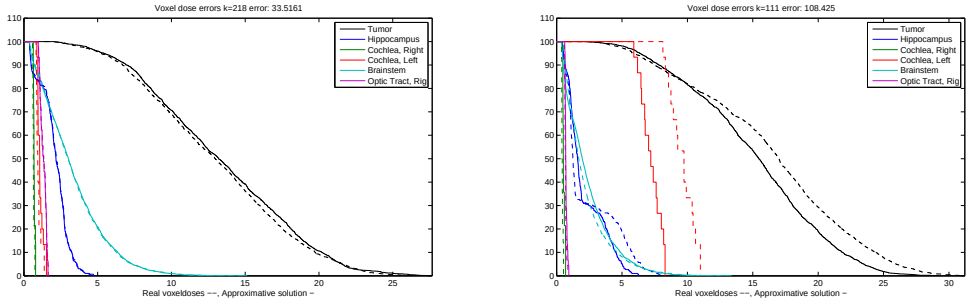


Figure 5.6: The eigenvalues of the covariance matrix of the solutions. The y-axis is logarithmic to show detail for small numbers.



(a) The DVH of plan with $k = 218$.

(b) The DVH of plan with $k = 111$.

Figure 5.7: The solid line represents the distribution from the reduced solutions and the dashed line represents the corresponding real solution. Both plots show the voxel doses of the plan with largest voxel residuals.

each additional principal component used, however the maximum residual of the voxel doses $\|\boldsymbol{Dx} - \boldsymbol{Dx}_k\|$ may not. Both these maximum residuals for an increasing value of k are plotted and can be seen in Figure 5.8.

The manual search for the number of principal components to consider resulted in $k = 145$ with not too large deviations in the voxel doses as seen in Figure 5.9.

A possibility of reducing dimensionality by this amount would have an impact on the optimization part of the algorithm. However the results of the optimization performed on reduced variables have not been studied, and whether or not this would yield similar results is uncertain. My results only suggest that this should be looked into. To keep in mind also is that my results originate from the same patient and same shot positions.

However if one wants to have more data and compare different patients a new problem arises since the solutions $\boldsymbol{x}$ depend on shot positions. Due to the fact that there are infinitely many possible shot positions the matrix $\boldsymbol{X}$ containing all

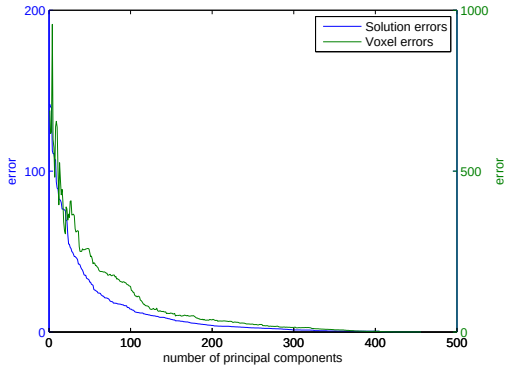


Figure 5.8: The plot shows the connection between the largest residuals of the voxel doses and the largest residuals of the solutions for different values of k.

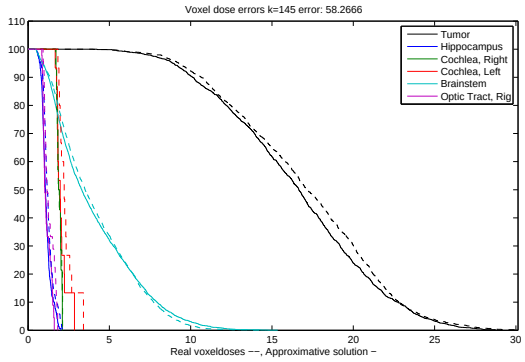


Figure 5.9: The DVH plot of the plan with the largest voxel residual, for $k = 145$.

solutions can become quite large.

To guarantee positivity of the beam times $\boldsymbol{x}_k$ new constraints have to be added on the form $\boldsymbol{W}_k^T \boldsymbol{y}_k \geq 0$. Since this affects the computational time the overall performance of this modified problem has to be studied further.

Another possible dimensionality reduction would be based on a smaller number of shots. The effects however on the optimal solutions from a reduced number of shots and their fixed positions must also be investigated, since there is no way to add or adjust shots in the algorithm.

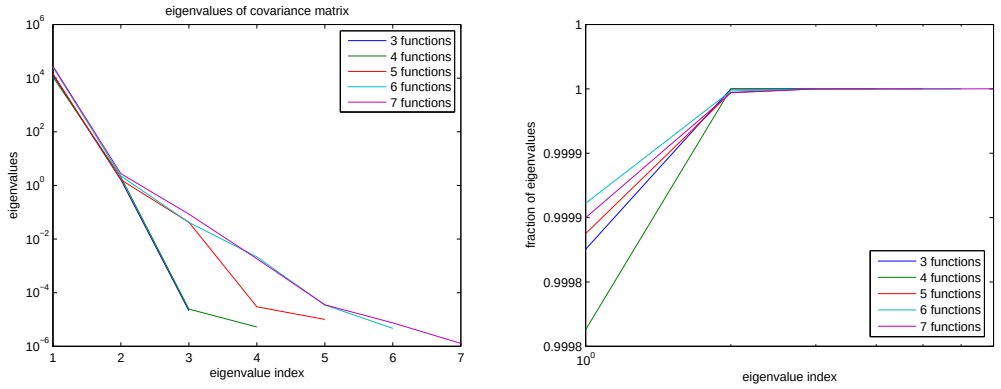
In the pursuit of lower computational times one has to keep in mind that the main goal is finding a good treatment plan and that low computational times have no purpose if the plans found are way worse than plans found by the original problem.

5.3.3 Dimensionality of objective functions

Since the computational times highly depend on the number of functions, there is a strong reason to try to keep it down. To see if there were functions that seemed

to have the same aim in the optimization, the dimensionality of the Pareto optimal plans was studied.

The eigenvalues of the covariance matrix for the Pareto optimal points are plotted in Figure 5.10a and the fractions $\frac{\sum_{i=1}^{k} \lambda_i}{\sum \lambda}$ for increasing k are plotted in Figure 5.10b.



(a) Plots of the eigenvalues of the Pareto optimal points in the different test cases.

(b) Each plot shows how the fraction $\frac{\sum_{i=1}^{k} \lambda_i}{\sum \lambda}$ increases for each additional principal component.

Figure 5.10: Plots of the eigenvalues of the Pareto optimal points in the different cases and the fraction of 'information' that is kept by an increasing number of eigenvalues.

Figure 5.10a suggests that most of the 'information' can be represented in three dimensions, for all of the test cases. Not too surprising since for example OARs lying in the same direction from the tumour would have similar preferences when it comes to radiation directions. Also the function controlling treatment time probably has the same goal as the OAR functions, namely no radiation at all. However the time function was very convenient in the navigation tool to disregard plans with too high treatment time and also by being able to adjust it directly.

Another way to perform the PCA, that was not done in this thesis, would be to consider all points from all test cases at once. For this the dimensionality of the points would have to be made equal. This can be achieved by evaluating the functions for the organs which were not considered in each test case to get these coordinate values.

6 Discussion and future work

The aim of this project was to provide a Graphical User Interface to help the planner find a good treatment plan in real-time and not having to wait for calculations when exploring different plans. This goal is achieved by the navigation tool that was created, though more testing needs to be done. An issue which has not been focused on in this project is usability, even though it is a very important aspect.

6.1 Conclusions

The results of the studied computational times suggest that one should try to keep the number of objectives in the optimization to a minimum, since the number of objectives itself adds a lot of computational time. The obvious way is by considering a minimum number of OARs. But another way could be to consider several OARs as one in the optimization. The DVH could still be calculated for each of the organs separately for visualization purposes.

Since the number of voxels affects the optimization time, one might want to consider the possibility of choosing only critical parts of these organs or larger voxel sizes for these organs, to be able to take large organs into account in the optimization.

Sensitivity of the dual algorithm concerning some parameters is further discussed together with other observations regarding the performance of the dual algorithm.

6.2 Parameter issues and further investigation

Numerical experiments showed that it seemed to matter where the upper bounds to create the primal polytope where added. This has to be investigated further, since the dependence was not clear. A possible explanation is the polytopes created and geometrical problems that might occur when determining the normals of the facets and checking visibility of the new dual point.

To ensure that the added upper bounds end up being larger than all plans during the calculations, a better way of approximating the nadir point than using the pay-off table should be looked into. In this project the bounds were found by trial and error.

The objective functions were quite simple, so investigating other objective functions is a matter that could to be investigated further. Related to this is how function parameters, which sometimes may be difficult to calibrate, affect the Pareto surface, so that the surface could be adjusted instead of redetermined for a new set of parameters. The idea of the Pareto surface and navigation is avoiding these ad hoc methods of parameter tweaking in the optimization and might be lost if there are too many uncertain parameters in the functions.

6.3 Future work on the Graphical User Interface

Limited amount of time for this project did not allow to add more useful features to the Graphical User Interface (GUI). For example a save button, for saving 'best so far'-solutions, and an undo button to undo moves resulting in undesired plans. Using the real medical images as background for the organ and dose plots also needs to be added for better visualization. Further, more testing of the navigation tool is needed to guarantee proper results.

A Appendix

Algorithm A.1 Vertex enumeration

for all starting points $\boldsymbol{z}^{(i)}$ **do**
  $\boldsymbol{z}_{trans}^{(i)} \leftarrow \boldsymbol{z}^{(i)}$
end for
for all hyperplanes $(-\boldsymbol{w}^{(i)})^T \boldsymbol{z}_{trans}^{(i)} = b^{(i)}$ **do**
  $\boldsymbol{d}^{(i)} \leftarrow \frac{-\boldsymbol{w}^{(i)}}{b^{(i)}}$
end for
for all facets in dual convex hull Conv($\{\boldsymbol{d}^{(i)}\}$) **do**
  find normal $\boldsymbol{n}_{dual}^{(j)}$
  find dual point $\boldsymbol{d}^{(j)}$ on facet $\boldsymbol{n}_{dual}^{(j)}$
  $b_{dual}^{(j)} = (\boldsymbol{n}_{dual}^{(j)})^T \boldsymbol{d}^{(j)}$
  $\boldsymbol{\nu}_{trans}^{(j)} \leftarrow \frac{\boldsymbol{n}_{dual}^{(j)}}{b_{dual}^{(j)}}$
end for
for all translated outer vertices $\boldsymbol{\nu}_{trans}^{(j)}$ **do**
  $\boldsymbol{\nu}^{(j)} \leftarrow \boldsymbol{\nu}_{trans}^{(j)}$
  if $\boldsymbol{\nu}^{(j)} \notin \boldsymbol{z}_{aug}^{(i)}$ **then**
    keep $\boldsymbol{\nu}^{(j)}$
  end if
end for

Algorithm A.2 Find new vertices

```
dualize z^new to find d^new
facet ← facet corresponding to ν_max
function FIND HORIZON(facet)
    for all neighbouring facets do
        if neighboring facet visible and not visited then
            mark facet as visited
            FIND HORIZON(neighbour)
        else
            save ridge between facet and neighbor
            save neighbour
        end if
    end for
end function
for all saved ridges do
    create new facet by connecting to d^new
    dualize new facet to find ν_new
end for
for all new facets do
    add adjacency to saved neighbours
end for
remove all visible facets
```

A.1 Test Cases

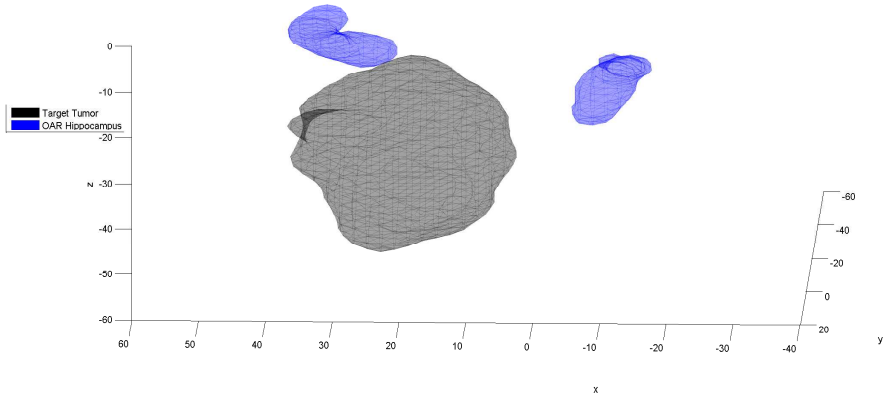


Figure A.1: Tumour and organs in test case 1

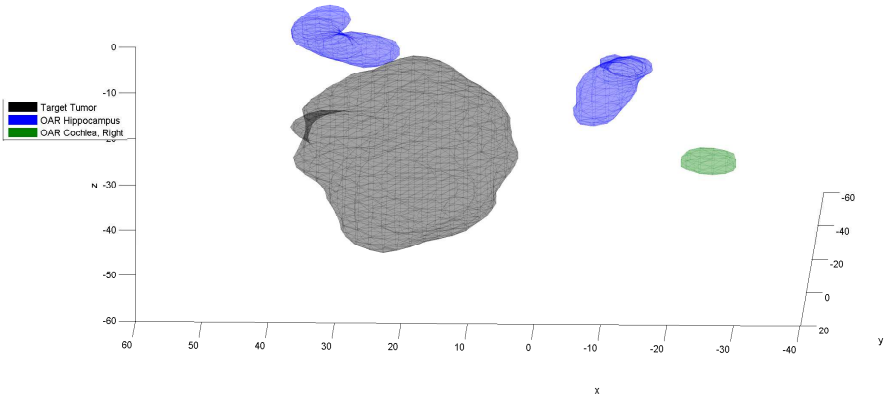


Figure A.2: Tumour and organs in test case 2

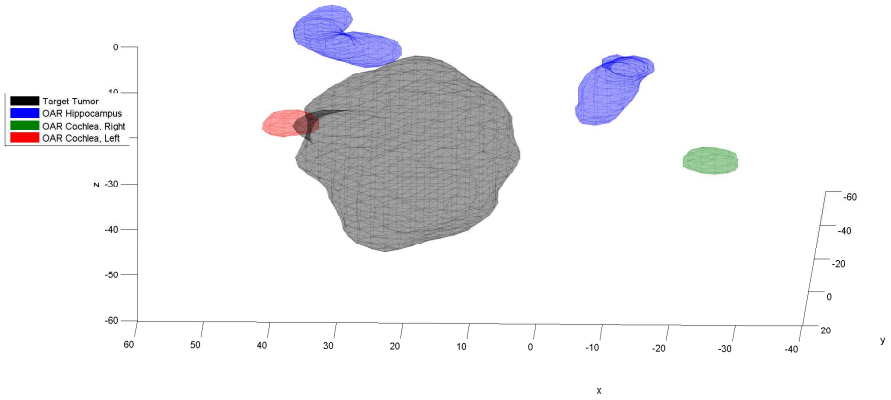


Figure A.3: Tumour and organs in test case 3

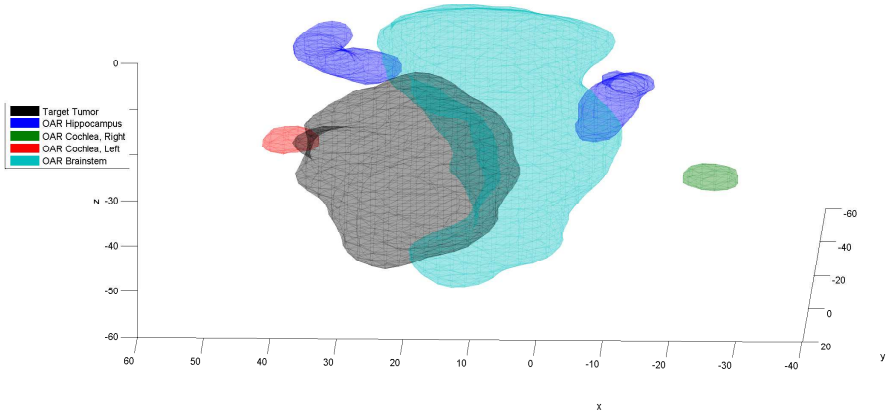


Figure A.4: Tumour and organs in test case 4

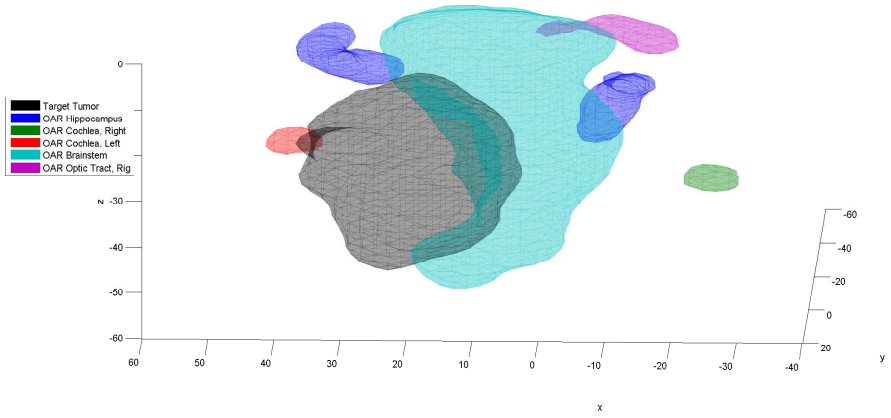


Figure A.5: Tumour and organs in test case 5

Bibliography

[1] White paper: Inverse planning in leksell gammaplan$^{(\textregistered)}$ 10. Technical Report Article no. 018880.02, Elekta, September 2011.

[2] Rasmus Bokrantz and Anders Forsgren. An Algorithm for Approximating Convex Pareto Surfaces Based on Dual Techniques. *INFORMS J. on Computing*, 25(2):377–393, April 2013.

[3] Jacek Gondzio. Multiple centrality corrections in a primal-dual method for linear programming. *Computational Optimization and Applications*, 6(2):137–156, 1996.

[4] Nick Gould and Philippe L. Toint. Preprocessing for quadratic programming. 100(1, Ser. B):95–132, 2004.

[5] Pekka Korhonen, Seppo Salo, and Ralph E. Steuer. A heuristic for estimating nadir criterion values in multiple objective linear programming. *Operations Research*, 45(5):751–757, 1997.

[6] Sanjay Mehrotra. On the implementation of a primal-dual interior point method. *SIAM Journal on Optimization*, 2(4):575–601, 1992.

[7] K. Miettinen. *Nonlinear Multiobjective Optimization*. International series in operations research & management science. Kluwer Academic Publishers, 1999.

[8] M Monz, K H Küfer, T R Bortfeld, and C Thieke. Pareto navigation—algorithmic foundation of interactive multi-criteria IMRT planning. *Physics in Medicine and Biology*, 53(4):985, 2008.

[9] Y. Zhang. Solving large-scale linear programs by interior-point methods under the MATLAB environment. Technical Report Technical Report TR96-01, Department of Mathematics and Statistics, University of Maryland, Baltimore County, Baltimore, MD, July 1995.

TRITA-MAT-E 2014:25
ISRN-KTH/MAT/E—14/25-SE

www.kth.se