Fachhochschule Regensburg

Mathematik

D I P L O M A R B E I T

Studie zur Entwicklung einer inversen Planung mit direkter Aperturoptimierung zur Verbesserung der Strahlentherapie von Tumoren

Vorgelegt von:
Marco Alt

Studiengang: Mathematik
Matrikelnummer: 2217467

Regensburg, den 28.03.2008

Gutachter (intern):
Prof. Dr.H.-J. Wagner
Prof. Dr. R. Hornung

Gutachter (extern):
Prof. Dr. L. Bogner

Inhaltsverzeichnis

Zusammenfassung

Aufgrund der sich stetig weiterentwickelnden Computertechnologien wird die intensitätsmodulierte Strahlentherapie (IMRT) für medizinische Anwendungen zunehmend wichtiger. Die Verwendung intensitätsmodulierter Felder ermöglicht eine effektive und präzise Bestrahlung des Zielvolumens bei gleichzeitiger Schonung gesunden Normalgewebes. Die Bestimmung optimaler Feldmodulationen und die Verifikation der daraus resultierenden Dosisverteilung gestaltet sich wegen der Notwendigkeit nichtlinearer Optimierungsverfahren noch immer als rechenintensiv und qualitativ verbesserungsbedürftig.

In der vorliegenden Arbeit wurde das von Shepard et. al. [18] im Jahre 2002 erstmals vorgestellte Prinzip der direkten Aperturoptimierung (DAO) mit stochastischen Optimierungsmethoden erneut aufgegriffen. Als Methode zur Berechnung der Dosisverteilungen wird jedoch die am Universtitätsklinikum Regensburg von L. Bogner et. al. [2] entwickelte IK-Doseengine verwendet. Im Rahmen einer Studie sollte ein auf Simulated Annealing (SA) basierender Algorithmus entwickelt und untersucht werden, der DAO mit der Monte-Carlo präzisen IK-Technik verbindet.

Die Verifikation der Leistungsfähigkeit der DAO Methode erfolgt durch den Vergleich mit IKO [2] optimierten IMRT Bestrahlungsplänen. Es werden dazu zwei Prostatafälle und ein Quasimodo Phantom untersucht.

Die Ergebnisse zeigen, dass DAO die herkömmlichen IKO Fluenzoptimierungsmethoden, die weitere Segmentierungs- und Reoptimierungsmaßnahmen erfordern, im Allgemeinen qualitativ übertrifft. Es wird eine bessere Homogenität im PTV bei mindestens gleichwertiger Risikoorganschonung erreicht.

Hohe Laufzeiten und Probleme bei konvex geformten Zielstrukturen erfordern noch

zusätzliche Studien zur Verbesserung und Weiterentwicklung des DAO Verfahrens. Dennoch sind bereits jetzt praktische Anwendungen möglich. Desweiteren kann wegen der einfachen Adaption zusätzlicher MLC-Constraints in DAO die IK-Technik auf sämtliche IMRT Bestrahlungsgeräte übertragen werden.

1 Einleitung

Durch die ständige Weiterentwicklung und Erforschung neuer Technologien verbessert sich auch die Medizintechnik laufend. Vor allem rasante Fortschritte in der Computertechnik ermöglichen die Anwendung immer komplexer und präziser werdender strahlentherapeutischer Maßnahmen. Neben den Photonenstrahlen kommen hier auch einige andere Strahlungsarten zum Einsatz, deren spezifische Eigenschaften für verschiedenste therapeutische Anwendungen ausgenutzt werden können. Als Beispiele seien hier etwa Protonen- Neutronen- und schwere Ionenstrahlung genannt. Solche Teilchenstrahlungen bieten Vorteile gegenüber den Photonen, da sie bessere physikalische Dosisverteilungen ermöglichen. Die notwendigen Apparaturen sind in ihrer Technik jedoch weitaus aufwändiger und deshalb sehr teuer [20]. Im klinischen Betrieb werden aus Kostengründen daher überwiegend Photonen-Linacs eingesetzt. Auch wegen der sehr kompakten Bauweise solcher Linearbeschleuniger erfolgen IMRT Bestrahlungen primär durch Photonen.

1.1 Grundlagen der intensitätsmodulierten Strahlentherapie

Ziel der intensitätsmodulierten Strahlentherapie (IMRT) ist die Dosiseskalation im Zielvolumen bei gleichzeitiger Schonung gesunden Normalgewebes [4]. Um das zu erreichen ist eine komplexe Dosisverteilung erforderlich, die durch Überlagerung mehrerer intensitätsmodulierter Teilstrahlen erzeugt werden kann. Diese Intensitätsmodulationen bezeichnet man auch als *Fluenzverteilungen* und deren Gesamtheit bestimmt die vollständige Dosisverteilung eindeutig. In Abbildung 1.1 (links) wird dieses Konzept auf schematische Weise dargestellt und zeigt skizzenhaft die mit zwei inhomogenen Strahlen durch Überlagerung erzeugte Dosisdeponation im Schnittbereich. Das rechte Bild zeigt

die praktische Anwendung dieses Prinzips für Photonenstrahlen. Zu sehen ist ein Schnittbild aus der optimierten Dosisverteilung eines realen Prostata Bestrahlungsplanes mit sechs Feldern (Einstrahlrichtungen). Um in der Praxis die verschiedenen Einstrahlrich-

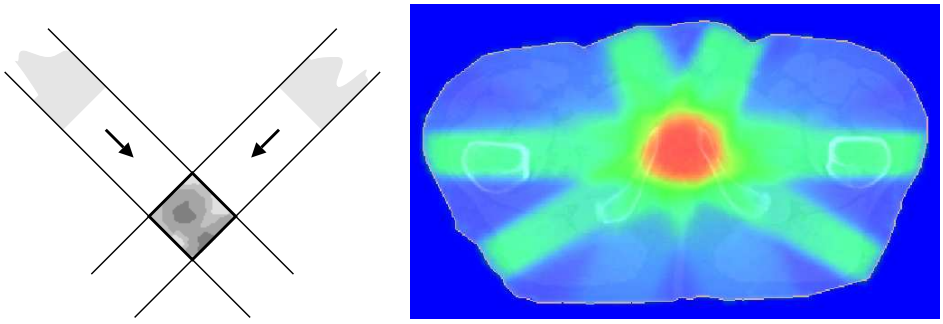

Abbildung 1.1: links: Schematische Darstellung des Überlagerungsprinzips; rechts: Überlagerung von sechs Photonenstrahlen an einem realen Prostatafall

tungen zu realisieren kann der gesamte Linearbeschleuniger um 360° koplanar um den liegenden Patienten rotieren (Abbildung 1.2).

Die Form eines abgestrahlten Photonenstrahls wird am Bestrahlungsgerät durch die *Apertur* (lat. Apertura: Öffnung) des Strahlerkopfes bestimmt. Mit in X-Richtung mechanisch verfahrbaren *Leafs*, die jeweils als Paare angeordnet sind und zusammen den Multi Leaf Collminator (MLC) des Beschleunigers bilden, können solche Aperturen realisiert werden. Zwei Y-Blenden sorgen für eine Reduzierung der Transmission durch geschlossene Leafpaare und können den Photonenstrahl in Y-Richtung weiter einschränken. Abbildung 1.3 zeigt den prinzipiellen Aufbau eines Strahlerkopfes und die Lage der Leafs (G) und Y-Blenden (F; Bewegungsrichtung senkrecht zur Papierebene) direkt vor der Austrittsöffnung. Dieses Grundprinzip der Aperturerzeugung wird in Abbildung 1.4 weiter verdeutlicht. Durch Vorgabe der Zeitdauer, für die durch eine solche (feste) Apertur bestrahlt wird, lässt sich die Gesamtintensität des abgegebenen Photonenrahls bestimmen. Da Linearbeschleuniger nur homogene Strahlung abgeben, können für lokale Bereiche keine Intensitätsmodulationen erzeugt werden. Die Strahlungsintensität ist über der gesamten Apertur immer gleich groß. Erst durch die aus einer Richtung erfolgende Abstrahlung einer Reihe verschiedener Aperturen, mit individuell festgelegten Intensi-

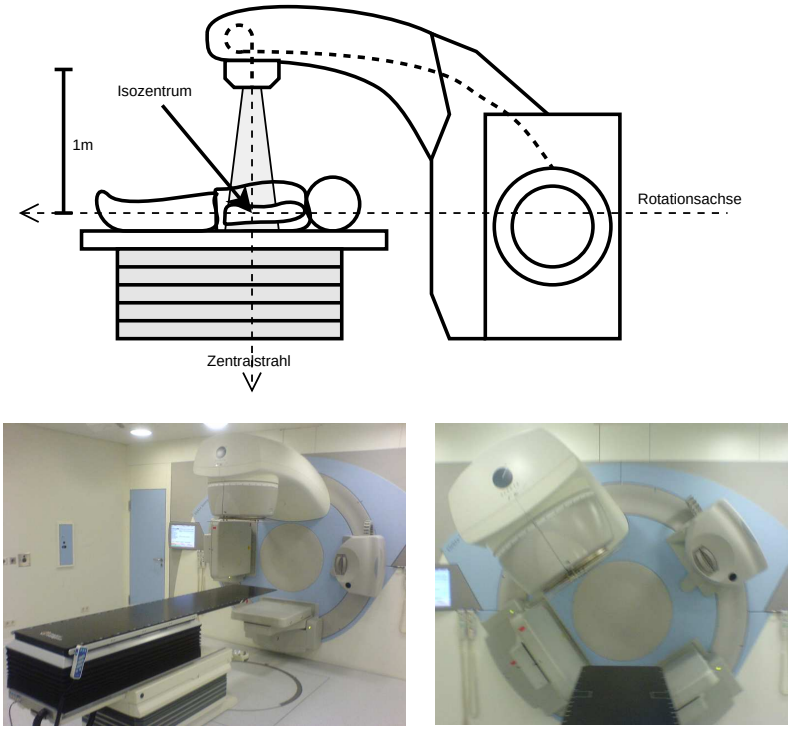

Abbildung 1.2: oben: Schematische Darstellung einer IMRT Bestrahlung; unten: Der *Synergy* Beschleuniger von Elekta des Universitäts-Klinikums Regensburg

täten, lässt sich eine komplexe Fluenzverteilung durch Überlagerungseffekte nachbilden. Man bezeichnet einzelne Aperturen als *Segmente* und die zugehörige Bestrahlungszeit als *Segmentgewicht*. Eine Veranschaulichung dieser Methode zur Erzeugung von Intensitätsmodulationen wird in Abbildung 1.5 gezeigt. Die wichtigste Konsequenz daraus ist, dass beliebige Fluenzverteilungen in der Praxis nicht ohne Weiteres erzeugt werden können und durch Segmentüberlagerungen angenähert werden müssen, was einer Diskretisierung gleichkommt.

Die Verifikation der Qualität eines IMRT Plans erfolgt durch die Erstellung des Dosis-Volumen Histogramms (DVH) anhand der Dosisverteilung und dessen anschließende Überprüfung und Bewertung durch zuvor festgelegte Vorgaben (DV-Constraints). DV-Constraints müssen für jede betrachtete Volumenstruktur (ROI, Region Of Interest)

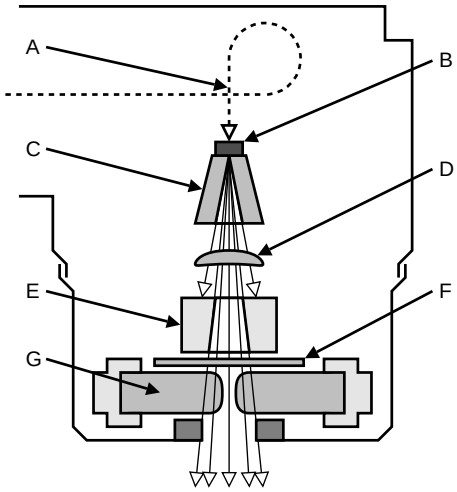

Abbildung 1.3: Prinzipieller Aufbau des Strahlerkopfes eines Linearbeschleunigers: A) Elektronenstrahl, B) Target, C) Primärkollminator, D) Ausgleichsfilter, E) Sekundärkollminator, F) Y-Blenden, G) Leafs

definiert werden (Abbildung 1.6), also in jedem Fall für das PTV (Planning Target Volume; Zielvolumen) und die OARs (Organ At Risk; Risikoorgan) [4]. Zusätzlich werden oft Hilfskonturen (UT, Unspecified Tissue) definiert, die als künstliche OARs fungieren um Optimierungsalgorithmen auf bestimmte Weise zu beeinflussen. Üblicherweise finden Hilfskonturen als Randbereich (Margin) um das PTV Anwendung um außerhalb dessen einen steileren Abfall des Dosisgradienten zu erreichen [9].

1.2 Motivation und Zielsetzung

Herkömmliche IMRT Optimierungsverfahren verfolgen das Prinzip der inversen IMRT Planung. Die Problemstellung ist dabei die Bestimmung der Intensitätsmodulationen anhand einer vorgegebenen Dosisverteilung [4]. Auf diese Weise ist es theoretisch möglich die zu einer optimalen Dosisverteilung gehörenden Fluenzverteilungen direkt zu ermitteln. Wegen der hohen Komplexität dieses Optimierungsproblems müssen jedoch nichtlineare Optimiersungsalgorithmen verwendet werden, die eine optimale Lösung im Allgemeinen nicht garantieren können. Das am Universitätsklinikum Regensburg von L.

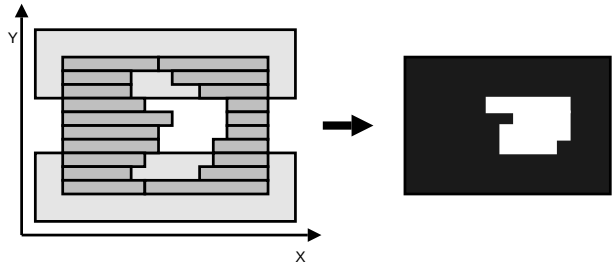

Abbildung 1.4: Durch jede bestimmte anordnung der einzelnen Leafs (dunkelgrau) des MLCs und der Y-blenden (hellgrau), erreicht man eine Modulation der Form des Photonenstrahls (rechts)

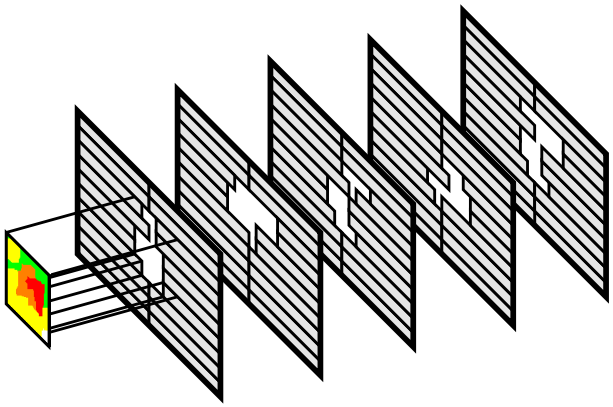

Abbildung 1.5: Komplexe Fluenzverteilungen werden durch die Überlagerung einzelner Segmente nachgebildet

Bogner et. al. [2] entwickelte Bestrahlungsplanungssystem IKO (Inverse Kernel Optimierung) arbeitet ebenfalls nach diesem Schema, verwendet jedoch einen Monte-Carlo genauen Dosisalgorithmus (IK-Doseengine [9]).

Aufgrund der Funktionsweise von Linearbeschleunigern müssen die ermittelten Fluenzverteilungen jedoch erst in Segmente übersetzt und deren jeweilige Gewichte bestimmt werden (Segmentierung) um die praktische Bestrahlung zu ermöglichen. Zu diesem Zweck wird die kommerzielle Software ImFast® (Siemens) eingesetzt. Weil dabei aus praktischen Gründen und vor allem in Hinsicht auf die Behandlungsdauer nur endlich viele Segmente erzeugt werden, entstehen bei der Segmentierung immer Qualitätsverlu-

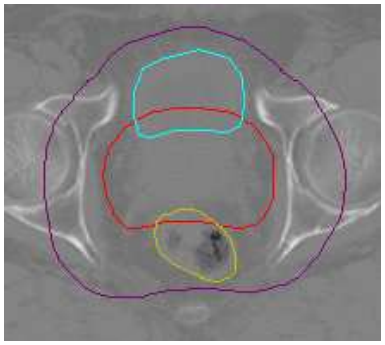

Abbildung 1.6: Auschnitt aus einem CT Schnittbild eines Prostata Patienten; eingezeichnet sind das PTV (rot), die OARs (Blase: cyan; Rektum: ocker) und eine Hilfskontur (UT-Margin: violett)

ste. Durch eine nachträgliche Anpassung der Segmentgewichte durch IKO [16] kann der IMRT Plan wieder weitgehend auf das Niveau der Fluenzoptimierung gebracht werden (Reoptimierung). Hinzu kommt allerdings das Problem, dass für manche Linearbeschleuniger gar keine Segmentierer existieren und eine IKO Planung für diese Geräte deshalb nicht möglich ist. Dazu gehört auch der *Synergy* Beschleuniger von Elekta.

Zur Lösung dieses Problems wurde das von D.M. Shepard et. al [18] im Jahre 2002 erstmals vorgestellte Konzept der direkten Aperturoptimierung (DAO) aufgegriffen. Dabei handelt es sich um ein stochastisches Optimierungsverfahren auf Basis eines Simulated Annealing (SA) Algorithmus, das als Ergebnis eine bereits zur Bestrahlung geeignete Aperturkonfiguration liefert. Mit dieser Technik lassen sich aufwändige Segmentierungs- und Reoptimierungsmaßnahmen umgehen.

In der vorliegenden Arbeit wird die Umsetzung eines im Rahmen einer Studie entwickelten DAO Optimierungsalgorithmus beschrieben. Innovativ ist dabei die Verwendung der IK-Doseengine, welche sich besonders gut für ein DAO Verfahren eignet. Anhand zweier realer Prostatafälle und einem Quasimodo Phantom [8] wird die Planqualität mit der von IKO verglichen.

Da der DAO Algorithmus in naher Zukunft in das IKO System integriert werden soll, wurden weitgehend identische Datenmodelle verwendet oder geringfügig adaptiert. Es

erfolgt daher zunächst eine Einführung in die Grundlagen von IKO, bevor im Anschluss näher auf Details der DAO Optimierung eingegangen wird. Vor der Beschreibung der Funktionsweise der verwendeten Variante eines Simulated Annealing Algorithmus erfolgt ebenso ein kleiner Exkurs zur Beschreibung des grundlegenden Prinzips eines solchen stochastischen Optimierungsverfahrens. Die Arbeit wird mit der Darlegung und Diskussion einiger Ergebnisse abgeschlossen. Im Anhang befindet sich ein weiterer Abschnitt, in dem der Vergleich einiger Zufallsgeneratoren bezüglich deren Eignung für die praktische Verwendung in DAO behandelt wird.

2 Material & Methoden

2.1 Verwendetes Computersystem und Ausstattung

Alle Teile dieser Arbeit wurden an einem Klinikum Standard PC mit folgenden Spezifikationen und Entwicklungstools durchgeführt:

Prozessor:	Pentium 4 (2.8 GHz)
Arbeitsspeicher:	3 GB
Festplatte:	500 GB
Betriebssystem:	Debian - Etch

MatLab:	Version 7.5.0.338 (R2007b)
C++ Compiler:	g++-2.95
ImFast®:	Version 1.05

2.2 Inverse Kernel Optimierunge (IKO)

IKO ist ein im Rahmen mehrerer Studien [2, 3, 9, 16] am Universitätsklinikum Regensburg entwickeltes inverses Planungssystem für IMRT Bestrahlungen mit dem Schwerpunkt auf Planoptimierung und Validierung. Die Durchführung der Fluenzoptimierung basiert auf drei wesentlichen Komponenten:

- Einem Algorithmus zur Dosisberechnung
- Einer Zielfunktion zur Validierung von Dosisverteilungen

- Einem inversen Optimierungsalgorithmus

Die Funktionsweise und mathematische Modellierung dieser Komponenten wird in den folgenden Unterabschnitten erläutert.

2.2.1 Dosisberechnung mit der IK-Doseengine

Die Berechnung von Dosisverteilungen wird als diskretisiertes Problem betrachtet, indem das Patientenvolumen durch eine 3D-Matrix (Volumenmatrix) repräsentiert wird. Einzelne Volumenelemente werden als *Voxel* bezeichnet und identisch dimensioniert. Die Volumenmatrix wird anhand der CT (Computer Tomographie) Aufnahme des Patienten erstellt, die aus äquidistant verteilten Schichtbildern besteht (Z-Richtung). Durch die Vorgabe einer Auflösung für die XY-Ebene wird die Volumenmatrix eindeutig dimensioniert (Abbildung 2.1). Aus den Daten der CT Aufnahme werden im Weiteren

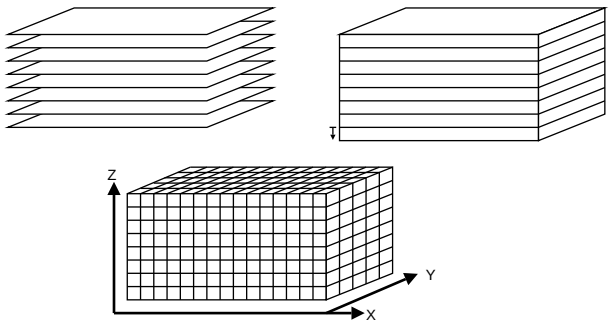

Abbildung 2.1: oben links: Einzelschichten der CT Aufnahme; oben rechts: Umwandlung in einen Quader; unten: Unterteilung in Voxel

Informationen über die Beschaffenheit des Gewebes gewonnen und den Voxeln entsprechende Dichtewerte zugeordnet. Der XVMC (X-Ray Voxel Monte Carlo) Algorithmus von M. Fippel [6] ermöglicht eine Monte-Carlo (MC) genaue Simulation von Photonen und deren Wechselwirkungen im Gewebe anhand einer solchen Dichtematrix. Dadurch wird eine genaue Bestimmung der aus beliebigen Aperturkonfigurationen folgenden Dosisverteilungen möglich. Der ursprüngliche Algorithmus beschränkt sich jedoch auf die Bestimmung der Dosisverteilung, so dass die Information, aus welchem Teilgebiet der

Strahlerkopföffnung ein einzelnes Photon stammt und in welchen Voxeln es für Dosiseskalationen gesorgt hat, dabei verloren geht.

Die Besonderheit der IK-Doseengine ist nun gerade ein modifizierter XVMC Algorithmus, der jeder Dosisdeponation durch simulierte Teilchen einen Ursprungsort innerhalb der Beschleunigeröffnung zuweist [2]. Diese wird durch die Strahlmodulationsebene (BMP, Beam Modulation Plane) in Form einer 2D-Matrix beschrieben, die den austretenden Photonenstrahl schachbrettartig in einzelne *Beamlets* (Teilstrahlen) unterteilt (Diskretisierung). Die Elemente der BMP werden als *Bixel* bezeichnet und repräsentieren das entsprechende Teilgebiet der Beschleunigeröffnung. Hier ist es zweckmäßig die Bixelgrößen in Y-Richtung an den zu verwendenden MLC-Typ anzupassen, da die Leafbreite ausschlaggebend für die tatsächlich realisierbaren Beamlets ist. In X-Richtung muss ein Kompromiss zwischen Rechenzeit und Genauigkeit gefunden werden. Für den *PRIMUS* Beschleuniger (Siemens) werden quadratische Bixel verwendet. Abbildung 2.2 zeigt eine so angepasste BMP und verdeutlich das Prinzip.

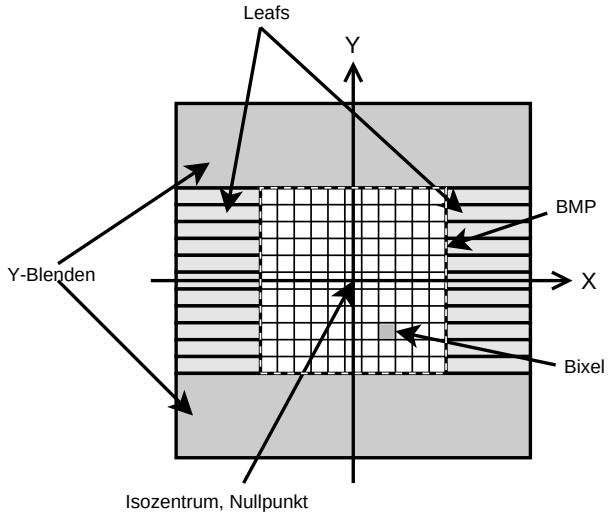

Abbildung 2.2: Die BMP wird schachbrettartig in Bixel unterteilt und an die Leafbreiten angepasst

Mit der IK-Doseengine kann somit nun für jeden Bixel ein Datensatz erzeugt werden, der als *inverser Kernel* (IK) bezeichnet wird und eine Dosismatrix repräsentiert, die nur

diejenigen Dosisbeiträge des durch den Bixel definierten Beamlets enthält. Die Hinterlegung aller inversen Kernels eines IMRT Plans erfolgt im Sinne von Sparse-Matrizen listenförmig in einer Kernel-Datei (Abbildung 2.3). Jeder Datensatz besteht aus einer

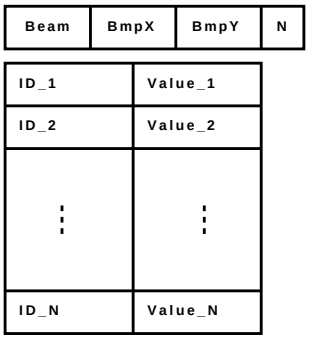

Abbildung 2.3: Aufbau eines Kernel-Datensatzes

Header-Zeile und einem *Body*-Teil. Im Header sind die zur eindeutigen Identifikation eines Bixel notwendigen Informationen hinterlegt: die Einstrahlrichtung (Beam) und die Koordinate in der BMP-Matrix (BmpX|BmpY). Zusätzlich enthält der Header die Größe (N) des folgenden Body-Datensatzes. Darin sind alle Voxel nach ihrer ID (Kennzahl zur eindeutigen Ortszuweisung in der Volumenmatrix) aufsteigend sortiert und mit dem Wert (Value) des jeweiligen Dosisbeitrags gepaart [9].

Durch die Zuordnung von Gewichtungsfaktoren ω_j zu jedem Bixel j lässt sich eine Intentistätsmodulation mit BMP Auflösung darstellen. Die resultierende Dosis D_i in einem Voxel i berechnet sich dann nach (2.1), wobei $IK_{i,j}$ der ungewichtete Dosisbeitrag des Bixels j im Voxel i ist. Die Menge $Bixel$ enthält alle Indexnummern (IDs), die jedem existierenden Bixel eindeutig zugeordnet sind.

$$D_i = \sum_{j \in Bixel} \omega_j \cdot IK_{i,j} \tag{2.1}$$

Führt man diese Berechnungsmethode für alle Voxel genau einmal durch, so erhält man die gesamte Dosisverteilung mit MC-Genauigkeit in Abhängigkeit der Bixelgewichte, ohne eine auwändige XVMC Vorwärtsrechnung durchführen zu müssen [2]. Die Gewichte aller Bixel einer BMP lassen sich in Matrixform zusammenfassen. Pro Einstrahlrichtung

b wird demnach jeweils eine *Fluenzmatrix* F_b definiert, die zur Darstellung einer Fluenzverteilung verwendet wird (Abbildung 2.4). Die IK-Doseengine ist also ein Verfahren

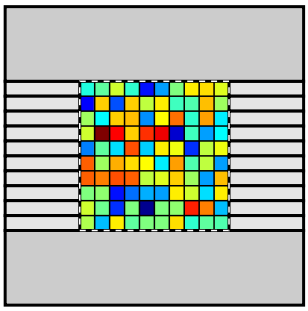

Abbildung 2.4: Schematische Darstellung einer Fluenzmatrix; Die Intensitäten der einzelnen Beamlets ist im Bild durch farbcodierte Bixel dargestellt

zur Berechnung einer diskretisierten Dosisverteilung anhand diskretisierter Fluenzverteilungen unter Anwendung des *Look-Up-Table* Prinzips. Wegen der dazu nur einmalig erforderlichen XVMC Simulation ergibt sich eine enorme Rechenzeitersparnis unter Beibehaltung der MC-Genauigkeit.

2.2.2 Physikalische Zielfunktion

Bei Optimierungsproblemen ist es im Allgemeinen notwendig die zu optimierende Eigenschaft des Systems (hier die Dosisverteilung) anhand eines einzigen Zahlenwertes auszudrücken, der die Güte dieser Eigenschaft repräsentiert. Dieser Zahlenwert muss anhand einer geeigneten Funktion, der *Zielfunktion* (OF, Objective Function), für jeden Systemzustand ermittelt werden können.

Im Falle der Dosisverteilung wird in IKO die Funktion (2.2) als Zielfunktion für ein Minimierungsproblem verwendet, das sich auf die physikalische Dosisdeponation in den einzelnen Voxeln bezieht (physikalische Zielfunktion). Diese setzt sich aus der Summe spezifischer Zielfunktionen der ROIs zusammen, also für das PTV (2.3) und alle N_{OAR} OARs (2.4). Es erfolgt jeweils eine Normierung durch die Anzahl der Voxel V_{ROI} und die Gewichtung mit einem *Penaltyfaktor* (Bestrafungsfaktor). Diese Teilfunktionen sind im wesentlichen Fehlerquadratsummen für die Dosis D_i (gemäß Gleichung 2.1) über

alle Voxel i bezogen auf die ROI spezifische Solldosis D_0. Im PTV gilt $D_0 = D_0^{PTV} = konstant$, während das D_0 in den OARs von den definierten DV-Constraints und D_i abhängt und dem Wert $D_0^{OAR_j}(D_i) =: D_0^{OAR_j}$ entspricht. Da die Zielfunktion von der Anzahl der Voxel (des PTVs und der OARs) abhängt, folgt unmittelbar die Abhängigkeit der Rechenzeit von der Genauigkeit des Ergebnisses, das eben gerade von der Anzahl der Voxel (bestimmt durch die XY-Auflösung) abhängt. Ein Voxel wird jedoch immer nur dann verwertet, wenn seine Dosis einen DV-Constraint verletzt. Mittels des binären Schalters ${}_{ROI}\Theta_{\mathrm{i}} := {}_{ROI}\Theta(\mathrm{D_i}) \in \{0, 1\}$ lässt sich dies in der Formel berücksichtigen [9].

$$OF := \frac{p^{PTV}}{V_{PTV}} \cdot OF_{PTV} + \sum_{j=1}^{N_{OAR}} \frac{p^{OAR_j}}{V_{OAR_j}} \cdot OF_{OAR_j} \tag{2.2}$$

$$OF_{PTV} := \sum_{i=1}^{V_{PTV}} (D_i - D_0^{PTV})^2 \cdot {}_{PTV}\Theta_{\mathrm{i}} \tag{2.3}$$

$$OF_{OAR_j} := \sum_{i=1}^{V_{OAR_j}} (D_i - D_0^{OAR_j})^2 \cdot {}_{OAR_j}\Theta_{\mathrm{i}} \tag{2.4}$$

2.2.3 IMRT Planung mit IKO

Die Patientenverwaltung und Planbearbeitung durch medizinisches Fachpersonal, sowie die Übertragung fertig optimierter Pläne auf den Beschleuniger zur realen Bestrahlung muss bei IKO über externe Programme erfolgen. Der Kreislauf einer IMRT Planung mit IKO ist in Abbildung 2.5 dargestellt.

Die Neuerstellung von Bestrahlungsplänen erfolgt hier am Universitätsklinikum anhand des kommerziellen Planungssystems OTP® (Oncentra Master Plan, Nucletron) und werden im DICOM[1]-Format nach IKO exportiert. Solche Rohpläne enthalten bereits die vom Arzt eingezeichneten ROIs und fest definierte Einstrahlrichtungen. Beim Import werden sämtliche Daten (z.B. die Dichtematrix für XVMC, ROI Definitionsdateien und VMC Plandateien) für die weitere Verwendung erstellt.

Im nächsten Schritt müssen die inversen Kernels (unter Vorgabe der gewünschten XY-Auflösung der Volumenmatrix) mit dem adaptierten XVMC Algorithmus erstellt werden,

[1]DICOM: Digital Imaging and Communications in Medicine

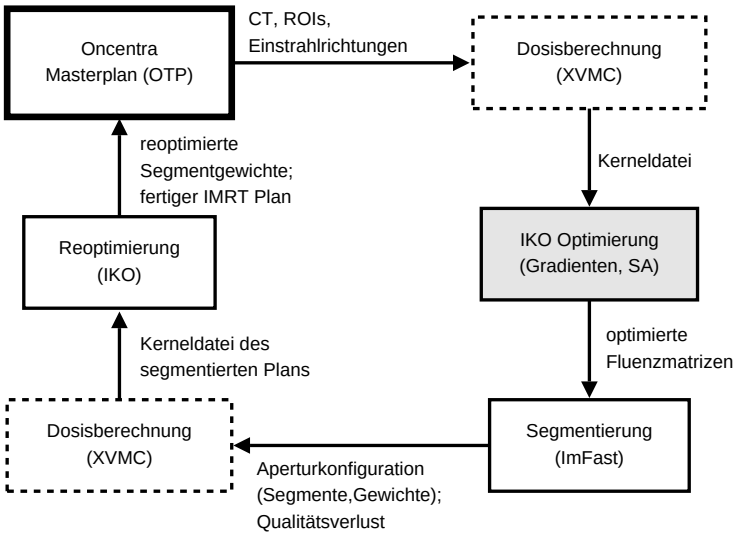

Abbildung 2.5: Flussdiagramm der IMRT Planung mit IKO

damit im Folgenden die Fluenzoptimierung ausgeführt werden kann. Bei der Berechnung wird für jede Einstrahlrichtung ein rechteckiger Ausschnitt der BMP verwendet, aus dessen entsprechender Blickrichtung das PTV vollständig enthalten ist (Open-Field Rechnung). Jeder dieser Bereiche bedeutet eine zu optimierende Fluenzmatrix.

Um dem Optimierungsalgorithmus mitzuteilen in welchen Voxeln welche Bedingungen herrschen sollen, müssen DV-Constraints und Penaltyfaktoren für jedes ROI definiert werden, sofern nicht bereits vorgegeben. Insbesondere muss anhand der importierten ROI Definitionen (Polygonzüge) eine ROI-Datei erstellt werden, die Zuordnungen jedes Voxels der Volumenmatrix zu dem entsprechenden ROI enthält. Im Anschluss kann die Fluenzoptimierung mit der IK-Methode erfolgen, wobei der Benutzer zwischen einer Reihe verschiedener Optimierungsalgorithmen wählen kann, die im Laufe der Entwicklung von IKO implementiert wurden. Dazu zählen sowohl deterministische Gradientenverfahren wie Steepest-Descent [19] und CFSQP [12], als auch stochastische Verfahren wie Simulated Annealing [9, 18]. Bei den Gradientenverfahren werden anstelle der binären Schalter Θ in der Zielfunktion geeignete Fehlerfunktionen (errorfunction: $erf(z) := \frac{2}{\sqrt{\pi}} \int_0^z e^{-\tau^2}\, d\tau$) verwendet, um Differenzierbarkeit zu erreichen. Über Gleichung (2.1) kann die Zielfunktion allgemein nur in Abhängigkeit der (N vielen) Bixel angege-

ben werden, wodurch ein nur N-dimensionales Optimierungsproblem entsteht. Gradientenverfahren haben wegen des vergleichsweise geringen Rechenaufwandes gegenüber stochastischen Methoden daher auch bei komplexeren Plänen einen hohen Geschwindigkeitsvorteil. Bei erreichen eines lokalen Minimums können diese jedoch nicht mehr verlassen werden. Deterministische Verfahren bergen daher die Gefahr das globale Optimum zu verfehlen [4].

Nach erfolgreicher Durchführung der Fluenzoptimierung kann das Ergebnis mit IKO direkt validiert werden. Dazu stehen Tools zur DVH Auswertung und Betrachtung der Dosisverteilung zur Verfügung. Ist das Ergebnis zufriedenstellend, wird zur anschließenden Segmentierung das Programm ImFast® (Siemens) verwendet, wofür die optimierten Fluenzdaten exportiert werden müssen (DICOM). Man erhält eine mit dem *PRIMUS* Beschleuniger (Siemens) bestrahlbare Aperturkonfiguration bestehend aus einigen Segmenten und zugeordneten Segmentgewichten. Wegen der endlichen Anzahl an erzeugten Segmenten pro Einstrahlrichtung werden die ursprünglichen Bixelintensitäten der Fluenzverteilung dabei diskretisiert, d.h. die gegebene Fluenzverteilung kann nicht exakt nachgebildet werden [16]. Dadurch entsteht ein Qualitätsverlust, der bei Verwendung weniger Segmente umso erheblicher sein kann. Zur Verifikaiton der tatsächlichen Qualität des segmentierten IMRT Plans wird die Aperturkonfiguration wieder per DICOM-Dateien ins IKO System reimportiert.

Da bei der Bestrahlung echter Aperturen Streueffekte an den Leafkanten auftreten, ist die Nachbildung der Dosisverteilung anhand der inversen Kernels, die durch offene Felder bestimmt wurden, fehlerbehaftet. Nach der Reimportierung des Segmentierten Plans ist mit XVMC daher eine abschließende Dosisberechnung (Vorwärtsrechnung) zur gegebenen Aperturkonfiguration notwendig, bei der eine weitere, zur Reoptimierung benötigte Kernel-Datei erzeugt wird [16]. Hier zeigt sich dann die Qualität des tatsächlichen Ergebnisses, das im Allgemeinen immer etwas schlechter ist als bei der reinen Fluenzoptimierung.

Mit einem in IKO implementierten Reoptimierungsverfahren kann eine letzte Optimierung der Segmentgewichte erfolgen. Dadurch kann der durch die Segmentierung

entstandene Qualitätsverlust zum Teil relativiert werden [16]. Nach der Reoptimierung kann der nun fertig gestellte IMRT Plan zurück nach OTP® exportiert (DICOM) und zur Bestrahlung am Patienten verwendet werden.

2.3 Direct Aperture Optimisation (DAO)

Die Entwicklung, Realisierung und Qualifizierung der direkten Aperturoptimierung unter Verwendung der IK-Doseengine war das Ziel dieser Arbeit. Durch die Einführung von DAO verringert sich der Aufwand einer IKO Optimierung maßgeblich, wie Abbildung 2.6 verdeutlicht.

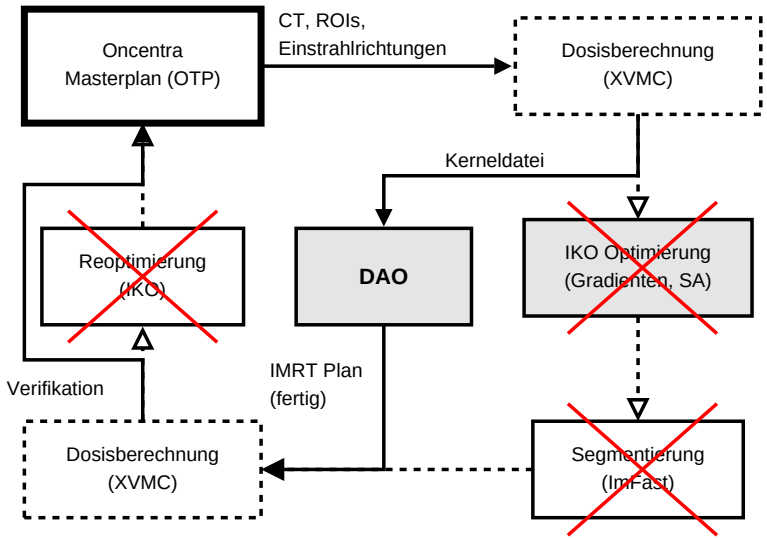

Abbildung 2.6: Flussdiagramm der IMRT Planung mit DAO

Die Umsetzung erfolgte mit der Erstellung einer C++ Konsolenanwendung, die eigenständig funktioniert (Stand-Alone Version). In den folgenden Abschnitten wird auf die Funktionsweise und Struktur des Programms näher eingegangen. Der Aufruf erfolgt in der Linux-Konsole (Terminal) und erwartet einige Übergabeparameter (Listing 2.1), die in den zugehörigen Abschnitten kurz erläutert werden.

```
dao <n_segs> <pat_ID> 'SetupFile' 'DeviceConfig' [<runID>]
```

Listing 2.1: DAO Kommandozeile

Die spitzen Klammern weisen auf numerische Werte und die einfachen Hochkommas auf Strings hin, die als Parametertyp erwartet werden. Eckige Klammern stehen für optionale Argumente.

2.3.1 Modellierung

Weil bei DAO die Aperturkonfiguration an sich, primär also Leafpositionen bestimmt werden, eignet sich das Modell der inversen Kernels dafür besonders gut. Denn durch die Verwendung des BMP Prinzips lassen sich Segmentaperturen vollständig durch binäre 2D-Matrizen, die im Folgenden *Bixelmatrizen* genannt werden, durch das *ein-* und *abschalten* einzelner Bixel darstellen und die zugehörige Dosis mit der IK-Technik einfach berechnen. Durch die feste Vorgabe der Anzahl der Segmente N_{seg}, die bei DAO für jede Einstrahlrichtung gleich gesetzt wird, und unter Verwendung der Segmentgewichte lässt sich so die entsprechende Fluenzmatrix darstellen. Wie auch bei einer durch Segmentierung gewonnenen Aperturkonfiguration lässt sich diese gemäß Gleichung (2.5) berechnen, wobei F_b die Fluenzmatrix, $B_{b,k}$ die Bixelmatrix und $\omega_{b,k}$ das zugeordnete Gewicht des k-ten Segments der Einstrahlrichtung b ist (siehe Abbildung 2.7, vgl. Abb. 1.5).

$$F_b = \sum_{k=1}^{N_{seg}} \omega_{b,k} \cdot B_{b,k} \tag{2.5}$$

Für den Wert $f_b(n, m)$ des Elements (n, m) der Fluenzmatrix F_b und damit für die

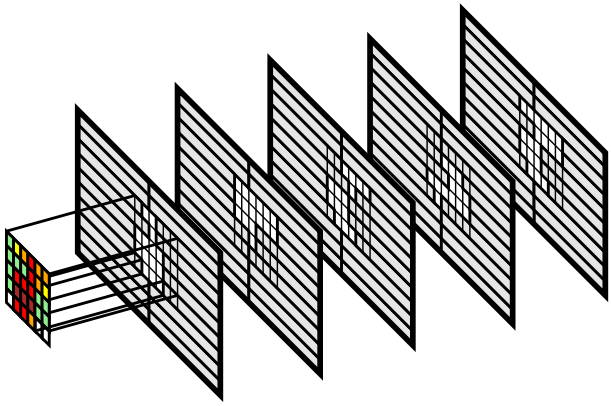

Abbildung 2.7: Die Gesamtheit aller Segmente einer Einstrahlrichtung bildet über (gewichtete) Bixelmatrizen eine eindeutige Fluenzmatrix ab

Intensität des entsprechenden Beamlets gilt (2.6).

$$f_b(n,m) = \sum_{k=1}^{N_{seg}} \omega_{b,k} \cdot \Theta_{b,k}(n,m) \tag{2.6}$$

Dabei ist $\Theta_{b,k}(n,m)$ ein binärer Schalter (2.7) und beschreibt den Zustand des Bixels (n,m) der Bixelmatrix $B_{b,k}$.

$$\Theta_{b,k}(n,m) := \begin{cases} 1 & \text{falls Bixel (n,m) des Segments } b,k \text{ offen} \\ 0 & \text{falls Bixel (n,m) des Segments } b,k \text{ geschlossen} \end{cases} \tag{2.7}$$

Dosisberechnung

Da sowohl $f_b(n,m)$ als auch ω_j die Intensität eines eindeutig (über (b,n,m) bzw. j) definierten Bixels ist, kann zusammen mit (2.6) aus (2.1) durch einfache Umformung die Gleichung (2.8) für die Dosis D_i im Voxel i bestimmt werden.

$$D_i = \sum_{k \in Segment} \left(\omega_k \cdot \sum_{j \in Bixel(k)} \Theta_{k,j} \cdot IK_{i,j} \right) \tag{2.8}$$

Hier ist nun ω_k das Gewicht des (eindeutigen) Segments k und $\Theta_{k,j} \in \{0,1\}$ der Zustand (geschlossen oder offen) des Bixels j dieses Segments. Die Menge $Segment$ enthält wie die Menge $Bixel$ eindeutige IDs für alle Segmente aller Einstrahlrichtungen. In der Menge $Bixel(k)$ sind die zu einem Segment $k \in Segment$ gehörigen Bixel-IDs enthalten, also ist $\forall k \in Segment : Bixel(k) \subset Bixel$ und es gilt:

$$\bigcup_{k \in Segment} Bixel(k) = Bixel$$

Da die gesamte Dosisverteilung nach diesem Modell also nur von genau zwei verschiedenen Parametertypen abhängt (Segmentgewichte ω_k und Bixelschalter $\Theta_{k,j}$), stellt die Gesamtheit all dieser Parameter (Aperturkonfiguration K) die zu optimierenden Größen, also die *Entscheidungsvariablen* dar. Das Minimierungsproblem (2.9) ist damit vollständig beschrieben, wobei Γ die Menge aller zulässigen Aperturkonfigurationen ist.

$$K_{min} = \min\{K \mid OF(K), K \in \Gamma\} \tag{2.9}$$

Zielfunktion

Die in DAO verwendete Zielfunktion ist mit der physikalischen Zielfunktion (2.2) von IKO fast identisch. Es unterscheidet sich nur die Berwertung des Fehlermaßes, welches in den Gleichungen (2.3) und (2.4) durch Quadrate festgelegt, in DAO aber durch Beträge ersetzt wurde. Dadurch wird bei großen Fehlern zwar das Strafverhalten weicher, doch der Rechenaufwand einer Betragsermittlung (`f = abs(dx)`) ist um den Faktor 40 geringer als bei Quadraturen (`f = (dx * dx)`), wie Messungen gezeigt haben. Da die Bestimmung des Zielfunktionswertes primär die Ermittlung der einzelnen Fehler für jeden betrachteten Voxel erfordert, hängt der Rechenaufwand für die Zielfunktionsbestimmung entscheidend von der Art dieser Fehlerbestimmung ab.

2.3.2 Datengrundlage

Vor Beginn der Optimierung müssen alle benötigten Daten in den vorgesehenen Formaten vorliegen. Folgende Dateien müssen dazu eingelesen werden:

- Kernel-Datei des Patienten (enthält die inversen Kernels)
- ROI-Datei des Bestrahlungsplans (enthält die Zuordnungen der Voxel zu den ROIs)
- VMC-Datei des Bestrahlungsplans (enthält die Startkonfiguration der Segmente)
- DAO Konfigurationsdatei (enthält Steuer- und Optimierungsparameter)
- MLC-Constraint-Datei (enthält verschiedene Constraint Definitionen für MLCs)

Die mittels XVMC erstellte Kernel-Datei hat mit bis zu mehreren GigaBytes an Dateigröße den dominiernden Anteil an einzulesendem Datenumfang. Der Einlesevorgang kann dabei bis zu 5 Minuten dauern. Um Datenumfang und Rechenzeit zu sparen kann die Option “BigKernels“ verwendet werden. Dabei wird die Kernel-Datei in zwei Einzeldateien aufgetrennt, von der eine nur die zu den ROIs gehörenden, interessierenden Voxeldaten und die Andere alle übrigen Voxeldaten enthält (Außenkontur). Das ist vor allem bei sehr großen Volumenmatrizen sinnvoll.

2.3.3 Startkonfiguration

Die durch Bixelmatrizen beschriebenen Aperturen können grundsätzlich so eingeschränkt werden, dass nur eine Kontur um das PTV Volumen (aus jeder Blickrichtung) gerade nicht überdeckt wird. Diese Kontur wird in der Regel mit einem Abstand von etwa einem Zentimeter (in der Isozentrumsebene) um das PTV gelegt. Größere Aperturen sind selten sinnvoll, da das Primärziel einer Bestrahlung nur der Tumor, also das PTV ist und umliegendes Gewebe daher nicht durch direkte Strahlung belastet werden sollte (Abbildung 2.8). Die Berechnung der Kernels wird daher nur mit solchen angepassten

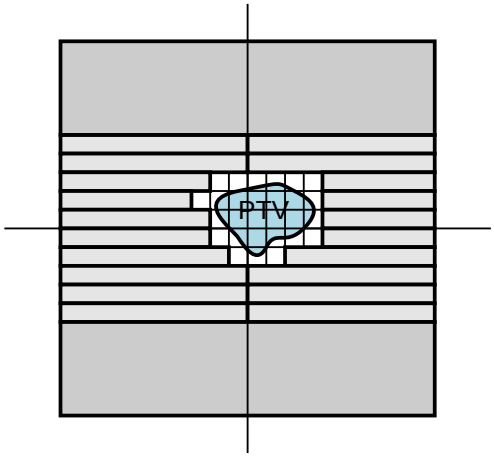

Abbildung 2.8: Die einzelnen Leafs werden an die PTV Kontur angepasst

Aperturkonfigurationen durchgeführt und für jede Einstrahlrichtung genau einmal über die VMC-Datei definiert. In DAO wird jene Konfiguration als Startkonfiguration und gleichzeitig als Definition der maximalen Leaföffnungen verwendet (Hard Constraints).

2.3.4 Segment-Objekte

Die Segment-Objekte sind das Kernstück des Programms. Als Instanzen der Klasse *Segment* bilden sie die einzelnen Aperturen (und Gewichte) des Bestrahlungsplans und damit die gesamte zu optimierende Konfiguration ab. Die Anzahl der Segmente pro Einstrahlrichtung wird beim Programmstart durch den Übergabeparameter `<n_segs>` festgelegt.

Koordinatensystem

Da mittels eines solchen Segment-Objekts eine einzelne Apertur und damit eine Bixelmatrix dargestellt werden soll, muss das Koordinatensystem dem der Kernel-Datei entsprechen. Das Koordinatensystem der Leafs, deren Positionen in der VMC-Datei in einem für das Bestrahlungsgerät lesbaren Format vorliegen, weicht davon allerdings ab. Da das Ergebnis der Optimierung ebenfalls als VMC-Datei ausgegeben werden muss um verifiziert werden zu können, müssen einfache Koordinatentransformationen vorgenommen werden. Abbildung 2.9 zeigt die wesentlichen Unterschiede zwischen den beiden Darstellungen. Im VMC-Format werden die Koordinaten der linken und rechten Leafs

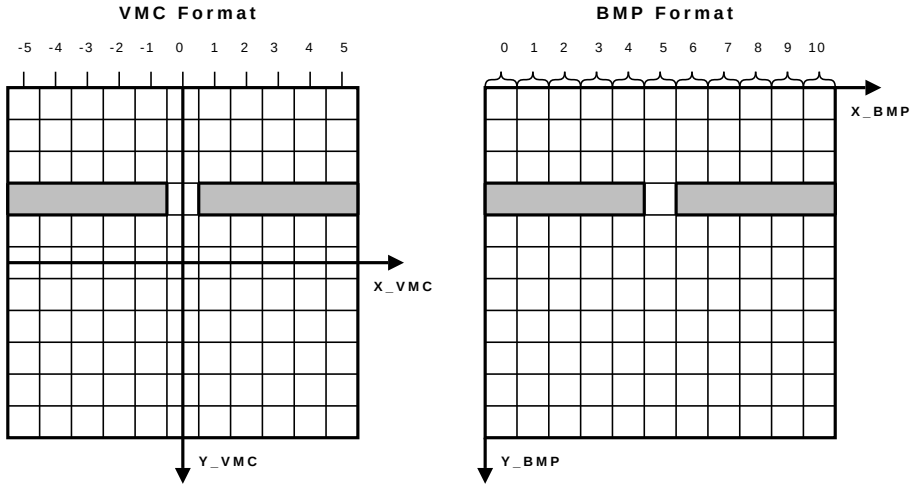

Abbildung 2.9: Unterschiede in der Koordinatendarstellung zwischen VMC-Dateien und Kernels; die grauen Balken repräsentieren Leafpaare

jeweils anhand ihrer Endkanten angegeben. Die Leafs in Abbildung 2.9 haben im VMC-Format die X-Koordinaten -0.5 und 0.5. In der VMC-Datei werden all diese Leafpaar Koordinaten untereinander listenförmig hinterlegt, weshalb eine explizite Angabe der Y-Koordinaten entfällt. Eine Umwandlung ins BMP-Format für Y-Koordinaten ist also nicht notwendig, da sie in der BMP ebenfalls listenförmig (zeilenweise) interpretiert werden. Bei den X-Koordinaten muss allerdings die Besonderheit beachtet werden, dass ein einzelnes Leaf über mehrere geschlossene bzw. offene Bixel dargestellt wird. Da diese Bi-

xel immer zusammenhängen müssen (weil sie sonst keine Leafs repräsentieren), genügt auch hier die Hinterlegung eines einzigen Zahlenpaars pro Leafpaar, die in Bixelmatrix Koordinaten angegeben werden. Damit vollständig geöffnete Aperturen dargestellt werden können, wird eine Leafposition durch die Koordinate des ersten offenen Bixels definiert. Im Beispiel aus Abbildung 2.9 wäre das für das linke Leaf 5 und für das rechte Leaf ebenfalls 5. Die Größe der Bixelmatrix ist grundsätzlich variabel, wurde im IKO System aber bisher auf 23×23 festgelegt. Für die Umrechnung der X-Koordinaten gelten daher die Gleichungen aus (2.10).

$$X_{BMP_{Links}} = X_{VMC_{Links}} + 11.5 \tag{2.10}$$
$$X_{BMP_{Rechts}} = X_{VMC_{Rechts}} + 10.5$$

Datenstrukturen

Die Speicherung der Bixeldaten eines Segments erfolgt durch klassische 2D-Arrays, deren genaue Größe jeweils über die Startkonfiguration bestimmt wird. Insgesamt werden drei Arrays verwendet um alle Bixelinformationen praktikabel zu hinterlegen:

- `MLCdata` [Datentyp: Integer] (Größe: $5 \times MLCs$)
- `BIXELid` [Datentyp: Integer] (Größe: $MLCs \times MLCwidth$)
- `BIXELon` [Datentyp: Boolean] (Größe: $MLCs \times MLCwidth$)

Dabei gibt $MLCs$ die Anzahl der Leafpaare und $MLCwidth$ „1 + die Differenz zwischen der kleinsten linken und größten rechten Leafposition“ für dieses Segment an, also die minimal benötigte (und verwendete) Segmentbreite. Das `MLCdata` Array beinhaltet die Zahlenpaare zur Beschreibung einzelner Leafpositionen (*Left*/*Right*) und deren durch die Startkonfiguration definierten Grenzen (*MinPos*/*MaxPos*). Abbildung 2.10 zeigt den Inhalt dieses Arrays für ein maximal geöffnetes Segment. Die Leafpaare (Bezeichnung: *Index*) sind hier spaltenweise von links nach rechts aufgeführt. Dem Screenshot lässt sich weiterhin die Anzahl der Leafpaare ($MLCs = 11$), sowie die Segmentbreite ($Width = 9 = MLCwidth$) und das aktuelle Segmentgewicht ($Weight = 0.1$) entnehmen. Redundant zu den Feldern *Left* und *Right* enthält das Array `BIXELon` die aktuelle Apertur als

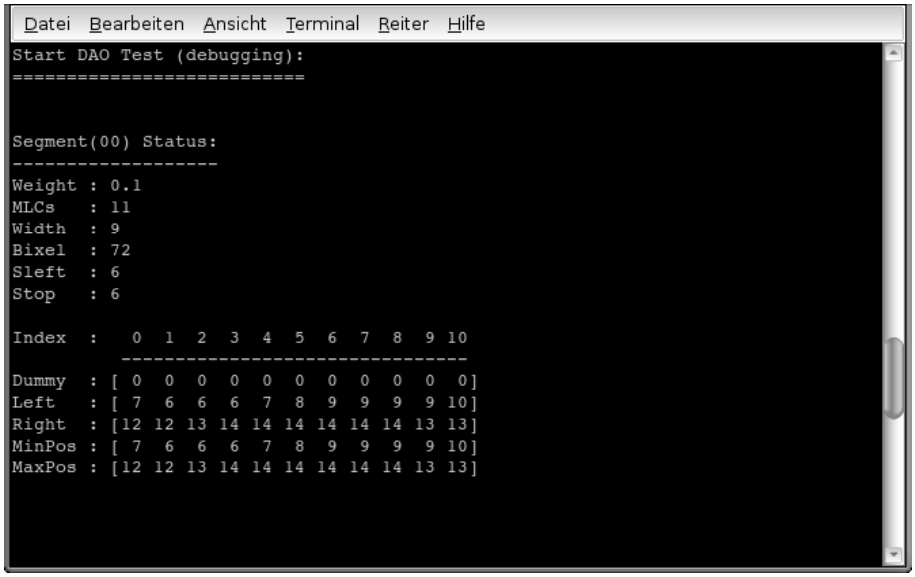

Abbildung 2.10: Status Anzeige eines Segment-Objekts; die Leafs sind maximal geöffnet

Bixelmatrix (Abbildung 2.11). Das identisch dimensionierte Array `BIXELid` beinhaltet für jeden Bixel die ID-Nummer, welche einen Bezug zu den Kernel-Daten herstellt. Nicht verwendete Bixel (die nicht in die Startkonfiguration fallen) erhalten die ID -1. Die Anzahl der tatsächlich verwendeten Bixel (*Bixel*-Wert in Abb. 2.10) ist also nur höchstens so groß wie die Bixelmatrix. Da auch nicht die gesamte Bixelmatrix, sondern nur der durch die Startkonfiguration (mit *MLCs* und *MLCwidth*) definierte Ausschnitt betrachtet wird, ist zur korrekten Interpretation und Ausgabe der VMC Daten die Relation (*Sleft*/*Stop*, siehe Abb. 2.10) zum Nullpunkt der Bixelmatrix erforderlich. Bei Änderungen sind die Arrays `MLCdata` und `BIXELon` gleichermaßen betroffen. `BIXELid` behält während der gesamten Optimierung seinen Inhalt bei.

Klassenbeschreibung

Abbildung 2.12 zeigt die wichtigsten Merkmale der Segment-Klasse. Neben dem Standard-Konstruktor, dem die Relation und Größe des zu verwendenden Ausschnittes der Bixelmatrix übergeben werden, wurde auch ein Copy-Konstruktor definiert. Dieser ist notwendig, da für jede Einstrahlrichtung gleich viele Segment-Objekte erzeugt werden müssen,

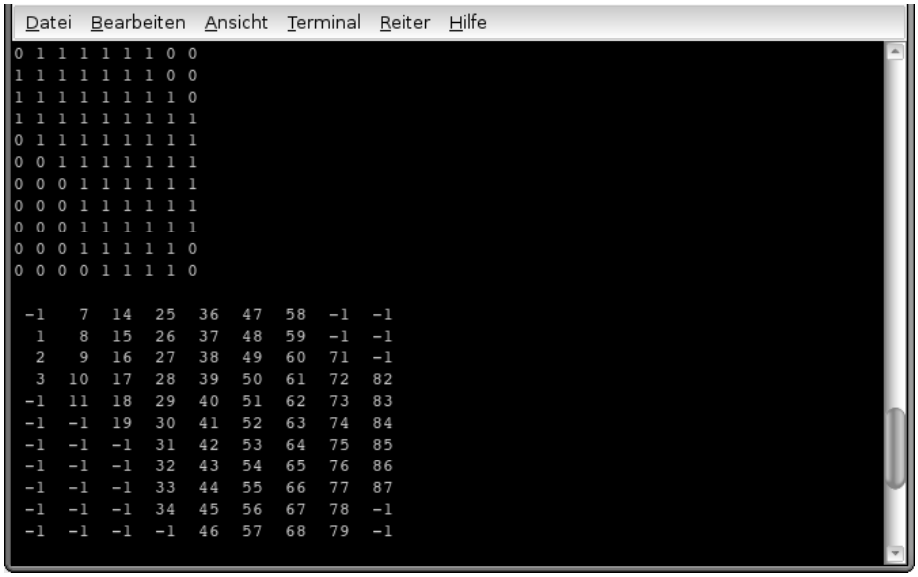

Abbildung 2.11: Bixelmatrix und Bixelindizierungen eines Segment-Objekts; die Leafs sind maximal geöffnet

die jeweils die exakt gleichen Constraints und Startkonfigurationen besitzen sollen. Eine einfache Kopie der über die VMC-Datei definierten *Grundsegmente* vereinfacht diesen Vorgang. Dennoch können in der VMC-Datei auch mehrere Segmente pro Eintrahlrichtung vordefiniert werden. Dabei erhalten dann alle Segmente einer Einstrahlrichtung die Constraints des zuerst definierten Grundsegments und behalten ihre vorgegebene Konfiguration bei. Sollen mehr Segmente erzeugt werden als vordefiniert wurden, so wird wieder das Grundsegment kopiert. Man erhält dadurch die Möglichkeit manuelle Startkonfigurationen vorzudefinieren, jedoch bleibt das Grundsegment davon ausgenommen, da es die maximale Aperturöffnung definieren muss. Die folgenden beiden Methoden der Segment-Klasse ermöglichen das gegenseitige Anpassen des `MLCdata` und `BIXELon` Arrays:

- SetBIXELbyMLC - Setzt das `MLCdata` Array entsprechend der Bixelmatrix
- SetMLCbyBIXEL - Setzt die Bixelmatrix entsprechend dem `MLCdata` Array

Zur praktischen Verwendung der Segment-Klasse sind die folgenden Property-Methoden ausreichend:

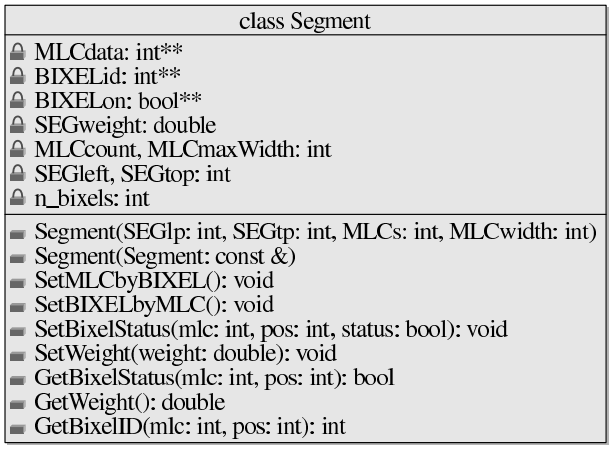

Abbildung 2.12: Die wichtigsten Attribute und Methoden der Segment-Klasse

- [Get/Set] BixelStatus - abfragen/setzen einzelner Bixel
- [Get/Set] Weight - Segmentgewicht abfragen/setzen
- GetBixelID - Abfragen der ID eines Bixels; ermöglicht Zugriff auf Kernel Daten

2.3.5 Programmstruktur

Damit DAO als eigenständiges Programm funktioniert und leicht in die IKO Software integriert werden kann mussten einige IKO Klassen eingebunden und geringfügig adaptiert werden. Die Dateinamen dieser adaptierten Klassen erhielten ein vorangestelltes *so* (*stand alone*) um sie von den eigentlichen DAO Klassen besser abzugrenzen. Die im Anhang befindliche Tabelle A.3 (Seite 108) listet alle verwendeten IKO Klassen auf. Die zu DAO gehörenden Neuentwicklungen sind in Tabelle A.4 (Anhang: Seite 109) aufgelistet. Zur Bereitstellung allgemeiner Informationen in den Klassen werden in den Konstruktoren jeweils Zeiger auf Datencontainer-Objekte übergeben, die in der DAO-Klasse erzeugt werden (Tabelle A.5 im Anhang, Seite 109). Diese Informationen sind im Einzelnen die eingelesenen Optimierungsparameter und die Eigenschaften des Bestrahlungsplans. Die Übergabe von Observablen erfolgt ebenfalls über einen Datencontainer. Die wichtigsten Assoziationen der verschiedenen Klassen untereinander, sind in Abbildung 2.13 als UML Grafik dargestellt.

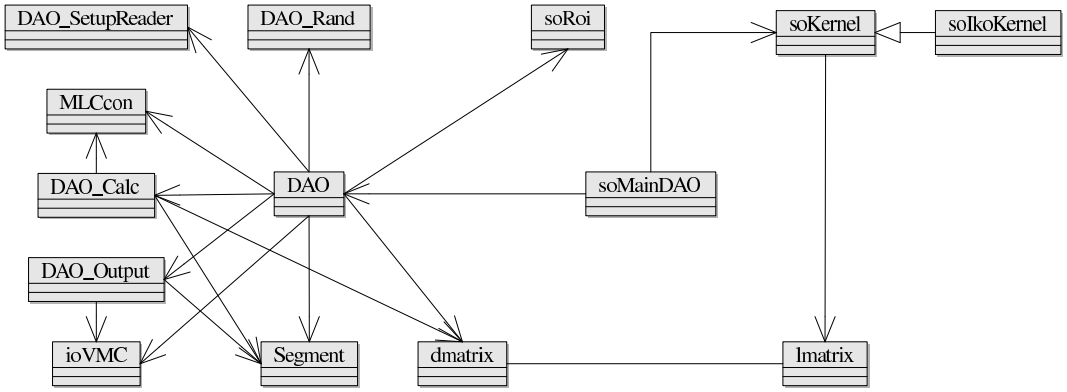

Abbildung 2.13: DAO Klassendiagramm

2.3.6 Programmablauf

Der Ablauf der DAO Optimierung lässt sich, wie im Flussdiagramm (Abbildung 2.14) angedeutet wird, in sechs Teile gliedern. In den folgenden Abschnitten werden diese näher erläutert.

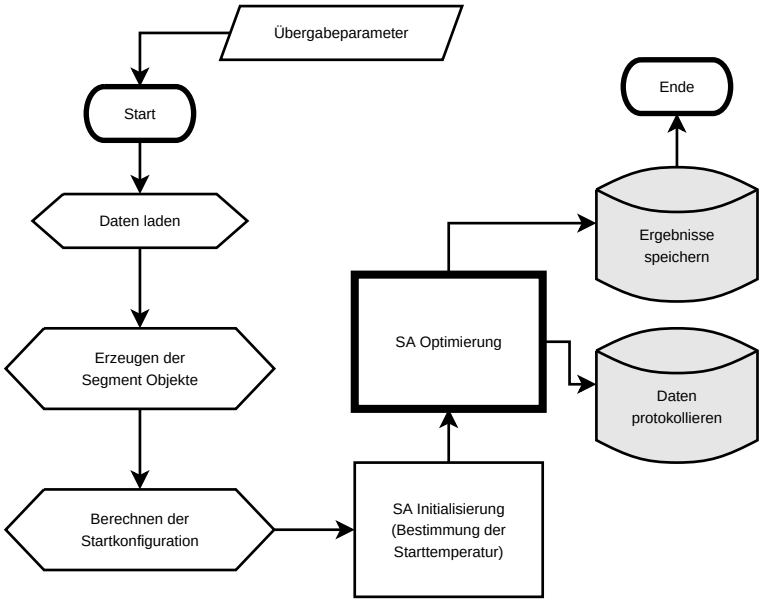

Abbildung 2.14: DAO Flussdiagramm

i) Daten einlesen

Nach dem Programmstart wird zuerst die Kernel-Datei eingelesen, die zuvor mit XVMC erstellt wurde. Bei der Verwendung von BigKernels erfolgt eine entsprechende Bildschirmausgabe und es wird nur die für die Optimierung relevante Datei verwendet. Ganz am Schluss des Programmdurchlaufs muss die zweite Kerneldatei ebenfalls eingelesen, mit den vorhandenen Daten verbunden (merge) und die Dosis im gesamten Volumen anhand der optimierten Konfiguration berechnet werden um das Endergebnis zu erhalten. Da beim ersten Auslesen zunächst nur Informationen über den benötigten Umfang an Arbeitsspeicher gesammelt werden, müssen Kernel-Dateien jeweils immer zwei mal eingelesen werden. Bei Erfolg wird die zum Bestrahlungsplan gehörige ROI-Datei geladen und den definierten OARs und dem PTV die entsprechenden Voxel-IDs zugewiesen. Bis zu dieser Stelle wurden bereits vorhandene IKO Routinen verwendet. Es folgen die DAO spezifischen Datendateien, beginnend mit der DAO Konfigurationsdatei, die durch den Übergabeparameter `'SetupFile'` angegeben wird. Diese Konfigurationsdatei enthält die Steuerparameter für den Simulated Annealing Algorithmus, sowie die Gewicht Constraints (Abschnitt 2.5.6). Es folgt das Laden der VMC-Datei, deren Inhalt danach auf Gültigkeit und Kompatibilität überprüft wird. Ein weiterer Übergabeparameter `'DeviceConfig'` legt schließlich fest welche Constraints für den (durch die VMC-Datei spezifizierten) Beschleuniger verwendet werden sollen. Danach stehen alle nötigen Informationen zur Verfügung. Die zugehörige Programmausgabe befindet sich im Anhang (Listing A.3, Seite 108).

ii) Segment-Objekte erzeugen

Nach dem Einlesen aller benötigten Informationen erfolgt die Instanzierung von Segment-Objekten, deren Anzahl und Eigenschaften anhand der VMC Daten durch den Benutzer (user-defined) festgelegt wurden. In der Instanz der `ioVMC` Klasse liegen diese Daten im Rohformat vor und müssen in Segment-Objekte übersetzt werden. Dazu erfolgt die Bestimmung der einzelnen Leafpositionen, die Anzahl der Leafpaare und damit die Größe der zu verwendenden Bixelmatrizen. Das Gesamtgewicht jeder Einstrahlrichtung (Sum-

me über die Einzelgewichte der zugehörigen Segmente) wird zu Beginn der DAO Optimierung auf 1 gesetzt. Entsprechend enthält jedes Segment das Gewicht $\frac{1}{<\text{n\_segs}>}$. Die Gewichte der VMC-Datei werden nur dann übernommen, wenn dort für alle Segmente eine benutzerdefinierte Konfiguration vorhanden ist. Dadurch können die Ergebnisse anderer Optimierungen als Startkonfiguration verwendet werden. Nach dem Erstellen der benutzerdefinierten Segment-Objekte erfolgt die Bestimmung und Ausgabe (siehe Anhang: Listing A.4, Seite 110) der Anzahl der Parameter, welche die Länge `n_para` der Parameterliste (Abschnitt 2.5.1) bestimmt. Da per VMC-Datei im Regelfall nur die Grundsegmente jeder Einstrahlrichtung definiert werden, müssen die noch fehlenden Segmente durch Kopieren erzeugt werden, so dass für jede Einstrahlrichtung genau `<n_segs>` viele Segmente vorliegen.

iii) Startwerte berechnen

Die Startkonfiguration des Systems ist durch die Erzeugung der Segmente festgelegt worden. Bevor mit der Optimierung begonnen werden kann, muss aber noch die zugehörige Dosismatrix und der entsprechende Zielfunktionswert bestimmt werden (siehe Anhang: Listing A.5, Seite 110). Die Dosisberechnung erfolgt über Methoden der Klasse `DAO_Calc`. Dazu wird für jeden (durch die Bixel-ID) eindeutigen Bixel die Fluenz gemäß Gleichung (2.6) bestimmt und anschließend anhand der inversen Kernels die Dosiswerte in allen betrachteten Voxeln berechnet. Die Solldosis für das PTV wird wie in (2.12) definiert, wobei D_{ref} die anhand der Startkonfiguration resultierende mittlere Dosis im PTV (2.11) ist und $D_{min}^{PTV}/D_{max}^{PTV}$ dessen minimale bzw. maximale Solldosis vorgeben.

$$D_{ref} := \frac{1}{V_{PTV}} \sum_{i=1}^{V_{PTV}} D_i \tag{2.11}$$

$$D_0^{PTV} := \frac{D_{min}^{PTV} + D_{max}^{PTV}}{2} \cdot D_{ref} \tag{2.12}$$

Da Kernel-Dateien absolute Dosiswerte enthalten, können durch den Faktor $\frac{1}{D_{ref}}$ sämtliche Berechnungen auf relative Werte bezogen werden. Dadurch vereinfacht sich die Anwendung der DV-Constraints. In der vorliegenden Version (1.3.8) von DAO sind diese noch hart im Quellcode programmiert, da der Aufwand einer Adaption der dazu

bereits vorhandenen IKO Routinen zu aufwändig gewesen wäre. Auch weil diese Routinen nach der Integration von DAO ins IKO System ohnehin ansprechbar sind, wurde auf eine Adaption vorerst verzichtet. Änderungen der Constraints und Penalties erfordern demnach jedes mal eine Neucompilierung des DAO Programms. Dabei wurden für alle verwendeten Bestrahlungspläne ID-Nummern vergeben, die beim Programmaufruf mit dem Parameter `<pat_ID>` angesprochen werden können und das Laden der entsprechenden Kernel- und ROI-Dateien veranlasst.

iv) SA Initialisierung

Bevor nun die eigentliche Optimierung startet wird die Starttemperatur T_{Start} des SA Systems bestimmt (SA Initialisierung). Dazu wird eine spezielle Form des SA Algorithmus (siehe Abschnitt 2.4) verwendet, bei dem einige Male wiederholt zufällige Änderungen durchgeführt werden, die alle von der Startkonfiguration ausgehen, also niemals übernommen werden. Nach jeder Änderung wird die Zielfunktion und daraus resultierend die Energiedifferenz $\Delta\mathcal{H}$ berechnet. Diejenigen Differenzen, für die sich Verschlechterungen ergeben ($\Delta\mathcal{H} > 0$), werden aufaddiert und daraus am Ende der Mittelwert gebildet. Man erhält so mittels Gleichung (2.13) eine Starttemperatur, für die sich eine Akzeptanzrate von mindestens P_{acc} ergibt [22].

$$T_{Start} = -\frac{\Delta\mathcal{H}_{mean}}{\ln P_{acc}} \tag{2.13}$$

Zu Beginn der Optimierung wird ein möglichst uneingeschränkter Gang durch den Phasenraum verlangt, weshalb für P_{acc} für gewöhnlich ein Wert nahe 0.9 gewählt wird. Nach dem Durchlauf der vorgesehenen Anzahl an Iterationen, wird die Dosismatrix und der Zielfunktionswert (RealOF) vollständig neu berechnet. Das ist notwendig, da sowohl bei der SA Initialisierung, als auch bei der SA Optimierung immer nur die Änderungen der Dosismatrix berechnet werden. Das bringt einen enormen Geschwindigkeitsvorteil, wobei jedoch die Gefahr von Genauigkeitsverlusten besteht. Durch die Verwendung der `dmatrix` Klasse können diese aber weitgehend vermieden werden. Dennoch wird während der Optimierung in bestimmten Abständen, die durch einen Optimierungsparameter

einstellbar sind, die Dosismatrix vollständig erneuert um eventuell fehlerhafte Werte zu korrigieren. Direkt nach der Initialisierung muss auf diese Weise eine fehlerfreie Grundlage für die folgende Optimierung geschaffen werden. Die zugehörige Bildschirmausgabe befindet sich im Anhang (Listing A.6, Seite 110) und zeigt als Ergebnis u.a. die ermittelte Starttemperatur.

v) SA Optimierung

Eine detailierte Beschreibung der Funktionsweise des DAO Simulated Annealing Algorithmus erfolgt in Abschnitt 2.5. Daher soll hier nur die Bildschirmausgabe (Anhang: Listing A.7, Seite 111) erläutert werden. Bevor die Optimierung beginnt werden ggf. (einstellbar durch einen Optimierungsparameter) Protokolldateien geöffnet und ein entsprechender Hinweis ausgegeben. Es folgt sofort der erste Iterationsschritt. Hinter dem Ausgabetext "Iterate(n): [" wird eine einfache Balkenanimation dargestellt, die sich regelmäßig nach 30 (versuchten) Parameteränderungen verändert. Dadurch kann der Benutzer die Programmaktivität wahrnehmen. Die in Klammern gefasste Zahl n gibt die Anzahl der Iterationen an die durchgeführt werden, bevor das nächste Mal die Temperatur abgesenkt wird (näheres siehe Abschnitt 2.5). Nach dem Abarbeiten dieser n Iterationen ist ein sogenannter *Loop* abgeschlossen, der einen erweiterten Iterationsbegriff des DAO SA Algorithmus darstellt. Nach jedem Loop erfolgt eine Bildschirmausgabe mit den Ergebnissen und direkt darunter die Balkenanimation des nächsten Loops. Die Ergebnisausgabe ist in drei Teile gegliedert. Der obere Teil gibt den Systemzustand wieder. Dabei wird ausgegeben welchem Anteil der Segment-Objekte ein Gewicht von 0 zugeordnet wurde. Über einen Optimierungsparameter lässt sich bestimmen, ob solche Segmente überhaupt zulgelassen werden sollen. Im Regelfall ist das, wegen der durch den Verlust von Segmenten verringerten Möglichkeiten zur Fluenznachbildung, nicht erwünscht. Die nächste Zeile zeigt die absolute Verbesserung der Zielfunktion an, die durch den letzten Loop erzielt wurde. Erfolgte aber eine Verschlechterung, so bleibt diese Zeile leer. Direkt folgend ist der bisher minimalste Zielfunktionswert (OFmin) und der zur Konfiguration des letzten Loops gehörige Zielfunktionswert (OFref) angegeben (bei einer Verbesserung sind

diese beiden Zahlen immer identisch). Der mittlere Teil der Ausgabe zeigt die Zustände der vier verschiedenen in DAO integrierten Abbruchbedingungen an (siehe ebenfalls Abschnitt 2.5). Hier ist auch die Systemtemperatur des vorherigen Loops zu sehen. Der untere Teil der Ausgabe gibt die in Leafs und Gewichte aufgegliederten Akzeptanzraten, sowie die Gesamtakzeptanzrate an. Falls nach dem letzten Loop die Dosismatrix neu bestimmt wurde, wird noch die Verifikation in Bezug auf den zuvor benutzten Zielfunktionswert ausgegeben. Bei mehr als zwei sich unterscheidenden Dezimalstellen wird die Optimierung abgebrochen, andernfalls Warnungen ausgegeben. Derartige Stellenverluste sind gewöhnlich auf fehlerhafte Bestrahlungspläne zurückzuführen und treten praktisch nicht auf.

vi) Ausgeben und speichern der Ergebnisse

Nach Abschluss der Optimierung wird die eingetretene Abbruchbedingung angezeigt (Anhang: Listing A.8, Seite 112) und die Dosismatrix anhand der zum minimalen Zielfunktionswert gehörenden Aperturkonfiguration neu berechnet. Die Gegenüberstellung mit dem Startwert liefert somit die prozentuale Verbesserung des Systemzustandes. Bei zwei verschiedenen Rechnungen sind diese Prozentwerte für Vergleichszwecke jedoch nur von Bedeutung, wenn sowohl die DV-Constraints, als auch die Penalties der ROIs und damit die gesamte Zielfunktion identisch sind. Es folgt schließlich die Speicherung verschiedener Ergebnisdateien. Die optimierte Aperturkonfiguration wird in der VMC-Datei “Result.vmc“ hinterlegt, wobei ggf. Segmente mit einem Gewicht von 0 ignoriert werden. Falls BigKernels verwendet wurden müssen im Folgenden noch die Kernels der Außenkontur eingelesen werden um daraus schließlich die vollständige Volumenmatrix (merged kernels) anhand der optimierten Aperturkonfiguration zu berechnen. Die Belastung für den Arbeitsspeicher des Computersystems ist an dieser Stelle am größten. Für gewöhnlich muss hier mindestens 90% bis 95% der Dateigrößen beider Kernel-Dateien an Arbeitsspeicher zur Verfügung stehen. Nach dem Erfolgreichen Verbinden der Kernels wird die berechnete Dosismatrix im ASCII- und Binärformat exportiert. Nach der darauf folgenden Hinterlegung der Optimierungsergebnisse (“Result.rec“) ist der Programmdurchlauf

beendet. Bei Angabe des optionalen Übergabeparameters `<runID>` werden alle Ausgabedateien in einen Unterordner mit eben dieser Nummer abgelegt, um bei Bedarf das Ordnen verschiedener Rechnungen zu ermöglichen.

2.3.7 Dosisberechnung

Ein wichtiger Bestandteil der DAO Optimierung ist die programmatische Umsetzung der Dosisberechnung. Da nach der Änderung einer Entscheidungsvariablen (*Parameter* der Zielfunktion) nur ein bestimmter Teil und nicht die gesamte Dosismatrix entsprechend aktualisiert werden muss, bietet es sich an diesen Umstand auszunutzen um Rechenzeit einzusparen. Problematisch ist nun, dass Parameteränderungen immer verworfen werden können, was bei niedrigen Temperaturen sehr häufig passiert. Deshalb müssen Änderungen in der Dosismatrix immer rückgängig gemacht werden können. In DAO erfolgt das mittels Rückrechnung.

Berechnen der Dosismatrix

Zur Darstellung der Fluenz, die letzendlich anhand der inversen Kernels die Dosisverteilung bestimmt, wird in der `DAO` Klasse das eindimensionale Array `flumc` erzeugt und per Referenz an die `DAO_Calc` Instanz übergeben. Dieses Fluenzarray hat die Länge `n_pixs`. Dessen Wert wird beim Einlesen der Kernel-Datei nach dem Programmstart ermittelt und gibt die Anzahl aller Bixel (und damit auch die Anzahl der inversen Kernels) an. Bei der DAO Initialisierung erhalten die einzelnen Elemente des Fluenzarrays die anhand der Startkonfiguration resultierenden Fluenzen gemäß Gleichung (2.6) (siehe Seite 21), die bei maximaler Öffnung der Segmente alle den Wert 1 haben. Da die Kerneldatensätze fortlaufend nummeriert sind (Bixel-IDs) können die zugehörigen Intensitäten dem Fluenzarray direkt entnommen werden. Zu jedem Bixel ist nun eine Intensität (Fluenz) und ein inverser Kernel gegeben, der einen Datensatz an Voxel-IDs mit entsprechenden Dosisanteilen (die sich auf eine Intensität von 1 beziehen) enthält. Durch multiplizieren dieser Dosisanteile mit der verknüpften Intensität aus dem Fluenzarray und dem aufaddieren dieser Produkte in den entsprechend der Voxel-IDs eindeutig definierten Elementen der

Dosismatrix, lässt sich diese vollständig bestimmen, indem dies für jeden (geöffneten) Bixel genau einmal durchgeführt wird. Die resultierende Dosismatrix repräsentiert dann die zur verwendeten Aperturkonfiguration gehörende Dosisverteilung. Durch die Verwendung der inversen Kernels lässt sich so die Dosisverteilung einer beliebigen (gültigen) Aperturkonfiguration darstellen.

Aktualisieren der Dosismatrix

Wegen dem enormen Datenumfang der Kernel-Datei, die mit der Größe der Dosismatrix korelliert, ist die vollständige Berechnung der Dosismatrix mit hohem Rechenaufwand verbunden. Da bei Parameteränderungen durch den Optimierungsalgorithmus aber immer nur eine Teilmenge der Bixel betroffen ist, lässt sich die Effizienz deutlich erhöhen indem man ausschließlich jene Elemente der Dosismatrix ändert, die durch diese Änderungen auch betroffen sind. Abhängig davon ob eine Leafposition oder ein Segmentgewicht geändert wurde, müssen dazu zwei verschiedene Methoden benutzt werden. Bei Leafänderungen werden nur wenige Bixel eines einzigen Segments entweder geöffnet oder geschlossen. Die Änderung der Dosis D_i im Voxel i (gemäß Gleichung (2.8), Seite 21) wird daher durch die Änderungen der Binärschalter von $\Theta_{k,j}$ in $\Theta_{k,j}^{neu} = 1 - \Theta_{k,j}$ vollständig beschrieben. Der Dosisbeitrag (2.14) des Segments k im Voxel i ändert sich entsprechend in (2.15) und kann durch einfache Addition der Dosisänderung $\Delta D_{i,k}$ (2.16) bestimmt werden, wobei $\Delta Bixel(k)$ die Menge der geänderten Bixel ist. Folglich gilt für die Gesamtdosis im Voxel i (2.17). Das Fluenzarray wird angepasst, indem das Segmentgewicht von den entsprechenden Elementen entweder addiert oder subtrahiert wird.

$$D_{i,k} = \omega_k \cdot \sum_{j \in Bixel(k)} \Theta_{k,j} \cdot IK_{i,j} \tag{2.14}$$

$$D_{i,k}^{neu} = \omega_k \cdot \sum_{j \in Bixel(k)} \Theta_{k,j}^{neu} \cdot IK_{i,j} \tag{2.15}$$

$$\Delta D_{i,k} := D_{i,k}^{neu} - D_{i,k} = \omega_k \cdot \sum_{j \in \Delta Bixel(k)} (1 - 2 \cdot \Theta_{k,j}) \cdot IK_{i,j} \tag{2.16}$$

$$D_i^{neu} = D_i + \Delta D_{i,k} \tag{2.17}$$

Für welche Voxel i diese Aktualisierung vorgenommen werden muss wird durch die inversen Kernels der betroffenen Bixel bestimmt. Muss die Dosis aufgrund eines geänderten Segmentgewichts aktualisiert werden, sind dabei grundsätzlich alle (offenen) Bixel dieses Segments betroffen. Die Anpassung der Voxel der Dosismatrix erfolgt dann ebenfalls durch Addition des Differenzbeitrags. Dieser kann als Produkt des alten Dosisbeitrags $D_{i,k}$ des Segments k mit dem angepassten Gewichtungsfaktor ω_k^* (2.18) direkt berechnet werden, da zusammen mit (2.14) und (2.19) folglich (2.20) und damit (2.21) gilt. Die Anpassung des Fluenzarrays kann hier durch einfaches addieren von ω_k^* auf die betroffenen Elemente realisiert werden.

$$\omega_k^* := \omega_k^{neu} - \omega_k \tag{2.18}$$

$$D_{i,k}^{neu} = \omega_k^{neu} \cdot \sum_{j \in Bixel(k)} \Theta_{k,j} \cdot IK_{i,j} \tag{2.19}$$

$$\Delta D_{i,k} := D_{i,k}^{neu} - D_{i,k} = D_i + (\omega_k^{neu} - \omega_k) \cdot D_{i,k} \tag{2.20}$$

$$D_i^{neu} = D_i + \Delta D_{i,k} = D_i + \omega_k^* \cdot D_{i,k} \tag{2.21}$$

Wegen der notwendigen Betrachtung vieler Bixel (und entsprechend vieler Voxel) bei Gewichtänderungen, können diese allgemein als weitaus rechenintensiver gegenüber den Leafänderungen angenommen werden.

Wiederherstellung der Dosismatrix

Falls der durch die geänderte Dosismatrix resultierende Zielfunktionswert vom DAO SA Algorithmus nicht akzeptiert wird, muss die entsprechende Parameteränderung und damit auch die Dosismatrix zurückgesetzt werden. Weil das Neuberechnen der gesamten Matrix zu rechenintensiv ist, sollten dabei nur die vollzogenen Änderungen rückgängig gemacht werden. Da bei Parameteränderungen die Aktualisierung der Dosismatrix allein durch Additionen und Subtraktionen erfolgen kann, besteht mit der Rückrechnung eine naheliegende Möglichkeit zur Datenwiederherstellung. Dazu werden die Änderungen der Dosismatrix invers wiederholt. Die zuvor addierten/subtrahierten Zahlenwerte werden also wieder abgezogen/aufaddiert. Diese Methode hat sich als weniger zeitintensiv im Vergleich zu dynamischen Datensicherungen erwiesen.

2.4 Simulated Annealing (SA) Grundlagen

Bei sehr komplexen Zielfunktionen liegt meist auch eine sehr komplexe Topologie des Suchraums vor. Das heißt es besteht die Möglichkteit der Existenz zahlreicher lokaler Minima. Deterministische Suchalgorithmen wie Gradienten- oder Simplexverfahren müssen hier im Allgemeinen zwangsweise scheitern, da diese wegen ihrer zugrundeliegenden Systematik bei derart komplexen und in ihrer Struktur unbekannten Suchräumen schnell in einem solchen lokalen Minimum *hängen* bleiben. Es ist daher oft zwingend notwendig alternative Algorithmen zu verwenden, die kein derartig hohes Risiko für diesen Fall bergen. Als erfolgsversprechend gelten hier stochastische Optimierungsverfahren [5], wie beispielsweise *Simulated Annealing* (simulierte Abkühlung) oder Genetische Algorithmen.

2.4.1 Temperatur und Energie

Das SA Verfahren stammt ursprünglich aus der Festkörperphysik und dem Problem, dass die Atome stark aufgeheizter Körper nach Abkühlung sich nur dann in einem Zustand niedrigster Energie befinden können, wenn diese Abkühlung des Körpers langsam erfolgt. Der Grund dafür ist, dass bei langsamer Abkühlung die Atome mehr Zeit haben um stabile Positionen einzunehmen. Übertragen auf Optimierungsprobleme entstand so die Idee ein zu optimierendes System als fiktiven Körper und die Entscheidungsvariablen als fiktive Atome zu betrachten. Das heißt man erzeugt durch stochastische Methoden Änderungen der Werte der Entscheidungsvariablen und kühlt das System mit einer gedachten Temperatur langsam ab, was das Einpendeln des Systems in einen optimalen Zustand ermöglichen soll. Aufgrund des historischen Hintergrundes bezeichnet man bei SA-Optimierungen den Wert der Zielfunktion und damit die Qualität des aktuellen Zustandes als *Energie* oder (physikalisch betrachtet) als Entropie $\mathcal{H}$. Auch der Verlauf der Optimierung wird diesem phyiskalischen Prinzip nachempfunden, indem Verbesserungen prinzipiell akzeptiert werden, aber die Akzeptanz von Verschlechterungen proportional zur Temeperatur abfällt und schließlich gegen Null geht. Der offensichtliche Vorteil gegenüber Gradientenmethoden ist hier, dass der Algorithmus bei genügend hohen Tem-

peraturwerten lokale Minima auch wieder verlassen kann. Ein solcher Vorgang lässt sich als Markow-Kette darstellen, wobei gezeigt werden kann, dass das Erreichen des globalen Minimums im Allgemeinfall immer möglich ist. Details dazu sind der Fachliteratur [7, 14] zu entnehmen.

2.4.2 Funktionsprinzip

Die Konfiguration K eines zu optimierenden Systems lässt sich als n-dimensionaler Vektor auffassen, dessen Elemente die Werte der einzelnen Entscheidungsvariablen annehmen. Der Phasenraum Γ ist dann ebenfalls n-dimensional. Um das globale Minimum K_{opt} oder eine dem globalen Minimum nahe liegende Lösung der Zielfunktion zu finden geht man mit SA Algorithmen prinzipiell folgendermaßen vor:

1. Vorgabe oder Würfeln einer zufälligen Startposition ($n = 1$): $K_1 = K_{Start}$
2. Auswertung der Zielfunktion (Energie) an der Stelle K_1
3. Zufällige Änderung von K_n in K_n^*
4. Auswertung der Zielfunktion (Energie) an der neuen Stelle K_n^*
5. Qualifizierung und ggf. Übernahme der Zustandsänderung:
   - Akzeptanz von K_n^* bei Verbesserung der Zielfunktion: $K_{n+1} = K_n^*$
   - Akzeptanz von K_n^* bei Verschlechterung nur falls ein temperatur- und zufallsabhängiges Kriterium erfüllt wird: $K_{n+1} = K_n^*$, falls $x < p(K_n, K_n^*, T)$, wobei x die Realisierung einer Zufallsvariablen $X \sim UNIF(0, 1)$ ist
   - Im Falle der Ablehnung gilt: $K_{n+1} = K_n$
6. Verringern der Temperatur gemäß einer festen Vorschrift: $T_{n+1} = t(n)$
7. Beenden des Algorithmus falls ein Abbruchkriterium erfüllt wird, andernfalls Beginn der nächsten Iteration ($n = n + 1$) ab Punkt 3
8. Ergebnis: $K_{min} = \min\{K_1, K_2, ...\}$, wobei gilt: $OF(K_{min}) \geq OF(K_{opt})$

An diesem Schema lässt sich ablesen, dass es sich bei SA um ein Verfahren handelt welches eine neue Konfiguration aus einer alten durch Veränderung erzeugt. Das SA Verfahren fällt damit in die Kategorie der physikalisch motivierten Alt-Neu-Verbesserungsheuristiken. Die Akzeptanz von Verschlechterungen wurde in DAO mit dem Metropolis-Kriterium [13] realisiert (2.22), wobei gilt: $\mathcal{H} := OF(K)$ bzw. $\Delta\mathcal{H} := OF(K_n^* - K_n)$.

$$p(K_n, K_n^*, T) := exp\left(-\frac{\Delta\mathcal{H}}{T}\right) \tag{2.22}$$

Die Übergangswahrscheinlichkeit der Markow-Kette von einem Zustand in einen anderen lässt sich damit als (2.23) zusammenfassen.

$$p(K_n \to K_n^*) = \begin{cases} 1 & \text{falls } \Delta\mathcal{H} < 0 \\ p(K_n, K_n^*, T) & sonst \end{cases} \tag{2.23}$$

Durch die Verwendung des Metropolis-Kriteriums konvergiert die Verteilungsfunktion $F(K)$ der Zustände der Markow-Kette gegen die Bolzmann-Verteilung (2.24), wodurch sich das System im thermischen Gleichgewicht befindet, da das *detailed balance* Prinzip (2.25) erfüllt wird.

$$F(K) := \frac{1}{Z} exp\left(-\frac{\mathcal{H}}{T}\right), \text{ mit der Zustandssumme } Z \tag{2.24}$$
$$\text{und } K \text{ Gibbs-Boltzmann verteilt}$$

$$\frac{p(K_n \to K_n^*)}{p(K_n^* \to K_n)} = \frac{P(K_n^*)}{P(K_n)} = exp\left(-\frac{\Delta\mathcal{H}}{T}\right) \tag{2.25}$$

Durch Ableiten des Erwartungswertes der Energie nach der Temperatur erhält man die *spezifische Wärme* $\langle C\rangle = E(C)$ (2.26), welche eine während der Optimierung wichtige *Observable* darstellt.

$$\langle C\rangle = \frac{\partial\langle\mathcal{H}\rangle}{\partial T} = \frac{\partial}{\partial T}\frac{1}{Z}\sum_{K\in\Gamma}\mathcal{H}\cdot exp\left(-\frac{\mathcal{H}}{T}\right) = \frac{1}{T^2}Var(\mathcal{H}) = \frac{\langle E^2\rangle - \langle E\rangle^2}{T^2} \tag{2.26}$$

Dabei ist $E := E(\mathcal{H}) = \langle\mathcal{H}\rangle = \frac{1}{Z}\sum_{K\in\Gamma}\mathcal{H}\cdot exp\left(-\frac{\mathcal{H}}{T}\right)$. Die Verringerung der Systemtemperatur erfolgt in der Praxis meist exponentiell (2.27) oder linear (2.28). Weitere Details

zu diesen Ausführungen können den Arbeiten diverser Autoren [9, 22] entnommen werden.

$$t_\alpha(n) := T_n \cdot \alpha, \text{ für } 0 < \alpha < 1 \tag{2.27}$$

$$t_\beta(n) := T_n - \beta, \text{ für } \beta > 0 \tag{2.28}$$

2.4.3 Das „magische Dreieck“

Im sehr einfachen Fall einer eindimensionalen Zielfunktion lässt sich das Prinzip der SA Optimierung gut anhand eines Ballonfahrers veranschaulichen (Abbildung 2.15). Bei

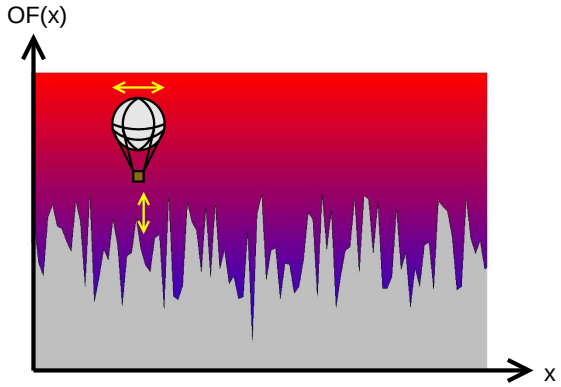

Abbildung 2.15: Schematische Darstellung des SA Prinzips anhand einer 1D-Zielfunktion $(OF(K) = OF(x))$

hohen Temperaturen schwebt der Ballon weit über der stark zerklüfteten Zielfunktion und nimmt den Wert $OF(x)$ an seiner aktuellen Position x an. Durch eine zufällige Änderung bewegt sich der Ballon entweder nach links oder nach rechts und nimmt einen neuen Wert $OF(x^*)$ an. Je höher die Temperatur ist, desto besser ist der Ballonfahrer in der Lage höhere Gipfel zu überwinden, so dass ohne Weiteres $OF(x^*) > OF(x)$ sein kann. Sinkt die Temperatur allmälich ab beginnt der Ballon sich in einem lokalen Minimum festzufahren. Nun gibt es einen Temperaturbereich, in dem der Ballon größere Gipfel gerade noch überwinden kann, ohne zu beginnen sich bereits festzufahren. Genau hier nimmt die spezifische Wärme ein Maximum an (Abbildung 2.16). Zwischen den

Wendepunkten T0 und T1 (die zusammen mit dem Maximum das „magische Dreieck“ bilden) liegt somit ein Temperaturbereich vor, in dem der Ballon theoretisch jeden beliebigen Zielfunktionswert annehmen kann und somit durch den Phasenraum schwebt ohne zu konvergieren [15]. Diesen Umstand nutzt man aus, indem innerhalb dieses Tem-

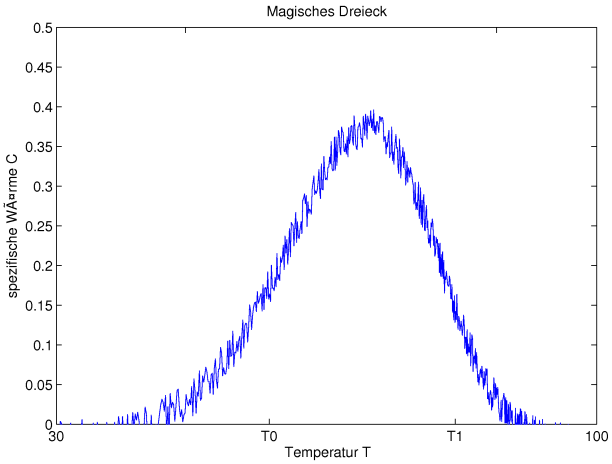

Abbildung 2.16: Beispiel eines magischen Dreiecks; die spezifische Wärme ist gegenüber der Temperatur in einem logarithmischen Diagramm aufgetragen

peraturbereichs die Anzahl Iterationsschritte des SA Algorithmus erhöht wird, wodurch die Wahrscheinlichkeit das globale oder ein sehr gutes lokales Minimum zu finden steigt. Der Temperaturbereich muss jedoch empirisch bestimmt werden. Auch bringt die Verwendung des magischen Dreiecks nicht immer signifikanten Nutzen und hängt vom spezifischen Optimierungsproblem ab. Ob sich der rechnerische Mehraufwand lohnt muss daher durch Tests herausgefunden werden. In DAO wurde die Möglichkeit der Nutzung des magischen Dreiecks mit eingebaut.

2.5 Der Simulated Annealing Algorithmus von DAO

Der DAO SA Algorithmus arbeitet nach dem Schema aus Listing 2.4.2 und ist vollständig in der `DAO` Klasse enthalten. Die Funktionsweise und wichtige Details werden in den folgenden Abschnitten behandelt.

2.5.1 Parameterliste

Das Herzstück stochastischer Optimierungsmethoden ist die zufällige Änderung des Systemzustandes. Erreicht wird das durch das zufällige Ändern von Entscheidungsvariablen bzw. Parametern. Welche Parameter wie und wann geändert werden hängen vom Algorithmus und dem zu optimierenden System ab. Bei DAO sollen immer nur einzelne Parameter modifizert werden. Die Auswahl des zu ändernden Parameters erfolgt dabei ebenfalls zufällig. Damit diese zufällige Parameterselektion programmatisch umgesetzt werden kann ist eine Parameterliste erforderlich. Diese wird als ein fünf spaltiges 2D-Array (`ParaList`) angelegt und enthält so viele Einträge (Zeilen) wie dem Wert `n_para` (vgl. Abschnitt 2.3.6, Seite 29) entspricht. Prinzipiell ist `n_para` die Anzahl der Parameter (in DAO also die Summe der Anzahl aller Leafs und Segmentgewichte), jedoch kann (per Optimierungsparameter) die Anzahl der Parameter für Segmentgewichte vervielfacht werden, so dass diese in der Liste mehrfach auftreten. Dadurch erreicht man eine höhere Auswahlwahrscheinlichkeit eben dieser Gewichtparamter, die im Vergleich zu den einzelnen Leafs in ihrerer Anzahl sehr gering sind. Die Parameterliste wird zu Beginn der Optimierung segmentweise der Reihe nach gefüllt und behält ihren Inhalt bis zum Schluss bei. In den fünf Spalten des `ParaList` Arrays werden alle zur Identifikation des Parameters nötigen Informationen hinterlegt: Parametertyp (Leaf/Gewicht), Beam, Segment und bei Leafs zusätzlich die MLC-ID (Zeile der Bixelmatrix) und der Leaftyp (links/rechts). Für die Auswahl des zu modifizierenden Parameters sollte ein möglichst gleichverteilter Zufallsgenerator verwendet werden um die Bevorzugung bzw. Vernachlässigung einzelner Parameter zu vermeiden (Details siehe Anhang A.2, Seite 84). Verwendet werde dazu Methoden der `DAO_Rand` Klasse.

2.5.2 Implementierung

Das Grundschema der SA Optimierung sieht nach der zufälligen Änderung des Systemzustandes die Absenkung der Systemtemperatur vor. Bei DAO wird der Systemzustand erst dann als vollständig geändert betrachtet, wenn für jeden Parameter genau ein Modifikationsversuch erfolgt ist. Wegen der zufallsabhängigen und niemals exakt gleichverteilten Auswahl der zu modifizierenden Parameter kann (bei insgesamt `n_para` Parametern) jedoch nicht sichergestellt werden, dass dies nach `n_para` vielen Parameterselektionen auch passiert. Durch das zeilenweise Abarbeiten einer zufällig durchmischten Parameterliste kann dieses Problem umgangen werden, bietet jedoch keine nennenswerten Vorteile. Zudem können bei einfach implementierten Listendurchmischungen ungewollte systematische oder unvorteilhaft verteilte Parameterselektionen erzeugt werden. Deshalb bleibt bei DAO der Listeninhalt statisch und die Parameterauswahl wird vollständig dem Zufall überlassen. Die Änderung des Systemzustandes ist also nach `n_para` vielen Parameterselektionen abgeschlossen, die programmatisch in einer Schleife abgearbeitet werden (Parameterschleife). Da ein Parameter direkt nach dessen Selektion zufällig geändert wird, muss direkt danach die Verifikation des so modifizierten Systemzustandes erfolgen und ggf. wieder verworfen werden. Natürlich muss vor der Anpassung des Systemzustandes ermittelt werden, ob die Änderung des Parameters überhaupt zulässig ist (MLC/Leaf-Constraints). Ist das nicht der Fall, so werden nachfolgende Punkte übersprungen und der nächste Parameter (zufällig) gewählt. Weil im Falle der Akzeptanz die Aktualisierung (und evtl. die Rücksetzung) der Dosismatrix erforderlich ist, zeichnet sich hier der potentiell hohe Rechenaufwand des DAO Verfahrens ab. Verstärkt wird dieser noch weiter, indem die Parameterschleife, die eine einzelne Iteration der SA Optimierung darstellt, vor der Temperaturabsenkung nochmals über eine höherrangige Schleife (Iterationsschleife) wiederholt wird. Dadurch wird die mehrfache Modifikation des Systemzustandes bei konstanter Temperatur ermöglicht, was die Wahrscheinlichkeit für das Auffinden einer guten Lösung des Optimierungsproblems erhöht. Das ist vor allem im Temperaturbereich des magischen Dreiecks wichtig, innerhalb dessen die Anzahl der Durchläufe der Iterationsschleife besonders hoch angesetzt werden sollte. Nachdem auch die Iterationsschleife

durchlaufen wurde ist ein Loop der DAO SA Optimierung abgeschlossen. Vor dem Start des nächsten Loops werden die Ergebnisse und protokollierten Daten des letzten Loops ausgewertet und auf dem Bildschirm, sowie ggf. in Protokolldateien ausgegeben. Es folgt schließlich die Temperaturabsenkung, welche nach dem exponentiellen Schema implementiert wurde, sofern keine der Abbruchbedingungen erfüllt wurde. Weitere Details sind dem Flussdiagramm (Abbildung 2.17) zu entnehmen. Nach Abschluss der Optimierung werden Methoden der Klasse `DAO_Output` aufgerufen um das Ergebnis in speziellen, zur Qualifizierung geeigneten Dateiformaten zu exportieren.

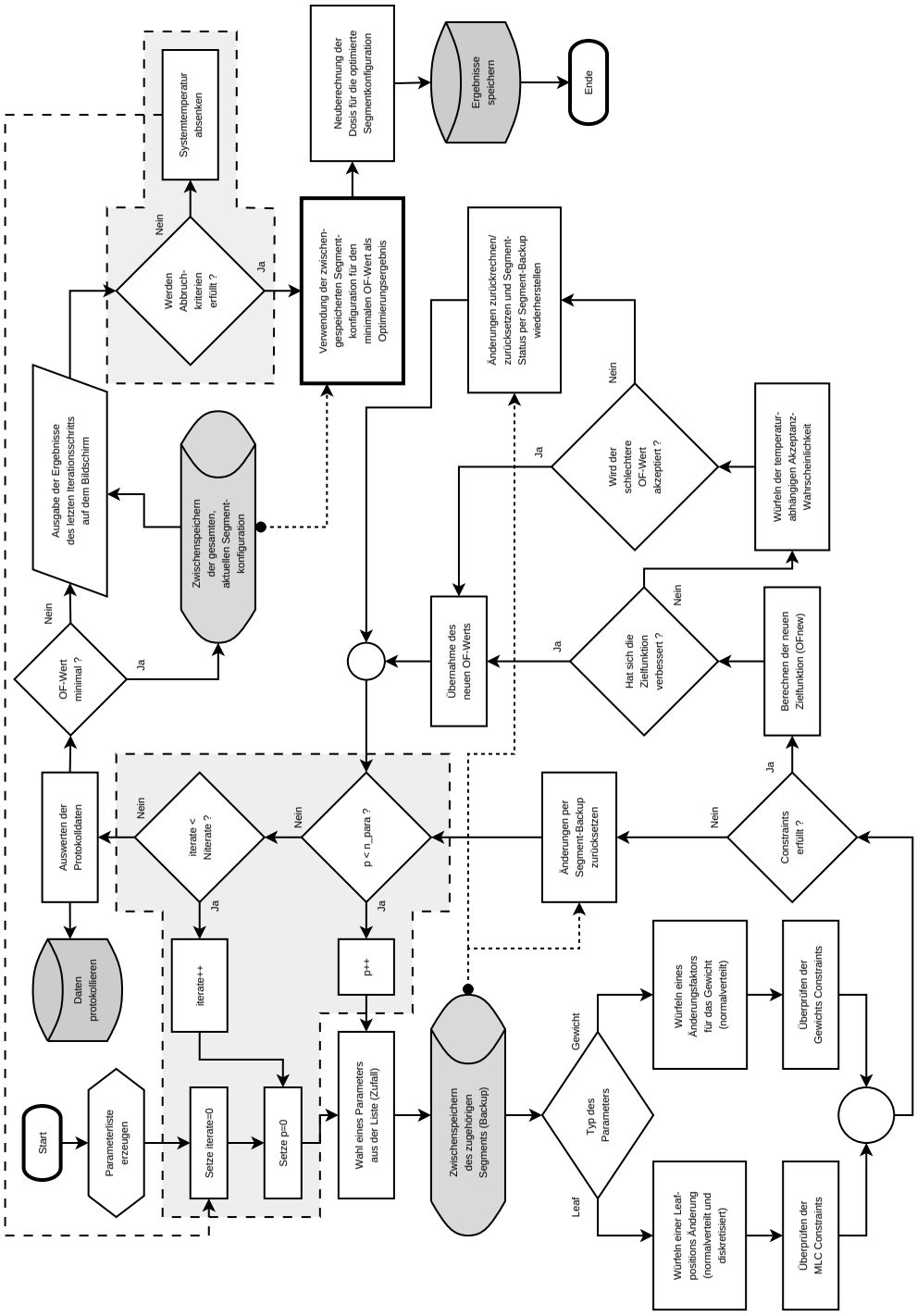

Abbildung 2.17: Flussdiagramm des DAO Simulated Annealing Algorithmus; die durch graue Boxen und gestrichelte Linien hervorgehobenen Bereiche fassen die dem Loop untergeordnete Iterations- und Parameterschleife zusammen; Gepunktete Linien stellen Zugriffe auf temporäre Datenspeicher dar

2.5.3 Parametermodifikation

Die Art und Weise der Modifizierung selektierter Parameter hängt von dessen Typ ab (Leaf oder Segmentgewicht). Beiden gleich ist aber das Prinzip, dass den Parametern kein zufälliger Wert zugewiesen wird, sondern die momentanen Werte zufällig abgeändert werden. Es werden dazu jeweils Gaußverteilungen verwendet, die mit der Polarmethode (siehe Anhang A.2.1.5, Seite 87) durch einen gleichverteilten Zufallszahlengenerator simuliert werden. Die gaußverteilten Parameteränderungen sorgen bei kleinen Standardabweichungen für eine Glättung der Aperturen und ermöglichen allgemein sprunghafte, aber entsprechend unwahrscheinliche Änderungen.

Modifikation der Leafpositionen

Da für Leafpositionen wegen des Bixelmatrix-Modells nur bestimmte diskrete Werte in Frage kommen, muss die gewürfelte Änderung diskretisiert werden. Abbildung 2.18 veranschaulicht diese Problematik. Naheliegend ist hier die Rundung auf ganze Zahlen

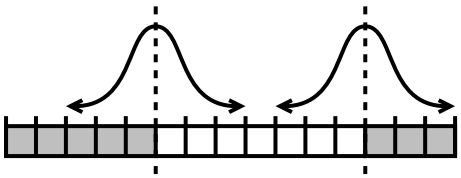

Abbildung 2.18: Die gaußverteilten Leafänderungen erfordern eine Diskretisierung

mittels (2.29).

$$\hat{x} = Round(x) = \lfloor x + 0.5 \rfloor \tag{2.29}$$

Da sich jedes Leaf nach links und auch nach rechts bewegen kann, ist für die Gaußverteilung ein Erwartungswert von 0 erforderlich. Durch die Rundungsmethode kann die Diskretisierung immer auch auf den Wert 0 führen, was gar keiner Leafänderung entspricht. Dieser Fall tritt genau dann ein, wenn die entsprechende Zufallsvariable einen Wert im Intervall $[-0.5, 0.5)$ annimmt. Wegen den Gaußklammern (in C++ `floor()`) ist dieses Intervall rechts offen. Da die Standardabweichung σ der Gaußverteilung die Wahrscheinlichkeit dafür beeinflusst, dass dieser Fall eintritt, hat dieser Parameter einen sehr

hohen Einfluss auf den Verlauf der Optimierung und das Ergebnis. Für die Änderung der Leafposition L gilt schließlich (2.30),

$$L_{neu} = L + \hat{x} \tag{2.30}$$

wobei x die Realisierung der $N(0, \sigma)$ Verteilten Zufallsvariablen ist und gemäß (2.29) auf $\hat{x}$ diskretisiert wurde. Solche Werte $\hat{x}$ werden in DAO mit einer Instanz der Klasse `DAO_Rand` erzeugt. Die Leafposition wird dann im `MLCdata` Array des zugehörigen Segment-Objekts aktualisiert und die Bixelmatrix (`BIXELon`) entsprechend angepasst.

Modifikation der Segmentgewichte

Die Modifikation der Segmentgewichte erfolgt ebenfalls durch gaußverteilte Zufallszahlen. Jedoch wird nicht wie bei den Leafs addiert, sondern das aktuelle Segmentgewicht mit dem so zufällig bestimmten Faktor multipliziert. Der Erwartungswert der zugehörigen Gaußverteilung wurde daher auf das neutrale Element 1 der Multiplikation gesetzt. Da negative Segmentgewichte negativen Bestrahlungszeiten entsprechen würden, dürfen nur positive Faktoren akzeptiert werden (Abbildung 2.19). Über Optimierungsparameter

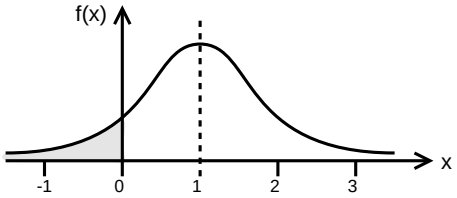

Abbildung 2.19: Dichte der Gaußverteilung für Gewichtänderungen; negative Werte sind ungültig

kann außerdem die 0 ausgeschlossen und eine minimale Auflösung $0 < \delta < 1$ der Segmentgewichte festgelegt werden. Das Ausschließen der 0 ist erforderlich, wenn die Verwendung aller Segmente erzwungen werden soll, was eine maximale Genauigkeit ermöglicht. Die Festlegung einer Auflösung für Segmentgewichte ist notwendig, da ein Linearbeschleuniger nicht für beliebig kurze Zeiten bestrahlen kann. Das δ gibt also gleichzeitig das minmal mögliche Segmentgewicht (bei Ausschließung der 0) und die minimale Differenz

zwischen zwei verschiedenen Segmentgewichten an. Die Anpassung der Gewichte an diese Auflösung entspricht wieder einer Diskretisierung, die gemäß Gleichung (2.31) jedoch erst bei der Exportierung der VMC Ergebnisdatei durchgeführt wird.

$$\hat{W} = Round(W/\delta) \cdot \delta = \lfloor \frac{W}{\delta} \rfloor \cdot \delta \tag{2.31}$$

Die Anpassung des Segment-Objekts während der Optimierung erfolgt bei Gewichtsmodifikation im Normalfall durch einfaches multiplizieren (2.32) des alten Gewichts W mit dem Modifikationsfaktor y.

$$W_{neu} = \begin{cases} 0 & \text{falls } y \leq 0 \text{ und } W_{neu} = 0 \text{ zugelassen} \\ W \cdot y & \text{falls } W \cdot y > \delta \\ \delta & \text{sonst} \end{cases} \tag{2.32}$$

Die Modifkationsfaktoren werden ebenfalls über Funktionen der Klasse `DAO_Rand` generiert.

2.5.4 Abbruchbedingungen

In früheren Studien [7] konnte gezeigt werden, dass mit der Simulated Annealing Optimierung das globale Minimum eines Systems theoretisch immer gefunden werden kann, jedoch nur in unendlich langer Zeit. Um die praktische Anwendung zu ermöglichen muss der Algorithmus vorzeitig abbrechen, wobei dennoch ein dem Optimum möglichst nahe liegendes Ergebnis erwünscht wird. Da sich aber nicht feststellen lässt ob das gefundene Minimum bereits global ist oder wie nahe man sich an diesem befindet, müssen vernünftige Abbruchkriterien formuliert werden die eine signifikante Verbesserung durch weitere Iterationen unwahrscheinlich machen. Der DAO SA Algorithmus wird beendet, wenn einer der vier folgenden Fälle eintritt:

- Unterschreitung einer minimalen Systemtemperatur
- Überschreitung einer maximalen Anzahl an Loops
- Keine Verbesserung während der letzten N Loops

- Keine *signifikante* Verbesserung während der letzten M Loops

Wegen dem festgelegten exponentiellen Abkühlschema sind die ersten beiden Abbruchkriterien im Grunde äquivalent, da sich zu jedem Iterationsschritt die Systemtemperatur berechnen lässt und umgekehrt. Dazu ist allerdings die Startemperatur erforderlich, die erst bei der SA Initialisierung bestimmt wird und vorher nicht bekannt ist. Aus diesem Grund ist die Temperaturbedingung nur dann relevant, wenn eine feste Starttemperatur vorgegeben wird. Im Normalfall wird die Laufzeit allein durch die Vorgabe einer maximalen Anzahl an Loops nach oben beschränkt. Das dritte Kriterium verlangt, dass eine Verbesserung des bisher besten Systemzustandes (`OFmin`) spätestens nach N aufeinanderfolgenden Loops erfolgen muss. Dadurch kann der Algorithmus bei sehr langsamer Konvergenz, die aufgrund einer zu hohen Starttemperatur oder zu geringer Temperaturabsenkung auftreten kann, vorzeitig abgebrochen werden. In diesem Fall müssen andere temperaturspezifische Optimierungsparameter gewählt und die Optimierung neu gestartet werden. Die vierte Abbruchbedingung ist die wichtigste von allen. Eine SA Optimierung sollte immer aufgrund dieses Kriteriums beendet werden. Das passiert genau dann, wenn bei M aufeinanderfolgenden Loops keine signifikante Verbesserung der Energie (bzw. des Zielfunktionswertes) erreicht wurde oder sich diese gar nicht mehr ändert. Als signifikant kann eine Verbesserung angesehen werden, wenn sich die Systemenergie dadurch um mindestens $\lambda\%$ (bezogen auf die Anfangsenergie) reduziert. Loops bei denen Verschlechterungen unterhalb dieser Toleranz auftreten werden ignoriert. Wirkt sich eine Verschlechterung jedoch stärker als $\lambda\%$ aus, so wird der Abbruchzähler auf 0 zurückgesetzt, so dass weitere M nicht signifikante Verbesserungen eintreten müssen um die Optimierung zu beenden. Durch dieses Kriterium wird ein vorzeitiges Abbrechen der Optimierung bei nur noch sehr geringen Änderungen der Energie ermöglicht. Das ist sinnvoll, da geringe Energievarianzen mit niedrigen Temperaturen einhergehen und die Konvergenz gegen ein lokales Minimum angenommen werden kann. Die prozentuale Toleranzgrenze λ wird über einen Optimierungsparameter gesetzt und empirischen Daten entsprechend sinnvoll gewählt.

2.5.5 Nebenbedingungen

Es ist klar, dass die zufälligen Parametermodifikationen durch den SA Algorithmus nicht beliebig ausfallen können und gewissen Einschränkungen unterliegen. Diese bilden die Nebenbedingungen (Constraints) des Optimierungsproblems. Entsprechend der verschiedenen Parametertypen lassen sich in DAO MLC und Gewicht Constraints unterscheiden. Das Einhalten dieser Nebenbedingungen ist für die praktische Umsetzbarkeit des optimierten Plans verantwortlich, weshalb diese als Hard-Constraints formuliert werden.

MLC Constraints

Die Bauweise der Multi Leaf Collminatoren weicht von Beschleuniger zu Beschleuniger ab. Dabei können sich sowohl die Dimensionen der Leafs, als auch die möglichen Positionierungsmöglichkeiten unterscheiden. Es ist deshalb notwendig Constraints zu formulieren die dafür sorgen, dass alle durch den Optimierungsalgorithmus bestimmten Aperturen vom Bestrahlungsgerät später auch realisiert werden können. Die in DAO verwendeten MLC Constraints sind in der Klasse `MLCcon` implementiert, welche auch über Methoden zum Einlesen der MLC-Constraint Datei (`MLCcon.dao`) verfügt. Damit können alle MLC Constraints in Abhängigkeit des Beschleunigertyps konfiguriert werden. Tabelle A.6 (Anhang, Seite 113) listet die in DAO realisierten Constraints auf, welche für den *PRIMUS* Beschleuniger (Siemens) formuliert werden mussten. Für andere Beschleuniger waren im Rahmen dieser Arbeit keine Untersuchungen vorgesehen. Im Anhang findet sich auch der Auszug eines vollständigen Konfigurationssatzes aus der MLC-Constraint Datei.

Gewicht Constraints

Auch bei den Segmentgewichten müssen Einschränkungen getroffen werden. Zum einen kann aufgrund technischer Bedingungen nicht für beliebig kurze Zeiten bestrahlt werden, zum anderen machen beliebig große Gewichte keinen Sinn. Auch wenn Segmentgewichte theoretisch noch oben unbeschränkt sind, können bei hohen Bestrahlungszeiten sogenannte „Hot Spots“ (lokale Hochdosisbereiche innerhalb der Außenkontur) auf. Diese

sind ungewollt und können durch die Beschränkung des Gesamtgewichts jeder Einstrahlrichtung verhindert werden. Um einen etwa gleichmäßig verteilten Anteil aller Segmente einer Einstrahlrichtung an diesem Gesamtgewicht zu ermöglichen, können die Gewichte einzelner Segmente zusätzlich nach oben beschränkt werden. Deshalb wird für Segmente ebenfalls ein Maximalgewicht vergeben. Da bis auf die minimal mögliche Bestrahlungsdauer keiner dieser Constraints vom Beschleunigertyp abhängt, werden die entsprechenden Parameter in den DAO Konfigurationsdateien hinterlegt, die jeweils einen Satz an Optimierungsparametern enthalten (z.B. `Std_Prostata.dao`). Die Gewicht Constraints werden jedoch zu den Planparametern gezählt (siehe Anhang Tabelle A.7 und Listing A.10, Seite 113). Weil alle Untersuchungen auf den *PRIMUS* Beschleuniger beschränkt wurden, erfolgte eine Auslagerung des Parameters zur Festlegung der minimalen Bestrahlungszeit in eine entsprechende Konfigurationsdatei bislang nicht.

2.5.6 Optimierungsparameter

SA Algorithmen sind stark parameterabhängig. Das heißt, der Verlauf der Optimierung wird durch einige Parameterwerte direkt beeinflusst. Im Falle von DAO werden fast zwanzig verschiedene Optimierungsparameter verwendet (siehe Anhang A.11, Seite 114). Bei einigen dieser Parameter ist jedoch nicht vorhersehbar, welchen Einfluss sie auf den Verlauf der Optimierung ausüben. Es ist deshalb notwendig die Abhängigkeit des Algorithmus von den Optimierungsparametern durch Tests (siehe Abschnitt 3.1) empirisch zu bestimmen und ggf. optimale Werte zu ermitteln. Weil solche Tests mit realen oder speziellen Bestrahlungsplänen durchgeführt werden müssen, können die so ermittelten Werte von der Artung des zu optimierenden Systems abhängen. Aus diesem Grund erfolgt die Speicherung der Optimierungsparameter in verschiedenen, systemspezifischen Dateien (z.B. `Std_Prostata.dao` für allgemeine Prostatafälle). Vor der Optimierung muss die zu verwendende Datei deshalb angegeben werden.

3 Ergebnisse

3.1 Bestimmung geeigneter Optimierungsparameter

Da bei einigen Optimierungsparametern nicht klar ist welche Werte optimal sind und wie sich diese auf das Optimierungsproblem auswirken, mussten Tests zur Ermittlung dieser Parameterabhängigkeiten erfolgen. Dabei wurde ein konventioneller Prostatafall mit gleich bleibenden DV-Constraints und Penalties verwendet, was für identische Ausgangsbedingungen eines jeden Testlaufs sorgt. Von einer als Ausgangsbasis festgelegten Parameterkonfiguration wurde pro Test nur ein einziger Parameter abgeändert um dessen Einfluss sicher bestimmen und vergleichen zu können. Folgende Parameter wurden dabei untersucht:

- Wiederholungen der Iterationsschleife (`SA_LOOP_ITERATE` =: $N_{iterate}$)
- Startakzeptanzrate (`SA_START_TEMP_LOG` =: P_{acc})
- Gewichtparameter Vervielfachung (`WEIGHT_MOD_RATE` =: N_{weight})
- Abkühlrate (`SA_COOL_EXPO` =: α_{cool})
- Sigma für Leafänderungen (`SA_SIGMA_LEAF` =: σ_{Leaf})
- Sigma für Gewichtänderungen (`SA_SIGMA_WEIGHT` =: σ_{Weight})

Für die beiden Sigmas wurden ausgiebigere Tests durchgeführt (siehe Abschnitt 3.1.1 und 3.1.2), bei allen anderen erfolgte eine Optimierung für jeweils zwei weitere, verschiedene Parameterwerte. Welcher Parameterwert sich besser eignet als ein anderer wurde anhand der erreichten Zielfunktionsminimierung (Verbesserungsfaktor in Prozent) und

der benötigten Zeit für 200 Loops bestimmt. Da für diese Untersuchungen eine recht grobe Voxelauflösung und pro Einstrahlrichtung nur wenige Segmente (vier) gewählt werden konnten, waren diese Zeiten sehr kurz. Die getesteten Werte und Ergebnisse sind in Tabelle 3.1 aufgeführt. Die zweite Spalte enthält die Ausgangskonfiguration, die erste Zeile das Optimierungsergebnis für diese Anfangskonfiguration.

Parameter	**Std**	**Testwert**	**OF Verbesserung**	**Zeit**
-	-	-	53.6 %	23m 19s
$N_{iterate}$	4	1	59.7 %	4m 6s
$N_{iterate}$	4	2	55.9 %	7m 18s
P_{acc}	0.5	0.9	54.5 %	13m 5s
P_{acc}	0.5	0.4	52.7 %	24m 5s
N_{weight}	2	1	52.6 %	17m 51s
N_{weight}	2	4	51.6 %	30m 53s
α_{cool}	0.95	0.9	56.7 %	18m 38s
α_{cool}	0.95	0.85	56,7 %	13m 10s

Tabelle 3.1: Ergebnisse der Optimierungsparametertests

3.1.1 Sigma-Leaf

Für das Sigma der Gaußverteilung für Leafmodifikationen wurden Werte zwischen 0.1 und 6.0 jeweils zwei mal getestet. Die Rechnungen wurden dazu für jeden Testwert genau einmal direkt aufeinanderfolgend durchgeführt und der gesamte Vorgang wiederholt. Die mittleren Ergebnisse sind in Abbildung 3.1 zu sehen. Um die Rechenzeiten in das gleiche Diagramm wie die Verbesserungsfaktoren einzeichnen zu können und weil nur deren Verhältnisse interessieren, wurden die Zeiten zweckmäßig normiert und als Zeitwerte bezeichnet.

3.1.2 Sigma-Weight

Wie bei dem Sigma für Leafmodifikationen wurde auch jenes für Gewichtmodifikationen mehrfach und auf die gleiche Weise getestet. Hier wurden jedoch nur Werte zwischen 0.1 und 1.2 verwendet und insgesamt vier Testläufe durchgeführt. Abbildung 3.2 zeigt die

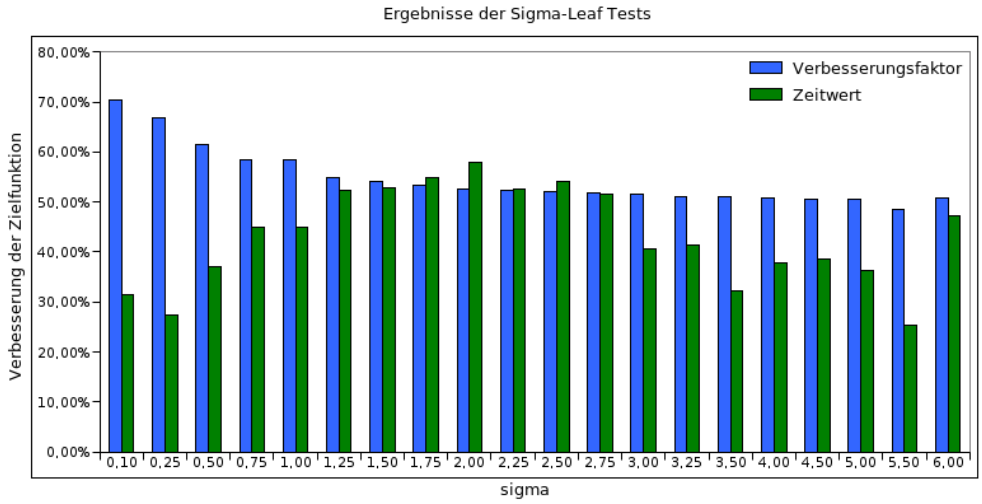

Abbildung 3.1: Ergebnisse der Sigma-Leaf Tests

mittleren Ergebnisse und Abbildung 3.3 die Einzelergebnisse der Zielfunktionsverbesserung.

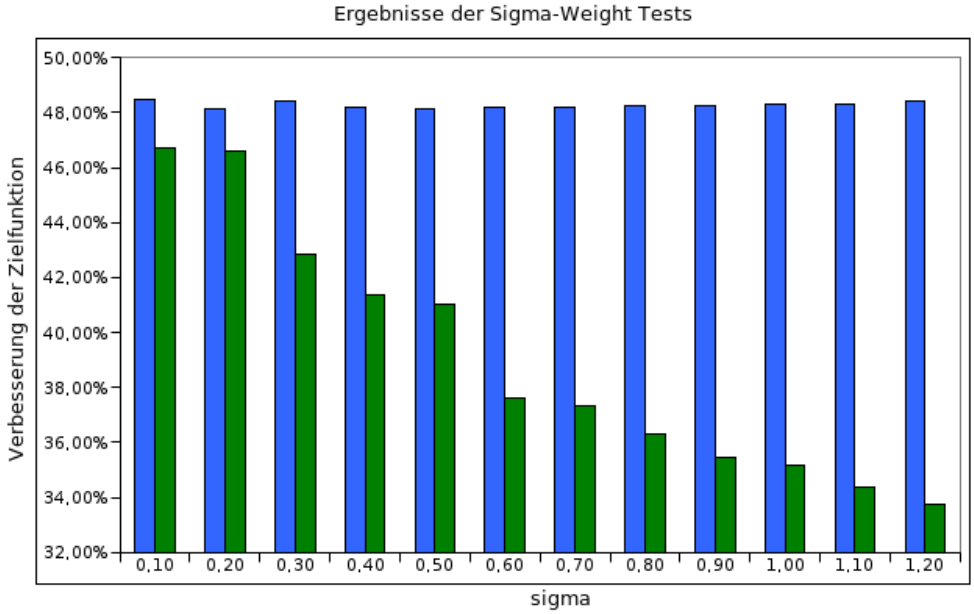

Abbildung 3.2: Ergebnisse der Sigma-Weight Tests

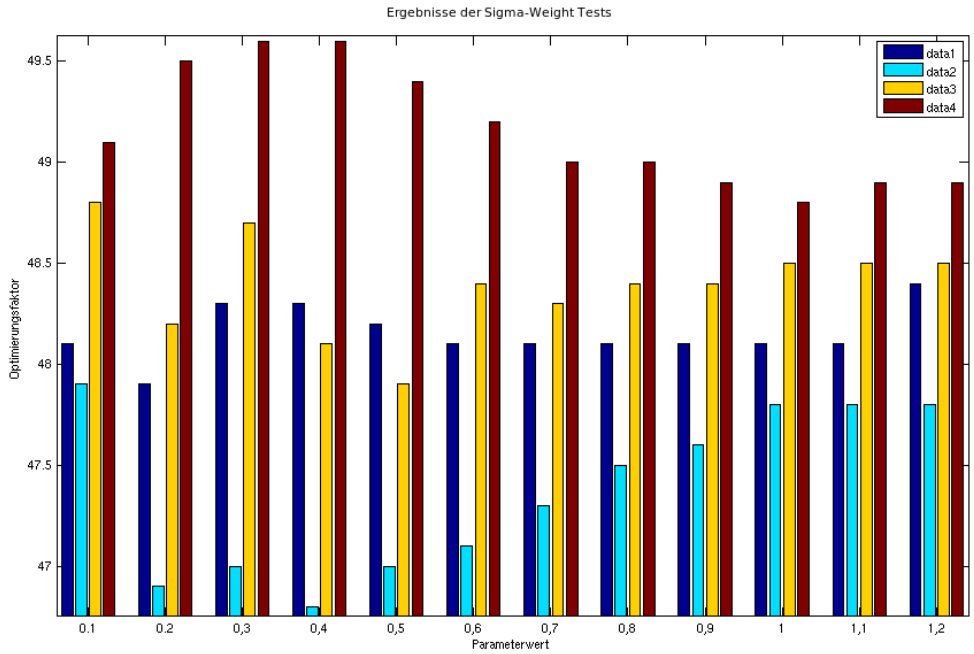

Abbildung 3.3: Einzelergebnisse der Verbesserungsfaktoren

3.2 Ergebnisse der DAO Optimierung im Vergleich mit IKO

Im Folgenden werden anhand zweier Prostatafälle und einem Quasimodo Phantom die Ergebnisse der DAO Optimierung mit IKO-Rechnungen verglichen. Da beide Optimierungsalgorithmen leicht unterschiedliche Zielfunktionen verwenden ist der Planvergleich nicht mit letzter Sicherheit exakt möglich. Denn die Penalties der Zielfunktionsteile für ROIs müssen auf den jeweiligen Algorithmus abgestimmt werden um gute Ergebnisse zu erhalten. Die DV-Constraints und Planparameter wurden jedoch in beiden Fällen gleichgesetzt, so dass ein Einfluss auf die Berechnungen nur noch über Penaltyfaktoren erfolgen kann. Das Ergebnis jeder IKO-Rechnung wurde mit der Optimierungsoption "Steepest-descent" (Gradientenabstiegsverfahren) durchgeführt, anschließend mit ImFast® segmentiert (Einstellungen siehe Tabelle 3.2) und dann reoptimiert. Für die DAO-Rechnung wurde in allen drei Fällen die Konfigurationsdatei `Std_Prostata.dao` mit den in Listing A.11 (Anhang, Seite 114) aufgeführten Parameterwerten und die MLC-Constraints für den *PRIMUS* Beschleuniger benutzt. Die Ergebnisse wurden mit XVMC nochmals vorwärtsgerechnet. Das magische Dreieck wurde nicht verwendet, da sich dessen Verlauf nicht klar als Dreieck darstellt und in vorangehenden Untersuchungen keine Verbesserung des Ergebnisses erkannt werden konnte.

Methode	Plattform optimal
Flussberechnung	Primär- und Streu- und Leckstrahlung
Korrekturen	Nut und Feder; Fluss
Flussfehler	Mittleren Fehler reduzieren
Reduzierung	Normalisierung der Verteilung auf nicht mehr als 10 Stufen
Erweitert	*(Keine Einstellungen)*

Tabelle 3.2: Verwendete Einstellungen für die Segmentierung mit ImFast®

3.2.1 Prostatafall ohne Hüftknochen

Die folgenden Ergebnisse wurden anhand eines realen 6-Felder Prostataplans gewonnen, der aus einem PTV (Prostata) und zwei OARs (Harnblase, Rektum) besteht. Um das PTV wurde eine zusätzliche 3cm UT-Margin definiert, die sich nicht mit Blase und Rektum schneidet.

ROI	D_{min}	D_{max}	$(D_1\|V_1)$	$(D_2\|V_2)$	$(D_3\|V_3)$	Penalty IKO	Penalty DAO
PTV	0.98	1.07	-	-	-	3000	1000
Blase	-	-	(0.50|0.50)	(0.75|0.25)	(1.00|0.00)	500	500
Rektum	-	-	(0.50|0.50)	(0.75|0.25)	(1.00|0.00)	500	500
UT-Margin	-	-	(0.50|0.50)	(1.00|0.00)	-	100	100

Tabelle 3.3: Verwendete DV-Constraints und Penalties bei der Optimierung des Prostatafalls

Voxelauflösung (XY-Ebene)	128×128
XVMC Fehler	2 %
Einstrahlrichtungen	6: 120°, 90°, 25°, 240°, 270°, 335°
Segmente DAO	60 (10 pro Richtung)
Segmente IKO	59
Betrachtete Voxel	24347
Voxelgröße	$0.37 \times 0.37 \times 0.50$ cm^3
MLC-Constraints	PRIMUS, Standard
Gewicht-Constraints	Standard

Tabelle 3.4: Planparameter; eine Einstrahlrichtung von 0° entspricht dem Blick senkrecht von oben auf den liegenden Patienten

Startwert (Zielfunktion)	6397.73
Endwert (Zielfunktion)	1079.26
Zielfunktionsverbesserung	16.87 %
Rechenzeit	5h 51m 50s
Loops	278

Tabelle 3.5: Ergebnisse der DAO Optimierung

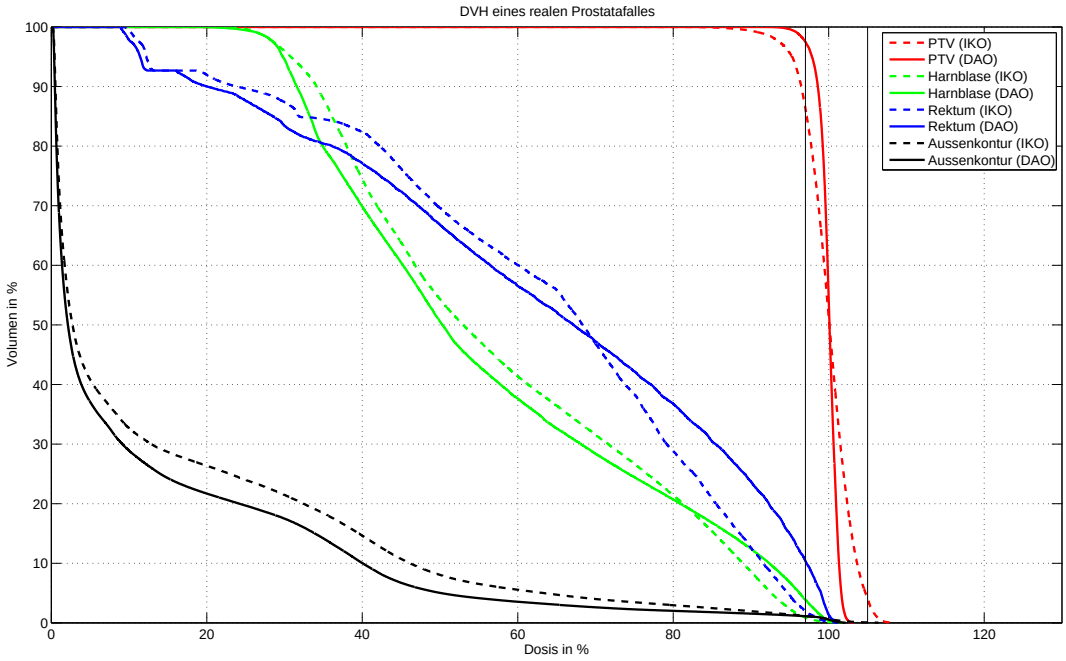

Abbildung 3.4: DVH Vergleich beider Optimierungsergebnisse (Normierung: Dosismittelwert im PTV)

	PTV			Blase	Rektum		Außenkontur
	H	D_1 (%)	D_{99} (%)	V_{50} (%)	V_{50} (%)	V_{90} (%)	V_{50} (%)
DAO	3.7	101.3	95.1	48.9	65.8	22.2	4.8
IKO	9.4	105.3	90.4	52.9	68.6	11.2	7.7

Tabelle 3.6: Vergleich einiger Kennwerte

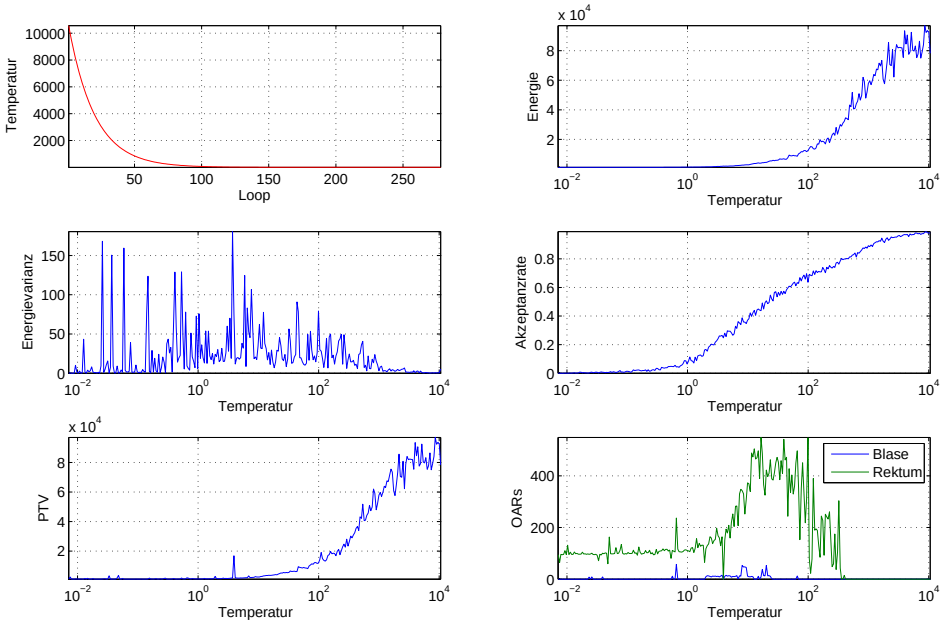

Abbildung 3.5: DAO Optimierungsverlauf

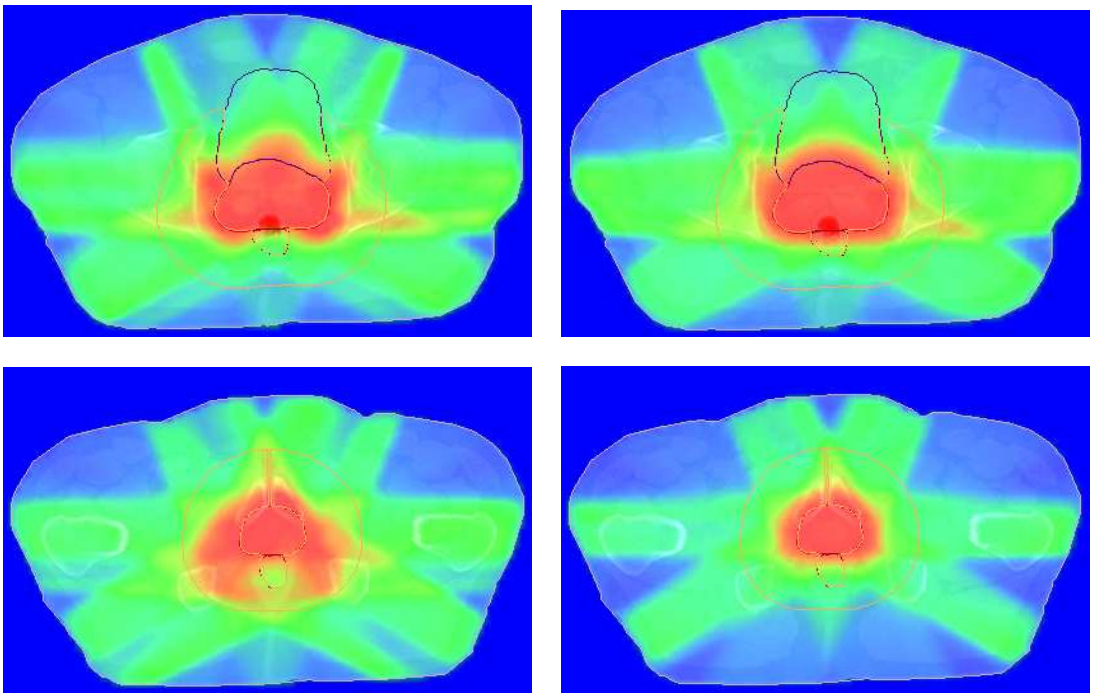

Abbildung 3.6: Vergleich von Schnittbildern; oben: Schicht 28, unten: Schicht 38; links: IKO, rechts: DAO

3.2.2 Prostatafall mit Hüftknochen

Das vorliegende Ergebnisse gehören zu einem realen 7-Felder Prostataplan, der als zusätzliche OARs den linken und rechten Hüftkopf enthält. Diese reagieren empfindlicher auf Strahlung als normales Körpergewebe und müssen bei Prostatabestrahlungen häufig besonders geschont werden.

ROI	D_{min}	D_{max}	$(D_1\|V_1)$	$(D_2\|V_2)$	$(D_3\|V_3)$	Penalty IKO	Penalty DAO
PTV	0.95	1.05	-	-	-	3000	3000
Blase	-	-	(0.50|0.50)	(0.75|0.25)	(1.00|0.00)	750	750
Rektum	-	-	(0.50|0.50)	(0.75|0.25)	(1.00|0.00)	1000	1000
Hüfte re.	-	-	(0.50|0.50)	(0.80|0.00)	-	500	500
Hüfte li.	-	-	(0.50|0.50)	(0.80|0.00)	-	500	500
UT-Margin	-	-	(0.50|0.50)	(1.00|0.00)	-	100	100

Tabelle 3.7: Verwendete DV-Constraints und Penalties bei der Optimierung des Prostatafalls mit Hüftknochen

Voxelauflösung (XY-Ebene)	128×128
XVMC Fehler	2 %
Einstrahlrichtungen	7: 0°, 52°, 103°, 155°, 206°, 258°, 309°
Segmente DAO	70 (10 pro Richtung)
Segmente IKO	84
Betrachtete Voxel	39031
Voxelgröße	$0.37 \times 0.37 \times 0.40$ cm^3
MLC-Constraints	PRIMUS, Standard
Gewicht-Constraints	Standard

Tabelle 3.8: Planparameter

Startwert (Zielfunktion)	122677
Endwert (Zielfunktion)	15437.8
Zielfunktionsverbesserung	12.6 %
Rechenzeit	12h 29m 2s
Loops	243

Tabelle 3.9: Ergebnisse der DAO Optimierung

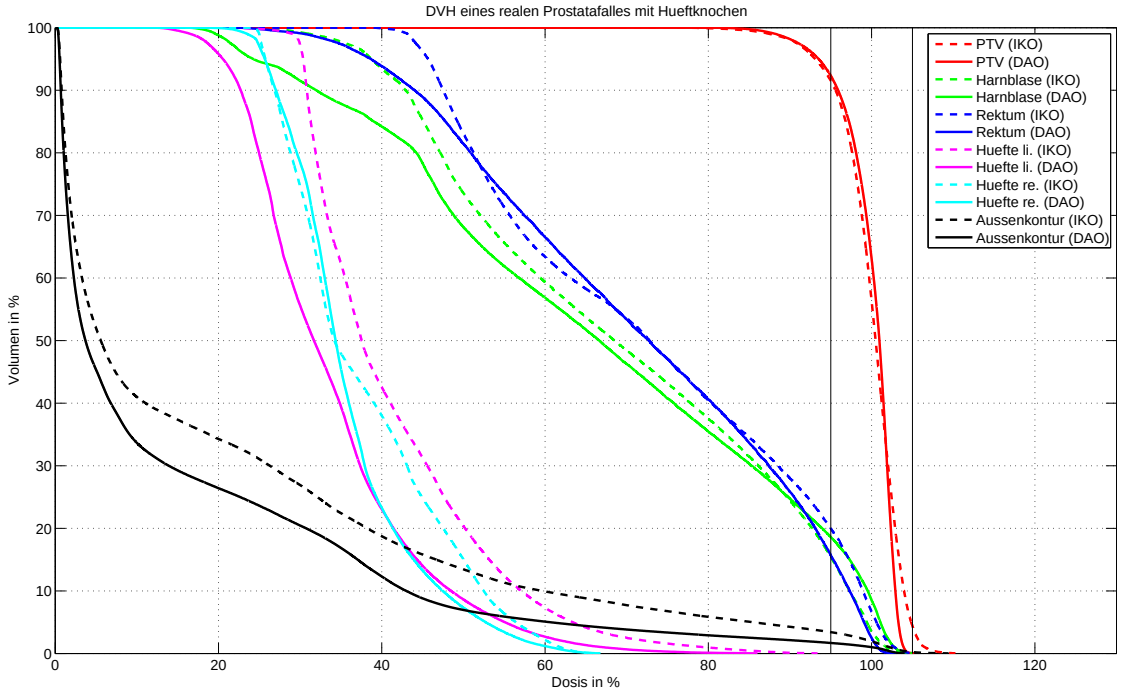

Abbildung 3.7: DVH Vergleich beider Optimierungsergebnisse (Normierung: Dosismittelwert im PTV)

	PTV			Blase	Rektum		Hüfte li.	Hüfte re.	Außenkontur
	H	D_1 (%)	D_{99} (%)	V_{50} (%)	V_{50} (%)	V_{90} (%)	V_{50} (%)	V_{50} (%)	V_{50} (%)
DAO	9.8	103.0	87.3	66.9	80.0	24.3	7.9	6.5	6.7
IKO	11.7	105.6	86.6	71.9	80.8	26.8	18.7	13.1	13.0

Tabelle 3.10: Vergleich einiger Kennwerte

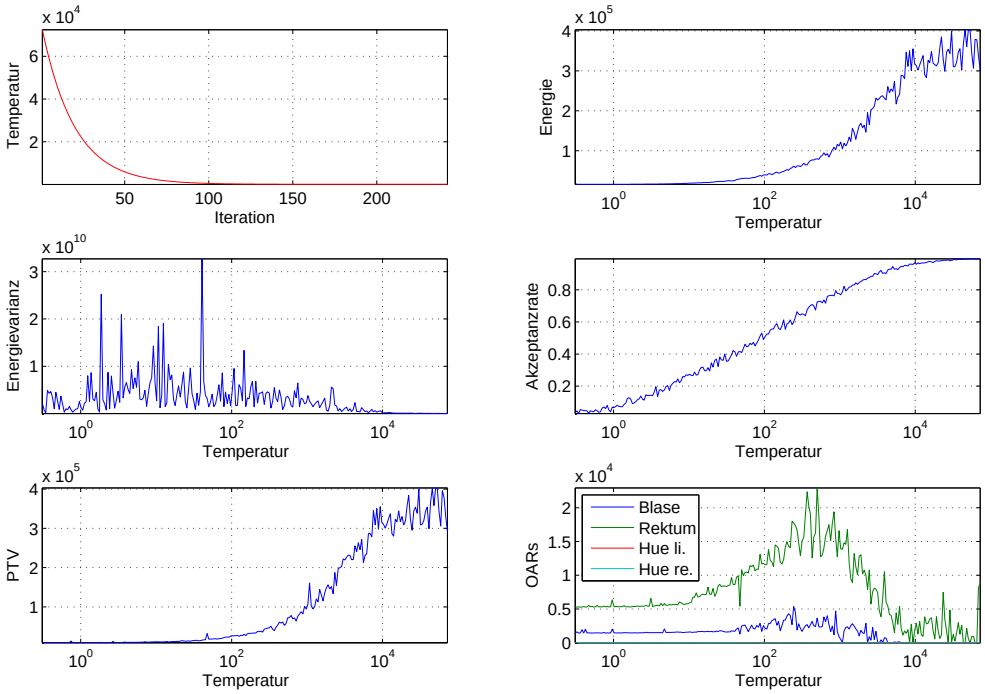

Abbildung 3.8: DAO Optimierungsverlauf

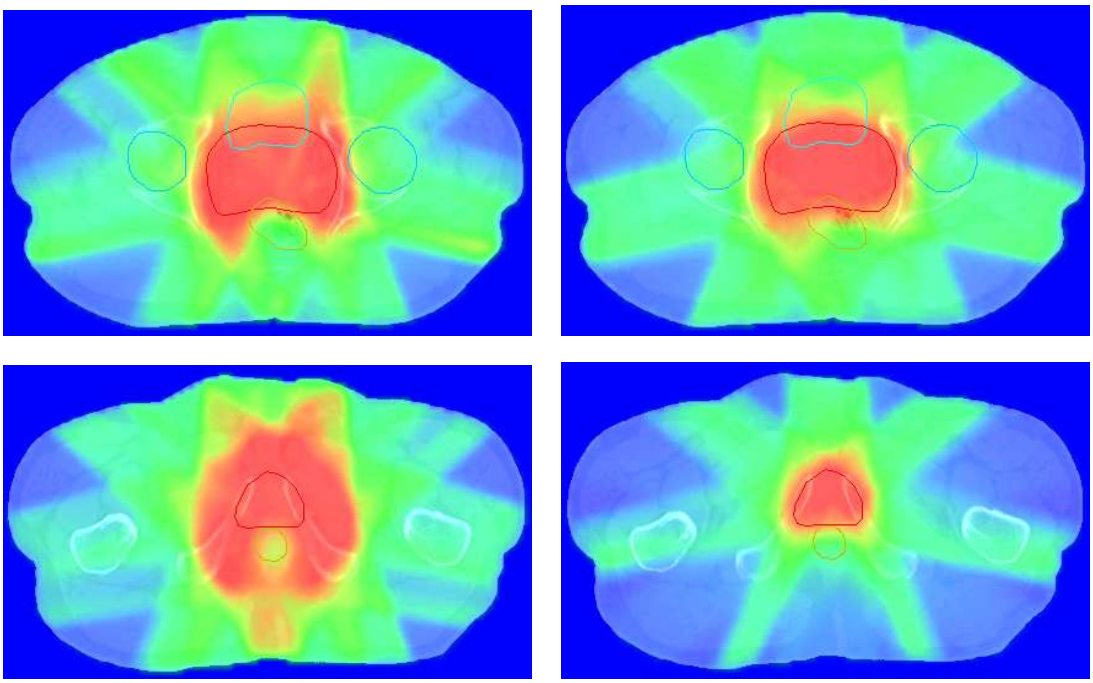

Abbildung 3.9: Vergleich von Schnittbildern; oben: Schicht 35, unten: Schicht 49; links: IKO, rechts: DAO

3.2.3 Quasimodo-Phantom

Das Quasimodo Phantom ist ein künstlicher Körper aus einem Material, welches sich gegenüber Strahlung wasserähnlich verhält und dadurch echtes Körpergewebe simulieren kann. Das PTV ist stark hufeiseförmig gebogen und enthält im Zentrum ein zu schondendes OAR. Die Bestrahlungsplanung ist bei solchen Strukturen besonders schwierig, weshalb das Quasimodo Phantom eine hohe Herausforderung für jedes IMRT Planungssystem darstellt.

ROI	D_{min}	D_{max}	$(D_1\|V_1)$	$(D_2\|V_2)$	$(D_3\|V_3)$	Penalty IKO	Penalty DAO
PTV	0.975	1.025	-	-	-	1000	1500
OAR	-	-	(0.66|0.40)	(0.70|0.00)	-	500	300
Hilfskontur	-	-	(0.66|0.40)	(0.70|0.00)	-	500	300
UT-Margin	-	-	(0.50|0.50)	-	-	500	100

Tabelle 3.11: Verwendete DV-Constraints und Penalties bei der Optimierung des Quasimodo Phantoms

Voxelauflösung (XY-Ebene)	64×64
XVMC Fehler	5 %
Einstrahlrichtungen	6: 30°, 90°, 150°, 210°, 270°, 330°
Segmente DAO	60 (10 pro Richtung)
Segmente IKO	62
Betrachtete Voxel	36138
Voxelgröße	$0.78 \times 0.78 \times 0.20$ cm^3
MLC-Constraints	PRIMUS, Standard
Gewicht-Constraints	Standard

Tabelle 3.12: Quasimodo Planparameter

Startwert (Zielfunktion)	74820.7
Endwert (Zielfunktion)	41661.5
Zielfunktionsverbesserung	55.68 %
Rechenzeit	15h 15m 38s
Loops	243

Tabelle 3.13: Ergebnisse der DAO Optimierung

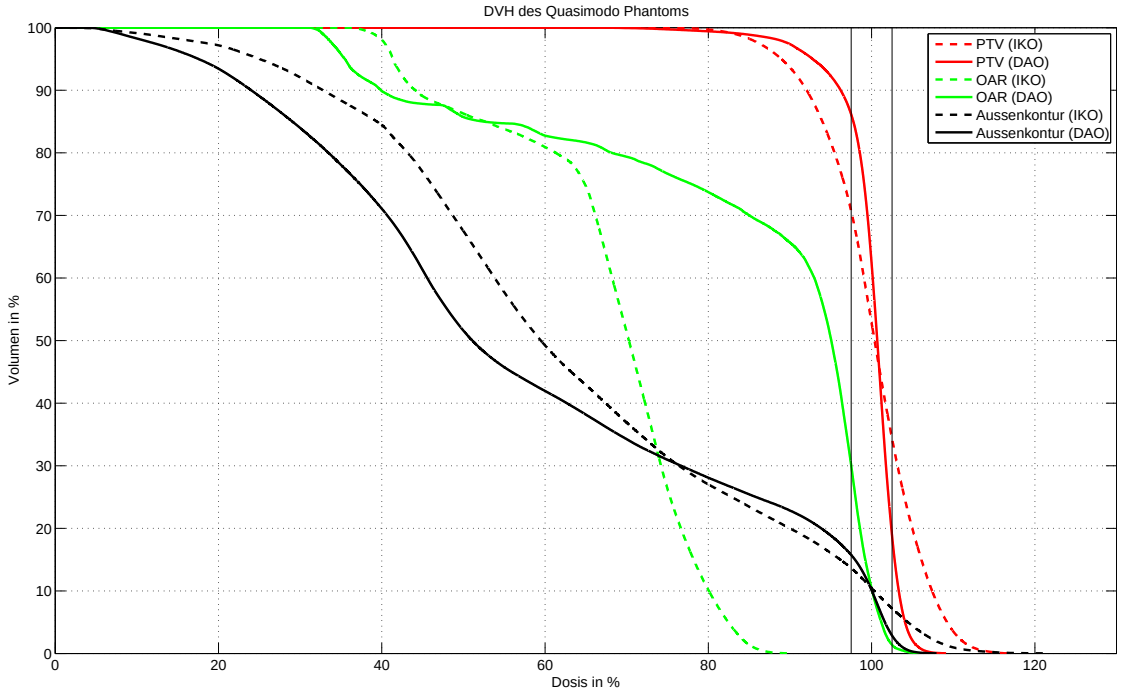

Abbildung 3.10: DVH Vergleich beider Optimierungsergebnisse (Normierung: Dosismittelwert im PTV)

	PTV			OAR		Außenkontur
	H	D_1 (%)	D_{99} (%)	V_{50} (%)	V_{90} (%)	V_{50} (%)
DAO	11.3	104.9	83.3	85.4	64.6	50.3
IKO	20.2	111.4	82.3	85.8	0.0	65.8

Tabelle 3.14: Vergleich einiger Kennwerte

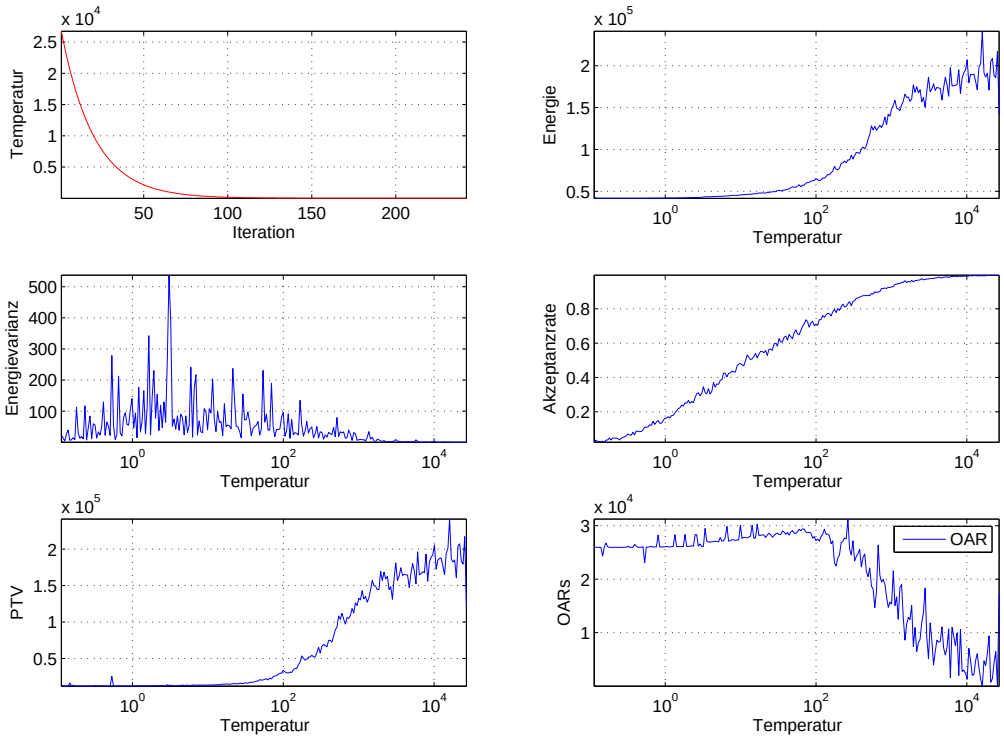

Abbildung 3.11: DAO Optimierungsverlauf

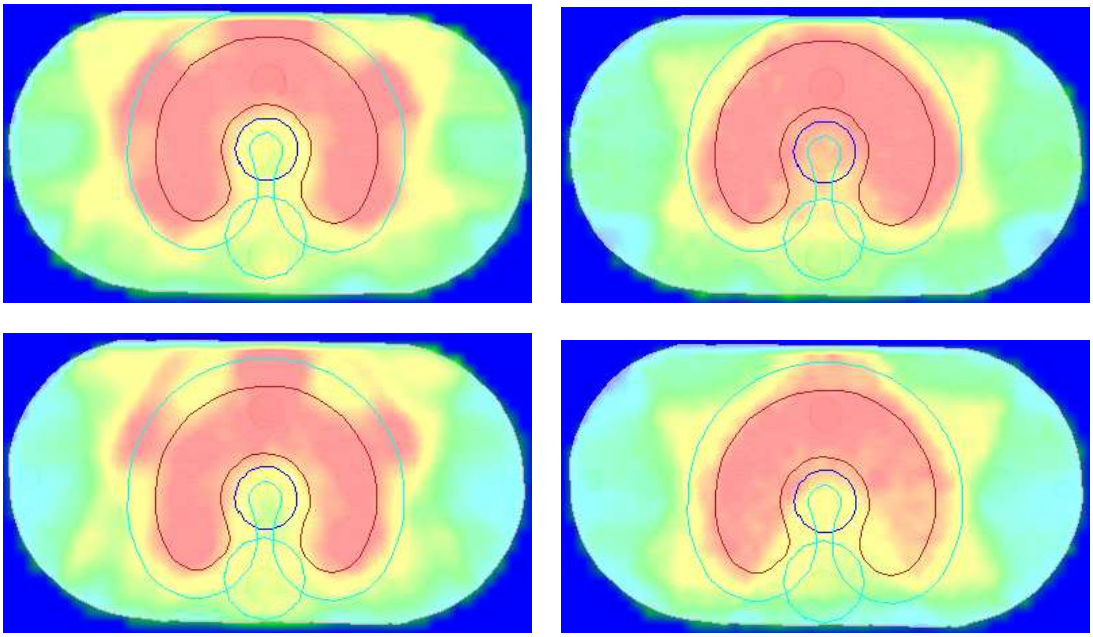

Abbildung 3.12: Vergleich von Schnittbildern; oben: Schicht 34, unten: Schicht 65; links: IKO, rechts: DAO

3.2.4 Vergleich von DAO mit DAO-XVMC Vorwärtsrechnung

Wegen Streueffekten an den Leafkanten ist die mit DAO optimierte Dosisverteilungen nie ganz exakt. Anhand des Ergebnisses des Prostatafalls (ohne Hüftknochen) wurde ermittelt, wie groß bei diesem die Unterschiede tatsächlich ausfallen.

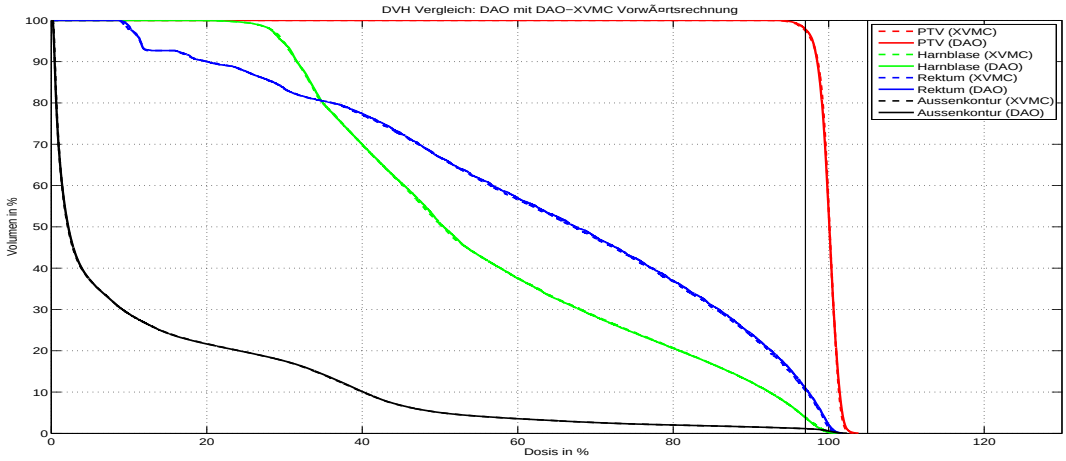

Abbildung 3.13: DVH Vergleich des DAO Ergebnisses mit der XVMC Vorwärtsrechnung

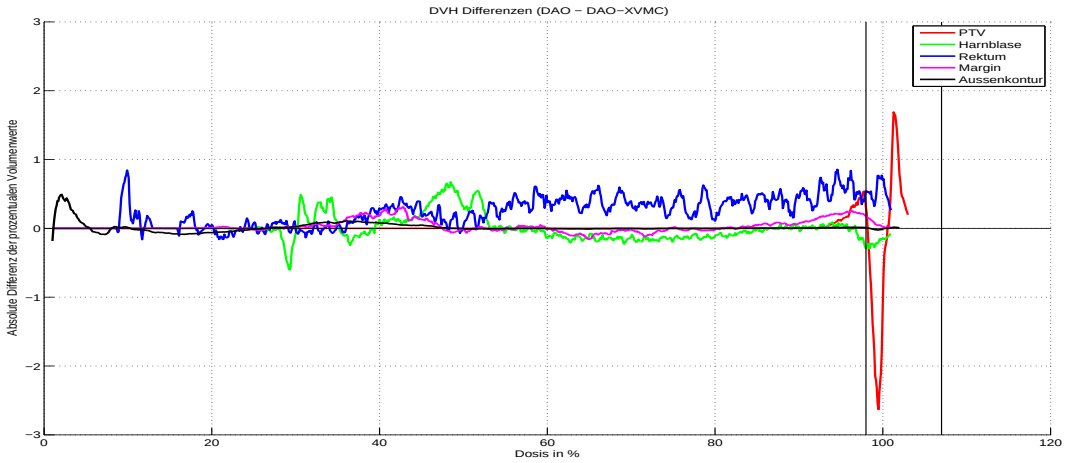

Abbildung 3.14: Absolute Differenzen der prozentualen Volumenwerte von DAO und DAO-XVMC

4 Diskussion

4.1 Optimierungsparameter

Bei der Verringerung der Iterationsschleifendurchläufe hat sich erwartungsgemäß eine signifikant kürzere Rechenzeit ergeben, da pro Loop weniger Berechnungen durchgeführt werden. Gleichzeitig hat sich jedoch eine Verschlechterung des Ergebnisses ergeben. Der Tendenz der Abhängigkeit des Ergebnisses von den Iterationswiederholungen lässt sich entnehmen, dass bei Wiederholungsraten größer 4 nur noch geringe Verbesserungen und viel höhere Rechenzeiten zu erwarten sind. Der Wert für $N_{iterate}$ wurde deshalb bislang bei 4 belassen.

Bei der Akeptanzrate P_{acc} zur Bestimmung der Starttemperatur hat sich gezeigt, dass größere Werte zwar kleinere Laufzeiten, dafür aber schlechtere Ergebnisse liefern. Das Verhalten der Laufzeiten ist darauf zurückzuführen, dass hohe Akzeptanzraten eine entsprechend geringe Anzahl an Rückrechnungen der Dosismatrix erfordern, was in der Summe sehr viel Zeit einspart. Das Verhalten der Zielfunktion ist dagegen überraschend, da gemäß der Fachliteratur hohe Starttemperaturen zu besseren Ergebnissen führen, was durch einen P_{acc} Wert nahe 1 realisiert werden kann. Ein Nebeneffekt erhöhter Temperatur ist allerdings eine langsamere Konvergenz, da diese erst ab einem bestimmten Temperaturbereich eintritt (magisches Dreieck). Für $P_{acc} = 0.9$ konnte dementsprechend eine Konvergenzverlangsamung festgestellt werden, so dass 200 Loops nicht für die vollständige Optimierung ausgereicht haben. Es ist daher zu erwarten, dass sich die Zielfunktion bei weiteren Iterationen noch entsprechend verbessert. Weitergehende Tests wurden hier nicht mehr durchgeführt, da bei folgenden Untersuchungen ein möglichst vollständiger Konvergenzverlauf vorliegen sollte um das Systemverhalten besser analysie-

ren zu können, weshalb ohnehin ein hoher Wert von 0.95 verwendet werden musste. Eine sinnvolle Optimierung des P_{acc} Parameters kann daher erst nach Abschluss zukünftiger Entwicklungen des DAO Verfahrens erfolgen. Es konnte jedoch bereits gezeigt werden, dass eine hohe Starttemperatur die Wahrscheinlichkeit besserer Ergebnisse begünstigt. Der Zeitgewinn wird aber aufgrund der langsamen Konvergenz, die einen Mehraufwand an Loops erfordert, wettgemacht.

Sowohl die Halbierung, als auch die Verdoppelung des Segmentvervielfachungsfaktors N_{weight} hatte Verbesserungen der Zielfunktion zur Folge. Die Rechenzeiten haben sich (gemäß der niedrigeren bzw. höheren Auswahlwahrscheinlichkeit eines Gewichtparameters) entsprechend um einen Betrag von etwa sechs Minuten verbessert bzw. verschlechtert. Letzteres war zu erwarten, wohingegen keine Verschlechterung durch die Halbierung der Gewichtparameter festgestellt wurde. Weiteführende Tests haben gezeigt, dass jenes Ergebnis offenbar eine Ausnahme darstellt und eine höhere Auswahlwahrscheinlichkeit von Gewichtparametern eben doch mit Zielfunktionsverbesserungen korelliert. Da die Verdoppelung der Auswahlwahrscheinlichkeit allerdings keinen signifikanten Vorteil gezeigt hat, wurde aufgrund der Zeitersparnis $N_{weight} = 2$ übernommen.

Bei niedrigeren Abkühlraten hat sich erwartungsgemäß eine kürzere Rechenzeit aufgrund schnellerer Konvergenz feststellen lassen. Die Zielfunktion hat sich hier in beiden Fällen verschlechtert, unterscheidet sich jeweils aber nur in unwesentlichen Nachkommastellen. Höhere Abkühlraten sind demzufolge immer zu bevorzugen. Tendenziell lässt sich desweiteren nur eine geringe Verbesserung des Ergebnisses bei weiterer Steigerung der Abkühlrate und des damit verbundenen, erhöhten Zeitaufwandes vermuten. Als Kompromiss zwischen Rechenzeit und Ergebnisqualität wurde $\alpha_{cool} = 0.95$ beibehalten.

4.1.1 Sigma-Leaf

Der Verlauf der Zielfunktionsverbesserung bei steigendem Sigma zeigt ein deutliches Konvergenzverhalten. Die Zeitwerte lassen hingegen keine besondere Systematik erkennen. Wegen dem besten Ergebnis für $\sigma_{Leaf} = 5.5$ wurde dieser Wert übernommen. Die Wahrscheinlichkeiten der betragsmäßigen Leafänderungen für dieses Sigma sind Tabelle

4.1 zu entnehmen. Es zeigt sich, dass die Wahrscheinlichkeit für $X = 0$ vergleichsweise

$\|x\|$	0	1	2	3	4	5	6	7	8	9	10	11
$P(X = \|x\|)$ [in %]	7.3	14.2	13.5	12.4	11.1	9.5	7.9	6.4	5.0	3.8	2.7	1.9

Tabelle 4.1: Wahrscheinlichkeiten für die Änderung eines Leafs um x Positionen; für $\sigma = 5.5$

gering ist, also trotz des Erwartungswertes von 0 der stetigen Gaußverteilung nicht dominant ist. Bei der diskretisierten Verteilung beträgt der Tabelle zufolge die betragsmäßig erwartete Leafänderung etwa 4 Bixel. Dieser Wert ist relativ hoch und führt zu sehr "zerzausten" Aperturen. Dadurch kommt es bei der Bestrahlung solcher Aperturen zu Streueffekten an den Leafkanten, weshalb die in der Optimierung verwendete Dosisverteilung geringfügig von der tatsächlichen abweicht. Um das exakte Ergebnis zu erhalten ist hier deshalb unbedingt eine weitere XVMC Vorwärtsrechnung der optimierten Aperturkonfiguration erforderlich. Wie groß die resultierenden Unterschiede aber ausfallen und ob diese überhaupt relevant sind muss durch separate Studien untersucht werden. Zur Vermeidung solcher ggf. problematischen Aperturen könnten in zukünftigen Weiterentwicklungen eventuell kleinere Sigma-Werte und zusätzliche MLC-Soft-Constraints, wie beispielsweise die Minimierung des Öffnungsumfangs einer Apertur, eingeführt werden.

4.1.2 Sigma-Weight

Im Balkendiagramm der Mittelwerte (Abbildung 3.2, Seite 56) zeigen die Zeitwerte ein deutliches Konvergenzverhalten, bei nahezu konstant bleibender Zielfunktionsverbesserung. Der sinkende Zeitaufwand bei steigendem Sigma lässt sich durch die Hard-Constraints für Gewichte erklären: Ein großes Sigma macht große Gewichtänderungen wahrscheinlicher, was dementsprechend auch die Überschreitung/Unterschreitung der Gewichtgrenzen wahrscheinlicher macht. Modifikationen werden bei größerem Sigma also öfter verworfen, bevor überhaupt eine Aktualisierung der Dosismatrix und ggf. eine Rückrechnung stattfindet. Der dadurch hohe Zeitgewinn führt deshalb im Gegenzug zu einer verringerten Segmentgewichtvariation, was das Finden eines Optimums erschwert.

Dennoch wurde $\sigma_{Weight} = 1.0$ gewählt, da sich für diesen Wert eine besonders hohe Stabilität der Zielfunktion ergibt. Abbildung 3.3 (Seite 56) verdeutlicht diese Eigenschaft. Hier sind alle vier Testdurchläufe und die zu den Sigmawerten gehörenden Zielfunktionsverbesserungen aufgetragen. Erstaunlicherweise unterscheiden sich die Verläufe deutlich, obwohl jeweils identische Ausgangsbedingungen vorhanden waren. Für $\sigma_{Weight} = 1.0$ war diese Variation der Ergebnisse am geringsten. Bei der Gaußverteilung mit Erwartungswert 1.0 ergibt sich allerdings das allgemeine Problem, dass eine Verkleinerung des Segmentgewichts um den Faktor $\frac{1}{x}$ $(x > 1)$ immer wahrscheinlicher ist als die Vergrößerung mit dem entsprechenden Faktor x. Umgekehrt ergibt sich jedoch wegen den entfallenden negativen Werten eine etwas größere Wahrscheinlichkeit für Vergrößerungen des Segmentgewichts. Um eine Balance für dieses Verhalten zu finden war die Bestimmung des Sigmas dieser Gaußverteilung notwendig. Zur weiteren Verbesserung der Modifikationsfaktoren müssen zusätzliche Studien durchgeführt werden.

4.2 Optimierungsergebnisse

Zur Auswertung der Optimierungsergebnisse werden im Folgenden für jeden Fall zunächst alle verwendeten Optimierungsparameter angegeben und qualitative Aussagen anhand direkter DVH-Vergleiche und der Auflistung einiger (D_V und V_D) Kennwerte getroffen. Das Primärziel einer jeden IMRT Planung ist eine möglichst hohe Homogenität der Dosisverteilung im PTV, welche durch den einheitenlosen Wert $H := \frac{D_5 - D_{95}}{D_{ref}} \cdot 100$ charakterisiert wird, der im Idealfall gleich Null ist. Die jeweiligen Diagramme, die den Verlauf der DAO Optimierung zeigen, enthalten alle wichtigen und während der Optimierung aufgezeichneten Daten. Bis auf die Temperaturkurve selbst (links oben) sind alle weitere Kurven in logarithmischen Koordinatensystemen gegenüber der Temperatur aufgetragen. Dort lassen sich, neben der spezifischen Wärme und der Akzeptanzrate von zufälligen Änderungen des Systemzustands, die Zielfunktionsanteile des PTVs und der OARs, sowie die Gesamtzielfunktion in ihrem Verlauf betrachten. In den verschiedenen Fällen lassen sich hier keine bedeutenden Unterschiede feststellen.

4.2.1 Prostatafall ohne Hüftknochen

Der Vergleich der DVH-Kurven (Abbildung 3.4) zeigt, dass trotz der im Verhältnis zu den OARs hohen Priorität des PTVs (rot) bei der IKO-Rechnung, die Homogenität des DAO Ergebnisses (Tabelle 3.6) mit 3.7 deutlich besser ist und die PTV-Constraints kaum verletzt werden. Bei den OARs lässt sich außerdem eine allgemein bessere Schonung erkennen, wobei deren Belastung im Hochdosisbereich aber größer ist als bei IKO. Die Constraints für die Blase wurden sowohl von IKO als auch von DAO erfüllt. Beide Kurven (grün) können im strahlentherapeutischen Sinne daher als identisch angesehen werden. Für das Rektum (blau) wurden in beiden Fällen alle definierten Constraints verletzt. Da ein Teil des Rektums aber im Regelfall zum Zielvolumen gezählt wird ist eine hohe Dosis dort durchaus erwünscht und nicht vermeidbar. Dennoch zeigen die Schnittbilder (Abbildung 3.6), dass DAO einen recht konstanten Randsaum höherer Dosis um das PTV erzeugt, wohingegen bei IKO ein deutlicher Einschnitt im Bereich des Rektums zu erkennen ist. Das Schnittbild aus Schicht 38 zeigt bei IKO sogar eine deutliche Aussparung innerhalb der hochdosierten Zone. DAO ist hier offenbar nicht so gut in der Lage Einbuchtungen im Randbereich des PTVs zu erzeugen, wobei jedoch zu beachten ist, dass wegen der direkten Angrenzung des Rektums daraus automatisch eine Verschlechterung des PTVs folgen würde. DV-Constraints und Penaltyfaktoren dienen daher immer nur dem Zweck einen Kompromiss zwischen hoher PTV Homogenität/Dosis und der OAR Schonung zu finden. Am DVH-Kurvenverlauf der Außenkontur ist zu erkennen, dass die mittels DAO bestimmte Dosisverteilung im gesamten modellierten Volumen eine insgesamt etwas geringere Belastung des Normalgewebes zur Folge hat. Die Schnittbilder der optimierten Dosisverteilungen in Abbildung 3.6 verdeutlich das anhand der Streuungen des Hochdosisbereichs außerhalb der PTV Kontur. Zu erwähnen ist hier, dass bei IKO die Constraints der Hilfskontur (UT-Margin, orange) im unteren Bild (Schicht 38) trotz der offenbar hohen Belastung insgesamt dennoch erfüllt wurden. Zur Vermeidung solcher Streuungen, die bei DAO nicht vorkommen, wären noch härtere Constraints notwendig. Die DAO Programmlaufzeit von knapp 6 Stunden ist zwar recht hoch, doch im Vergleich zu anderen DAO Rechnungen (siehe die folgenden zwei Ergeb-

nisse) deutlich geringer. Eine Erklärung dafür liefert die geringere Anzahl betrachteter Voxel und Entscheidungsvariablen (Parameter).

4.2.2 Prostatafall mit Hüftknochen

Hier zeigt das DVH (Abbildung 3.7) eine sehr gute Übereinstimmung beider PTV-Kurven (rot), wobei die Homogenität bei DAO etwas besser ist als bei IKO. Auch die Kurvenverläufe für Blase (grün) und Rektum (blau) sind kaum unterschiedlich. Bei DAO lässt sich für die Blase ein allgemein besseres und im Hochdosisbereich ein etwas schlechteres Ergebnis feststellen, wobei für das Rektum gerade das Gegenteil der Fall ist. Sowohl bei DAO als auch bei IKO wurden die Constraints (Tabelle 3.7) für Blase und Rektum stark verletzt. Bei den Hüftknochen (magenta, cyan) zeigen sich dagegen allgemein sehr gute Ergebnisse, die Schonung konnte hier mit DAO teilweise deutlich besser realisiert werden. Auch die Außenkontur (schwarz) zeigt bei DAO eine deutlich geringere Strahlenbelastung. Die Schnittbilder (Abbildung 3.9) zeigen das gleiche Verhalten wie auch schon beim Prostatafall ohne Hüftknochen. Die Schonung des Rektums wurde mit DAO hier jedoch etwas besser erreicht, vor allem im Hochdosisbereich. Zurückzuführen ist das auf den höheren Penaltyfaktor. Insgesamt konnte hier ein zu IKO weitgehend besseres Ergebnis erzielt werden. Die Laufzeit liegt jedoch bei etwas über 12 Stunden, was sich primär auf die hohe Anzahl der Voxel und wegen der vielen Segmente auch hohen Anzahl an Parametern zurückführen lässt.

4.2.3 Quasimodo-Phantom

Wie der DVH Vergleich (Abbildung 3.10) zeigt ist das DAO Ergebnis für das Quasimodo Phantom insgesamt erheblich schlechter als das von IKO. Im PTV konnte mit DAO jedoch eine signifikant bessere Homogenität erreicht werden. Betrachtet man dazu Abbildung 3.12 offenbaren sich hier die Schwächen des entwickelten DAO Verfahrens. Die Schonung des OARs gelingt hier nur ansatzweise. In anderen Schichten dagegen bilden sich keilförmige Aussparungen des Hochdosisbereichs, die in den unteren Ecken des PTVs zu Unterdosierungen führen. Bei stärkerer Gewichtung des OARs verstärkt

sich diese Keilform noch weiter und überträgt sich in weitere Schichten. Bei einer Absenkung der OAR Gewichtung wird dieses nahezu ignoriert und in folge dessen stark überdosiert. Die DAO Optimierung ist hier also offenbar nicht in der Lage allein durch das Zuweisen bestimmter Penaltyfaktoren eine Aperturkonfiguration zu erzeugen, die bei Berücksichtigung des OARs eine Keilbildung in der Dosisverteilung vermeidet. Im Rahmen dieser Arbeit war es nicht mehr möglich genauere Untersuchungen zur Bestimmung der Ursache für dieses Verhalten durchzuführen. Den bisherigen Ergebnissen nach zu urteilen liegt das Problem aber nicht bei den Planparametern, sondern an bestimmten Eigenschaften des implementierten Optimierungsverfahrens. Die Tatsache, dass sich der Verlauf der Optimierung (Abbildung 3.11) von denen der beiden Prostatafälle nicht erkennbar unterscheidet, unterstützt diese Vermutung. Die verwendeten und mit denen der Prostatafälle identischen Optimierungsparameter (`Std_Prostata.dao`) könnten andererseits auch Grund dieses Verhaltens sein. Es ist deshalb sinnvoll bei weiterführenden Untersuchungen zunächst für das Quasimodo spezifische Optimierungsparameter zu bestimmen. Überraschend ist zudem die Laufzeit von über 15 Stunden, obwohl im Vergleich zu dem Prostatafall mit Hüftknochen genau gleich viele Loops, weniger Voxel und auch weniger Segmente verwendet wurden. Wegen der enormen Größe des Zielvolumens des Quasimodo Phantoms sind pro Segment allerdings bis zu vier Leafpaare mehr erforderlich als bei klassischen Prostatafällen. Die Anzahl der Parameter erhöht sich also um $2 \cdot 4 \cdot 60 = 480$. Da sich die Wiederholungen der Iterationsschleife des Optimierungsalgorithmus nach der Anzahl der Parameter richtet, ist ein solches Verhalten der Laufzeit demnach sogar zu erwarten.

4.2.4 Vergleich von DAO mit DAO Vorwärtsrechnung

Um eine erste Abschätzung zu treffen wie groß sich die Streueffekte an den Leafkanten auf das direkte Ergebnis der DAO Rechnung auswirken, wurde dieses mit der XVMC Vorwärtsrechnung der VMC-Ergebnisdatei verglichen. Abbildung 3.13 zeigt beide DVHs in einem Diagramm. Visuell sind hier kaum Unterschiede zu erkennen. Bei Betrachtung der Differenzen (Abbildung 3.14) zeigt sich, dass sämtliche Unterschiede in den OARs

unterhalb von einem Prozent liegen. Außerdem zeigen die oberhalb der X-Achse liegenden Kurven, dass das tatsächliche (XVMC) Ergebnis sogar besser ist als das direkte DAO Ergebnis. Im PTV (rot) zeigen sich Differenzen von bis zu drei Prozent, die hier ebenfalls einer Verbesserung entsprechen (da hohe Dosen im PTV erwünscht sind ist das Diagramm hier umgekehrt zu interpretieren). Aus diesem Ergebnis lässt sich folgern, dass eine aus der DAO Optimierung direkt folgende Dosisverteilung als weitgehend korrekt angenommen werden kann. Unterschiede gegenüber des exakten Ergebnisses durch eine weitere XVMC Vorwärtsrechnung sind nur in geringem Maße und auf positive Weise zu erwarten.

5 Schlussfolgerungen und Ausblick

In der vorliegenden Arbeit wurde die Umsetzung eines Verfahrens zur direkten Aperturoptimierung mit einem Simulated Annealing Algorithmus unter Verwendung der IK-Doseengine beschrieben und anhand von Prostatafällen qualitativ untersucht. Im Weiteren wurde die Abhängigkeit der Optimierung von den zahlreichen Optimierungsparametern in Bezug auf das Ergebnis und die Laufzeit bei der Optimierung eines Prostataplans ermittelt und sinnvolle Werte festgelegt. Zur Ermittlung eines geeigneten Zufallsgenerators für den DAO Algorithmus wurden einige Kriterien formuliert und eine Auswahl an Zufallsgeneratoren dahingehend getestet.

Ein grundlegender und ganz erheblicher Vorteil des DAO Vefahrens gegenüber IKO ist die Unabhängigkeit von externer Segmentiersoftware und das Entfallen der arbeitsaufwändigen Segmentierungs- und Reoptimierungsmaßnahmen. Eine Adaption an andere MLC Typen ist zudem leicht, ggf. sogar nur durch die Anpassung weniger Konfigurationsparameter möglich. Aktuell wird von DAO nur der MLC des *PRIMUS* Beschleunigers (Siemens) unterstützt, jedoch ist die Erweiterung um den *Synergy* Beschleuniger (Elekta) bereits fest vorgesehen.

Die Entwicklung des DAO Algorithmus unter C++ war stets darauf gerichtet eine spätere Integration in das vorhandene IKO System möglichst einfach zu gestalten. Dennoch sollte DAO zunächst als eigenständig lauffähiges Programm umgesetzt werden, da zu Beginn der Studie noch nicht klar gewesen ist ob die Ergebnisse überhaupt Verwendung finden können. Wie die Ergebnisse an den beiden Prostatafällen aber zeigen, ist mit dem gegenwärtigen DAO Verfahren ohne Zweifel eine Planoptimierung möglich, die qualitativ mit reoptimierten IKO Plänen durchaus mithalten und diese in einigen Punkte

sogar deutlich übertreffen kann. Vor allem die PTV Vorgaben konnten sehr gut erfüllt werden. Bei den Risikoorganen ergaben sich nur wenig signifikante Verbesserungen, teilweise auch Verschlechterungen, die aber in keinem Fall relevant waren. Die DVH-Kurve der Außenkontur zeigt, dass die Strahlenbelastung des umliegenden Normalgewebes bei DAO im Allgemeinen deutlich geringer ist. Wie die Schnittbilder jedoch erkennen lassen hat die IKO Lösung gerade hier einige Mängel, die jedoch mit hoher Wahrscheinlichkeit auf zu schwache DV-Constraints für die UT-Margin Hilfskontur zurückgeführt werden können. Eine Verfeinerung der IKO Optimierung wird diese Differenz daher sicherlich verringern.

Das Quasimodo Phantom hat einen bestechenden Mangel der DAO Optimierung zutage gefördert. DAO ist offenbar nicht in der Lage ein hufeisenförmiges Zielvolumen geeignet zu berücksichtigen. Zwar hat der Algorithmus durchaus richtig funktioniert, doch aufgrund aktuell nicht nachvollziehbarer Gründe wird eine Aussparung erzeugt, die zwar die Dosis im OAR senkt, gleichzeitig aber für völlig unterdosierte Bereiche im PTV sorgt. Aufgrund der notwendigerweise geschickten Einstrahlung auf ein solches konvex geformtes Zielvolumen hat es den Anschein, als ob bestimmte Mechanismen des Simulated Annealing Algorithmus eine Änderung des Systemzustandes auf eine Weise verhindert, dass eine solche geschickte Konfiguration nicht angenommen werden kann. Genaueres zu den Ursachen dieses Verhaltens ist durch weitere Untersuchungen herauszufinden und konnte im Rahmen dieser Arbeit nicht mehr durchgeführt werden. Es dürfte sinnvoll sein zunächst spezifische Optimierungsparameter für das Quasimodo Phantom zu bestimmen, da die für Prostatafälle ermittelten Werte womöglich auch nur für solche zylinderähnlichen Volumenstrukturen geeignet sind.

Die Abhängigkeiten des Systems von den Optimierungsparametern konnte plausibel bestimmt werden und anhand der Testergebnisse wurden sinnvolle Werte ermittelt. Dennoch ist keineswegs ausgeschlossen, dass nicht noch bessere Wertkombinationen existieren, da nur eine empirische Bestimmung dieser Parameterwerte möglich ist. Auch der knappe Testumfang ist sicherlich nicht ausreichend um definitive Aussagen treffen zu können, sehr wohl aber hinreichende. Zu der für Gewichtänderungen zugrundeliegen-

den Normalverteilung ist anzumerken, dass die Wahl dieser Verteilung (wie auch bei den Leafänderungen) getroffen wurde um eine Glättung der gewürfelten Änderungen zu erreichen. Zwar wurde durch Testreihen ein möglichst optimaler Wert für jenes Sigma ermittelt, doch es ist eine Bevorzugung von Verkleinerungen gegenüber Vergrößerungen des Segmentgewichts wegen der multiplikativen Verwendung der gewürfelten Werte gegeben. Es kann daher sinnvoll sein auch andere Verteilungen zu untersuchen. Die Verwendung einer Chi-Quadrat Verteilung mit etwa vier Freiheitsgraden könnte einen Ausgleich zwischen der Wahrscheinlichkeit für Werte kleiner und Werte größer 1 liefern. Auch negative Werte wären automatisch ausgeschlossen.

Deutlich schwerwiegender sind die sehr hohen Rechenzeiten, die bei DAO Optimierungen mit Voxelauflösungen (XY-Ebene) von 128×128 selten unter 12 Stunden liegen. Grund dafür ist das verwendete Simulated Annealing Verfahren, welches eine sehr häufige Berechnung der Zielfunktion und Teilen der Dosismatrix erfordert. Die stetige Entwicklung neuer Computertechnologien minimiert dieses Problem jedoch zunehmend. Ein primärer Teil der zukünftigen Anstrengungen wird dennoch auf die Optimierung des Programmcodes, mit Absicht auf kürzere Laufzeiten, fallen.

Die Tests zu den Zufallsgeneratoren lieferten unerwartete Ergebnisse. Es hat sich gezeigt, dass zumindest in den getesteten Eigenschaften keine grundlegenden Unterschiede in der Leistungsfähigkeit der verschiedenen getesteten Generatoren festzustellen ist, abgesehen von der Laufzeit des ranlxd1 und der Spannweite des C++ rand() Generators. Da die Entwicklung und Durchführung der Tests nur sekundär erfolgte wurde bereits lange vor der endgültigen Auswertung in DAO der *GSL taus* Generator mit der Methode *uniform_pos* verwendet. Auch wenn der *GSL mt19937* Generator eine geringfügig bessere Häufigkeitsverteilung verspricht, wurde der implementierte Generator seither nicht mehr ausgetauscht. Die Ergebnisse zeigen zudem, dass der *GSL taus* Generator in jeder Hinsicht konkurrenzfähig ist.

Zusammenfassend lässt sich sagen, dass die DAO Methode durchaus vielversprechende Ergebnisse liefert und eine echte Alternative zu bisherigen Verfahren darstellt. Die Übertragbarkeit auf beliebige Beschleunigertypen macht DAO vielseitig einsetzbar und

unabhängig. In absehbarer Zeit könnte sich dieses Verfahren durchaus als qualitative Innovation im Bereich der IMRT Bestrahlungsplanung erweisen.

A Anhang

A.1 Hochrechnung zur Häufigkeit der Aufrufe des Zufallszahlengenerators in DAO

Das folgende Listing zeigt, wie auf die Anzahl der Aufrufe eines Generators *RND* (mit Periode 2^{31}) pro Optimierungsdurchlauf geschlossen werden kann, wenn die aufgeführten Anzahlen an Parametern verwendet werden. In runden Klammern sind einige Werte in Faktoren zerlegt angegeben um deren Herkunft zu verdeutlichen. Die eckigen Klammern zeigen zum Vergleich, wieviele Parameter bei Bestrahlungsplänen in der Praxis durchschnittlich tatsächlich verwendet werden:

Beams:	10 [7]
Segmente:	20 [10]
Bixel pro Segment:	529 (23x23) [64 (8x8)]
Iterationen pro Loop:	10 [4]
Parameter pro Loop:	1058000 (10x20x529x10)
Loops:	400 [250]
Aufrufe von RND pro Parameter:	4.54 $(1 + 1 + 2 \cdot 1.27)$
Periode von RND:	2147483648 (2^{31})
Aufrufe von RND insgesamt:	1921328000 (400x1058000x3)

Man erkennt, dass eine Periode von 2^{31} im Extremfall durchaus zu klein sein kann. Durch Protokollierung mehrerer realistischer Bestrahlungspläne während der Optimie-

rung wurde jedoch gezeigt, dass die Anzahl der Aufrufe des Zufallsgenerators sogar unter $3 \cdot 10^6 \approx 2^{21.5} < 2^{31}$ liegt. Im Vergleich: Laut Manual [1] besitzt der *GSL Taus*-Generator eine Periode von 2^{88}. Eine Beachtung der Periode des verwendeten Zufallsgenerators ist demnach nur bei der Optimierung extrem komplexer Pläne notwendig, wenn sich diese durch erhöhte Parameterwerte auszeichnen, die ein direkt proportionales Verhältnis zur Anzahl der Generatoraufrufe haben. Das ist in höchstem Maße bei der Gesamtzahl der verwendeten Segmente und geforderten Gesamtiterationen der Fall.[1]

A.2 Bestimmung geeigneter Zufallszahlengeneratoren

Stochastische Optimierungsverfahren, wie das in DAO integrierte Simulated Annealing, basieren auf zufälliger Variation der Ergebnisvariablen. Bei der programmatischen Umsetzung muss daher auf Zufallsgeneratoren zurückgegriffen werden. Die Anforderungen an solche Generatoren sind dabei vergleichsweise hoch, da die Variationen keiner erkennbaren Systematik unterliegen dürfen. Welche Anforderungen DAO hier im Detail stellt und welche Kriterien dazu erfüllt werden müssen, wird in den folgenden Abschnitten erläutert.

A.2.1 Zufallszahlengeneratoren

Es gibt zwei Arten von Zufallszahlengeneratoren: Deterministische und nicht-deterministische. Nicht-deterministische Generatoren zeichnen sich dadurch aus, dass sie *echte* Zufallszahlen liefern können, was sich z.B. durch radioaktive Zerfallsprozesse realisieren lässt. Da Computer aber deterministisch arbeiten, können als Softwarelösung auch nur deterministische Zufallsgeneratoren verwendet werden (Ausnahmen bilden höchstens *Hybrid-*

[1] Der Wert 4.54 für die Anzahl der RND Aufrufe ergibt sich wie folgt: Die Auswahl des Parameters erfolgt zufällig (1 RND Aufruf) und es wurde die unrealistische Annahme getroffen, dass jede Parameteränderung zu einer Verschlechterung des Systemzustands führt, die Akzeptanz des neuen Zustandes also gewürfelt werden muss (1 RND Aufruf). Der Erwartungswert für die Anzahl der Versuche eine gültige normalverteilte Zufallszahl mit der Polar-Methode zu erzeugen beträgt 1.27 (dieses Ereignis ist trivialerweise NB$(1, \frac{\pi}{4})$-verteilt) und erfordert pro Versuch 2 RND Aufrufe.

Generatoren, die intern aber ebenfalls deterministisch arbeiten). Weil die von deterministischen Generatoren erzeugten Zufallszahlen nicht wirklich dem Zufall unterliegen spricht man hierbei auch von *Pseudozufallszahlen*. Im Folgenden werden jedoch ausschließlich deterministische Generatoren behandelt, weshalb nicht mehr explizit darunter unterschieden wird.

A.2.1.1 Anforderungen an den Zufallsgenerator

Die Anforderungen an einen Zufallsgenerator sind je nach Anwendungsgebiet sehr unterschiedlich. Im Falle von DAO müssen vier verschiedene Kriterien möglichst gut erfüllt werden. Diese sind im Einzelnen:

- Hohe Periodizität
- Gleichverteilung
- Hohe Rechengeschwindigkeit
- Homogenität

Im Folgenden wird darauf eingegangen unter welchen Aspekten diese Anforderungen für das DAO-Verfahren wichtig sind und wie ein Zufallszahlengenerator als dafür geeignet bestimmt werden kann.

A.2.1.2 Signifikanz der Periodizität

Die hohe Periodizität ist ein extrem wichtiges Merkmal eines jeden deterministischen Zufallsgenerators, weil sich die generierten Zahlenwerte bei bekanntem Startwert (engl. *seed*) vorhersagen lassen und sich irgendwann wiederholen werden. Bei sehr niedrigen Periodizitäten würden die erzeugten Zufallszahlen daher einer deutlichen Systematik unterliegen, also eine kurze und statische Zahlenfolge liefern, was bei einem Zufallsgenerator aber gerade nicht erwünscht ist. Aus diesem Grund besitzen moderne Zufallsgeneratoren ohnehin recht hohe Periodizitäten von etwa 2^{31}. Für manche Anwendungen kann es aber wichtig sein eine besonders hohe Periodizität zu garantieren. Im Falle des in dieser Arbeit beschriebenen Simulated Annealing (SA) Algorithmus sind aber auch die Perioden

gängiger Generatoren völlig ausreichend, da das Optimierungsverfahren mit realistischen Parametern immer vor dem Erreichen des Periodenendes abbricht. Das lässt sich anhand einer Hochrechnung und durchgeführten Tests zeigen (siehe Anhang A.1, Seite 83). Sollte die Periode dennoch nicht ausreichen, so kann durch ein Neusetzen des Generator-Seeds eine neue Sequenz mit einer (im Allg.) gleich langen Periode erzeugt werden.

A.2.1.3 Signifikanz der Gleichverteilung

Der DAO Algorithmus enthält insgesamt an drei Stellen stochastische Methoden:

- bei der Wahl des zu modifizierenden Parameters
- bei der Wahl der Wertänderung des gewählten Parameters
- bei der Nichtakzeptanz eines neuen Zielfunktionswertes

Hierbei werden überall gleichverteilte Zufallsgeneratoren benötigt. Dabei ist der Anspruch an die Gleichverteilung bei der Wahl des zu modifizierenden Parameters besonders hoch. Denn falls der Generator einen oder mehrere Parameter insgesamt deutlich öfter auswählt als alle anderen, so wird der Freiheitsgrad des SA Algorithmus unter Umständen so stark beschränkt, dass keine gute Lösung mehr gefunden werden kann. Bei der Akzeptanz eines neuen Zielfunktionswertes dagegen spielt eine exakte Gleichverteilung eher eine untergeordnete Rolle, da es in diesem Fall nur darum geht irgendeine beliebige Zahl zwischen 0 und 1 zu würfeln.

A.2.1.4 Signifikanz der Rechengeschwindigkeit

Die Zeit die ein Zufallszahlengenerator für das Erzeugen seiner Zufallswerte benötigt spielt nur dann eine Rolle, wenn sehr oft Zufallszahlen generiert werden müssen. Unter Verwendung der Hochrechnung aus dem Anhang (A.1, Seite 83) lässt sich abschätzen, dass in extremen Fällen insgesamt bis zu 10^{10} Zufallszahlen generiert werden müssen. Unter der Annahme, dass ein Generator für die Erzeugung von 10^7 Zufallszahlen etwa eine Sekunde benötigt, ergibt sich pro Optimierung ein Rechenaufwand von etwa 1000 Sekunden bzw. 17 Minuten. Bei sehr komplexen Bestrahlungsplänen nimmt ein

Zufallszahlengenerator also durchaus einen markanten Anteil der Programmlaufzeit in Anspruch. In herkömmlichen Fällen liegt der gesamte Anteil der Rechenzeit für den Zufallsgenerator (unter vorig genannter Annahme) sogar unter einer Sekunde (siehe ebenfalls A.1). Die Rechengeschwindigkeit des Generators ist daher zwar kein entscheidendes Kriterium, jedoch sollten Generatoren mit besonders hohen Rechenzeiten möglichst vermieden werden, um bei Bedarf auch komplexere Fälle oder erweiterte Methoden effizient verwenden zu können.

A.2.1.5 Signifikanz der Homogenität

Die Homogenität eines Zufallsgenerators wird hier als Maß der Gleichverteilung von Punkten (also Paaren von Zufallszahlen) in der XY-Ebene verstanden, diesem erzeugt wird. Schlechte Homogenität bedeutet demnach, dass die von einem Generator gelieferten Zahlenpaare aufgrund von Abhängigkeiten zur Bildung von Punktanhäufungen (Clustern) neigen. Dadurch würde sich die Qualität der simulierten normalverteilten Zufallszahlen, wegen der Verwendung der Polarmethode (Listing A.1), verschlechtern, da diese dann ebenfalls zur Clusterbildung neigen. Homogenität spielt daher immer dann eine Rolle, wenn Paare oder n-Tupel von unabhängigen, gleichverteilten Zufallszahlen generiert werden sollen. In DAO ist das nur bei der Simulation normalverteilter Zufallszahlen der Fall, weshalb speziell dafür ein Generator mit Eigenschaften gesucht wurde, der die Anforderungen der Homogenität besonders gut erfüllt.

```
double u1=0., u2=0.;
double s=0.,z;

while( !s || s > 1.) {
  u1 = 2*Rand(0)-1;
  u2 = 2*Rand(0)-1;
  s = u1*u1 + u2*u2;
}

//http://en.wikipedia.org/wiki/Box_muller#Polar_form
z = mu + sigma * u1 * sqrt( (-2.*log(s)) / s );
```

Listing A.1: Simulation normalverteilter Zufallszahlen mit der Polarmethode (C++ Code)

A.2.2 Gütekriterien für gleichverteilte Zufallsgeneratoren

Um einen gegebenen Zufallsgenerator entsprechend des vorherigen Abschnitts darauf zu prüfen, ob und wie gut er die dort aufgeführten Anforderungen im Einzelnen erfüllt, müssen zunächst allgemeine Kriterien für die Güte eines Zufallsgenerators gefunden werden. Im Folgenden werden Methoden der Statistik herangezogen um solche Kriterien zu finden und zu formulieren.

A.2.2.1 Definition der Zufallsvariablen

Ein gleichverteilter Pseudozufallszahlengenerator liefert in der Regel Werte aus dem Intervall $(0, 1)$. Da für die Qualifizierung der Güte und auch bei vielen praktischen Anwendungen der Ereignisraum in Bins unterteilt werden muss, spielen auch abweichende Intervalle keine Rolle. Denn eine Unterteilung in Bins bedeutet nichts anderes wie das Zerlegen eines Intervalls in Teilintervalle gleicher Länge. Diese Unterteilung lässt sich auch als Abbildung auf ein diskretes Intervall realisieren.

Der Ereignisraum eines jeden Generators wird daher auf ein genügend großes, natürliches Intervall $D := [1, M]$ abgebildet. Sei nun X_i die Zufallsvariable, die den Generator mit der Kennziffer i und der Abbildung des Ereignisraums auf D repräsentiert. Dann unterliegt X_i einer unbekannten Verteilung, für die aber das Verhalten einer Gleichverteilung angenommen werden darf. Es gilt formal für jede Realisierung k: $x_{i_k} \in D$. Zu erwarten wäre bei einer Gleichverteilung für jeden *Merkmalwert* $x_k \in D$ die Eigenschaft:

$$P(X_i = x_k) = \frac{1}{M} = konstant \tag{A.1}$$

Nach insgesamt n mal Würfeln ergäbe sich bei exakter Gleichverteilung für jedes Ereignis k aus D die absolute Häufigkeit:

$$h_k = h := \frac{n}{M} = konstant \tag{A.2}$$

Tatsächlich ist es aber so, dass heutige Zufallsgeneratoren nicht in der Lage sind eine Gleichverteilung exakt zu simulieren. Es treten in der Praxis deshalb immer Abweichungen zur Gleichverteilung auf, die eben gerade für die Güte eines Generators ausschlaggebend sind. Um nun die Güte eines gleichverteilten Zufallsgenerators beurteilen zu können

werden zunächst statistische Tests durchgeführt. Dabei handelt es sich um einfache Zufallsstichproben vom Umfang n mit den Realisierungen $\{x_{i_1}, x_{i_2}, ..., x_{i_n}\}$ und jeweils N Versuchsdurchführungen.

A.2.2.2 Kenngrößen

Anhand dieser Stichproben lassen sich nun einige statistische Merkmale untersuchen, die anschließend eine Aussage über die Güte bzw. Tauglichkeit des getesteten Generators machen können, indem geeignete Gütekriterien abgeleitet werden. Diese Gütekriterien lassen sich dann unter Vorgabe von Sollwerten als normierte Kenngrößen darstellen. Die Normierung erfolgt dabei so, dass der Kennwert bei Erfüllung der geforderten Eigenschaften einen Wert ≤ 1 liefert, andernfalls > 1. Durch die Bestimmung dieser Kennwerte für Stichproben eines Generators lässt sich so sofort feststellen, ob dieser die Kriterien erfüllt und wie gut er das im Vergleich zu anderen getesteten Generatoren schafft. Dadurch wird ein aussagekräftiger Vergleich von Zufallsgeneratoren möglich.

A.2.2.3 Stichprobenumfang

Aus $x_{i_k} \in D$ folgt unmittelbar, dass der Umfang n der Stichprobe mindestens gleich M sein muss um alle möglichen Werte x_k zu erhalten, was bei exakter Gleichverteilung der Fall wäre. Da eine exakte Gleichverteilung aber (wie zuvor geschildert) nicht vorausgesetzt werden kann, ist es bei Stichproben angebracht, einen möglichst großen Stichprobenumfang zu wählen. Vor allem dadurch lässt sich gemäß dem Gesetz der großen Zahlen die Genauigkeit des Ergebnisses deutlich verbessern. Sollte ein erster Test eines Generators i mit großem Stichprobenumfang ($n \gg M$) bereits zeigen, dass mindestens ein Element aus D nicht als Realisierung vorkommt und ein Fehler bei der Abbildung des Ereignisraums auf D ausgeschlossen werden kann, so ist dieser Generator bereits an dieser Stelle als untauglich zu klassifizieren. Es macht deshalb Sinn, diesen Umstand zu überprüfen bevor überhaupt an tiefergehende Tests gedacht wird.

A.2.2.4 Einzelwahrscheinlichkeiten und Mittelwerte

Wie zu Beginn dieses Abschnitts (Gleichung A.1) bereits dargelegt wurde, erwartet man bei einer exakten Gleichverteilung für jedes mögliche Ereignis die gleiche, konstante Einzelwahrscheinlichkeit von $p = \frac{1}{M}$. Die Einzelwahrscheinlichkeiten für das Annehmen der Werte x_k sind demnach ein entscheidendes Maß für die Gleichverteilung des untersuchten Generators und lassen sich durch die relativen Häufigkeiten r_k (wie in A.3 definiert) abschätzen.

$$P(X_i = x_k) \approx r_k := \frac{\text{Absolute Häufigkeit } h_k}{\text{Umfang n der Stichprobe}} \tag{A.3}$$

Wegen $n = konstant$ genügt es, nur die absolute Häufigkeit h_k zu betrachten um eine Aussage über die Einzelwahrscheinlichkeiten treffen zu können. Eine exakte Gleichverteilung liegt demnach genau dann vor, wenn jeder Stichprobenwert die absolute Häufigkeit aus (A.2) besitzt. Weil die Summe aller absoluten Häufigkeiten einer Stichprobe immer gleich dem Stichprobenumfang ist,

$$\sum_{k=1}^{M} h_k = n \tag{A.4}$$

sind der (empirische) Mittelwert der relativen (und damit auch der absoluten) Häufigkeiten konstant

$$\bar{r}_k = \frac{1}{M} \sum_{k=1}^{M} r_k = \frac{1}{n \cdot M} \sum_{k=1}^{M} h_k \overset{(A.4)}{=} \frac{1}{M} = konstant \tag{A.5}$$

$$\bar{h}_k = n \cdot \bar{r}_k = \frac{n}{M} = konstant \tag{A.6}$$

und deshalb grundsätzlich nicht aussagekräftig. Der (empirische) Mittelwert der Stichprobe i hingegen variiert und liegt im Idealfall ($h_k = h$, für alle k) bei $\frac{M+1}{2}$ (Gleichung (A.7)).

$$\bar{x}_i = \frac{1}{n} \sum_{k=1}^{n} x_{i_k} = \frac{1}{n} \sum_{k=1}^{M} h \cdot x_k \overset{(A.2)}{=} \frac{1}{M} \sum_{k=1}^{M} x_k = \frac{M+1}{2} \tag{A.7}$$

Als Gütemaß ist der Stichprobenmittelwert jedoch ungeeignet, da er keinerlei Aussagen über den Verlauf der Dichte von X_i machen kann, was aber gerade für die Qualifizierung der Gleichverteilung notwendig ist.

A.2.2.5 Spannweite

Betrachtet man den Vektor $\vec{h}^* = (h_1^*, h_2^*, \ldots, h_M^*)$ der *aufsteigend sortierten* absoluten Häufigkeiten der Merkmalwerte x_k, so erhält man eine monoton wachsende Zahlenfolge, die sich als diskrete Funktion zeichnen lässt. In Abbildung A.1 ist diese Funktion für den *rand()* Zufallsgenerator (g++-2.95) mit $M = 1000$, $n = 10^7$ und einem Seed von 0 zu sehen. Man erkennt sehr schön, dass sich die absoluten Häufigkeiten um ihren (empirischen) Mittelwert (schwarze Linie) verteilen, der erwartungsgemäß konstant bei 10^4 liegt. Darüber hinaus ist der Wert 10^4 gerade die absolute Häufigkeit, die jeder einzelne Merkmalwert haben müsste, wenn eine exakte Gleichverteilung vorliegt (weil der Mittelwert $\frac{n}{M}$ der absoluten Häufigkeiten gemäß (A.6) immer gleich der Sollhäufigkeit h (A.2) einer exakt gleichverteilten Zufallsvariable ist).

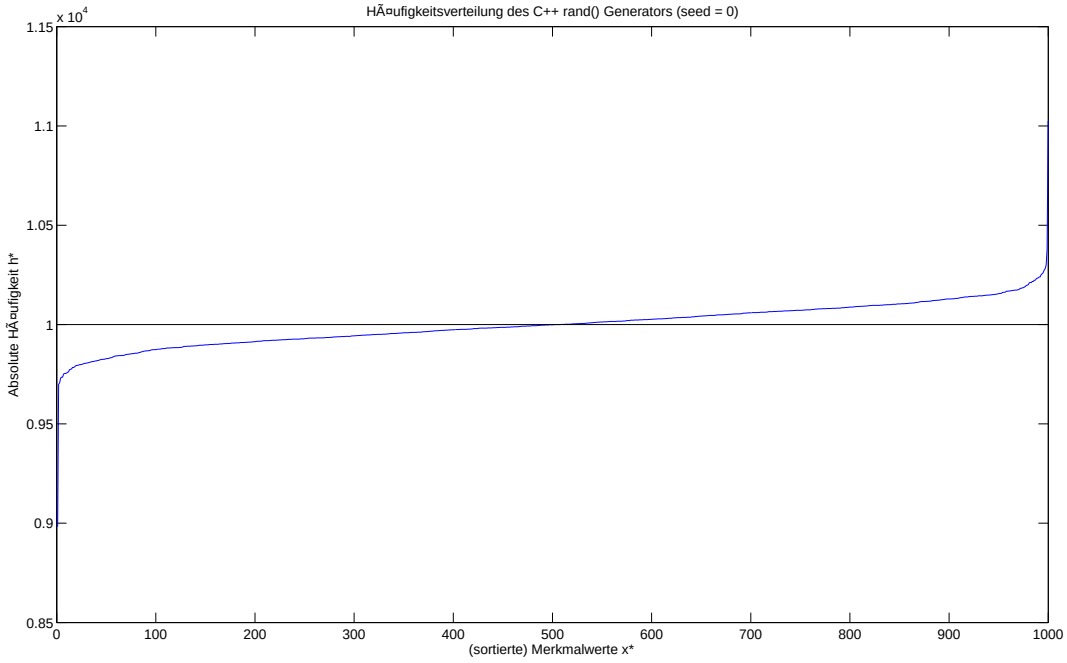

Abbildung A.1: Sortierte absolute Häufigkeiten von *rand(0)* (g++-2.95)

Man erkennt weiter, dass dieser getestete Zufallsgenerator einer Gleichverteilung schon recht nahe kommt. Allerdings zeigen sich am linken und rechten Rand starke Ausreißer. Das bedeutet, dass es vereinzelte Merkmalwerte gibt, die teilweise sehr stark benachtei-

ligt bzw. bevorzugt werden. Da sich bei jedem getesteten Zufallsgenerator das gleiche Verhalten gezeigt hat, ist die Spannweite (A.8) der Stichprobe ein geeignetes Maß um einzelne Generatoren anhand der Stärke der maximalen Ausreißer zu vergleichen. Denn je kleiner die Spannweite, desto geringer ist dementsprechend die maximale Benachteiligung bzw. Bevorzugung von Merkmalwerten eines Generators.

$$\Delta_h := h_M^* - h_1^* \tag{A.8}$$

Unter der Vorgabe einer geforderten Maximalspannweite $N_{\Delta_{max}}$, wird die normierte Kenngröße K_Δ definiert als

$$K_\Delta := \frac{\Delta_h}{N_{\Delta_{max}}}$$

A.2.2.6 Standardabweichung

Mit der Spannweite hat man zwar ein Maß für die maximale Abweichung der einzelnen absoluten Häufigkeiten vom Mittelwert, jedoch erhält man keine Aussage darüber, wieviele Werte wie stark abweichen. Das ist jedoch eine sehr wichtige Information, da zur gleichen Spannweite qualitativ völlig unterschiedliche Kurvenverläufe vorliegen können, wie Abbildung A.2 veranschaulicht. Es ist deshalb notwendig den gesamten Kurvenver-

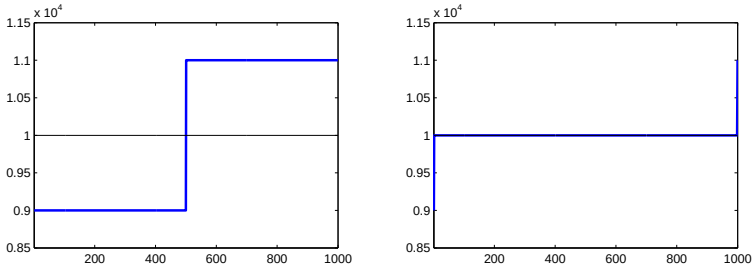

Abbildung A.2: Links: Insgesamt starke Abweichungen. Rechts: Insgesamt geringe Abweichungen

lauf von $\vec{h}^*$ zu betrachten und ein Maß für die Abweichung vom Sollverlauf (konstante

Funktion mit dem Wert $\bar{x} = \frac{n}{M}$) zu bestimmen. Ein solches Maß liefert gerade die (empirische) Varianz (A.9).

$$s^2 = \frac{1}{n-1} \sum_{k=1}^{M} h_k (x_k - \bar{x})^2 \tag{A.9}$$

Im Idealfall der diskreten Gleichverteilung hat die zugehörige Standardabweichung s den Wert $s_g := \sqrt{\frac{M^2-1}{12}}$. Bei abweichenden Verteilungen ist diese bei einer Zufallsstichprobe umso höher, je häufiger einzelne Abweichungen auftreten und je ausgeprägter diese sind. Es ist deshalb sinnvoll die (empirische) Standardabweichung der Stichprobe als Maß für die Abweichung von der Gleichverteilung zu verwenden. Für die Definition einer Kennzahl, mit der verschiedene empirische Standardabweichungen einfach untereinander verglichen werden können, ist es sinnvoll den relativen Fehler (A.10) von s bezüglich s_g zu bestimmen. Anschließend kann Anhand der Umrechnungsformel (A.11) die Anzahl der Dezimalstellen bestimmt werden, mit denen s und s_g übereinstimmen.

$$\text{relativer Fehler } = \xi := \frac{\tilde{x} - x}{x} = \frac{\text{Istwert - Sollwert}}{\text{Sollwert}} \tag{A.10}$$

$$\text{Dezimalstellen } \approx N_\xi := -log_{10}\left(|\xi|\right) \tag{A.11}$$

Am Rechner werden all diese Berechnungen mit Variablen vom Typ *double* vorgenommen, die eine Genauigkeit von etwa 15.65 Dezimalstellen aufweisen. Da nicht ohne Weiteres ausgeschlossen werden kann, dass ein Zufallsgenerator tatsächlich eine exakte Gleichverteilung liefert, ist es zweckmäßig eine obere Grenze von 16 Dezimalstellen festzulegen. Zusammen mit einer geforderten Mindestanzahl übereinstimmender Dezimalstellen $N_{S_{min}}$ und dem aus einer Stichprobe ermittelten relativen Fehler ξ_s, wird die normierte Kenngröße K_{Std} der Standardabweichung wie folgt definiert

$$K_{Std} := \frac{16 + log_{10}(|\xi_s|)}{16 - N_{S_{min}}}$$

A.2.2.7 Homogenität (PI-Test)

Eine wichtige Eigenschaft von gleichverteilten Zufallsgeneratoren ist deren Homogenität. Homogenität heißt in diesem Fall, dass gewürfelte Zahlenpaare, die sich als Punkte in der XY-Ebene interpretieren lassen, möglichst gleichmäßig verteilt sein sollten, also keine *Cluster* bilden dürfen. Das ist wichtig, weil das Erzeugen zufälliger 2D Vektoren häufig Anwendung findet. Im Fall von DAO werden solche Vektoren zur Erzeugung von normalverteilten Zufallszahlen benötigt, weshalb die Homogenität des dafür zuständigen Zufallsgenerators möglichst gut sein sollte. Ein ganz typischer Homogenitätstest von gleichverteilten Zufallsgeneratoren ist der sogenannte PI-Test. Es handelt sich dabei um einen Test der Kategorie *Kontrollaufgaben* [21] und besteht aus einem direkten Monte-Carlo-Verfahren, bei dem Zahlenpaare aus $[0,1] \times [0,1]$ gewürfelt und als Punkte im Einheitsquadrat interpretiert werden. Betrachtet wird dann die Schnittfläche mit dem Einheitskreis (siehe Abbildung A.3), indem die dorthin gefallenen Punkte gezählt werden. Anhand der Anzahl N_K der Punkte die nach N_{Ges} Zahlenpaarwürfen nun in diesem Viertelkreis liegen, lässt sich π durch Gleichung (A.12) annähern, da sich das Verhältnis der Punkteanzahl im Viertelkreis zur Gesamtanzahl auch als Flächenverhältnis des Viertelkreises zum Einheitsquadraht interpretieren lässt.

$$\pi \approx \tilde{\pi} := 4 \cdot \frac{N_K}{N_{Ges}} \tag{A.12}$$

Unter der Voraussetzung eines großen Stichprobeumfangs erhält man bei guter Homo-

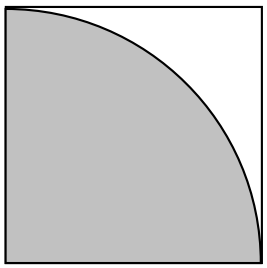

Abbildung A.3: Gewürfelt werden Punkte im Einheitsquadraht, wodurch sich der Wert von π annähern lässt

genität auch eine gute Näherung für π. An dieser Stelle lässt sich bestimmen wie genau,

also wieviele Übereinstimmungen in den Dezimalstellen von π bei einer gegebenen Anzahl N_{Ges} an Punkt-Testwürfen höchstens erreicht werden können und wieviele Punkte dazu im Viertelkreis liegen müssen. Zunächst erhält man durch Umformung von (A.12) das Verhältnis der Flächen im exakten Fall (A.13).

$$\frac{N_K}{N_{Ges}} = \frac{\pi}{4} = 0.785398... \tag{A.13}$$

Sollen etwa $\hat{N}_{Ges}$ Zahlenpaare gewürfelt werden, so ergibt sich nach (A.14) die Anzahl $\hat{N}_K$ der Punkte die im Viertelkreis liegen müssen um die bestmögliche Näherung für π zu erreichen.

$$\tilde{N}_K := 0.785398... \cdot \hat{N}_{Ges} \Longrightarrow \hat{N}_K = \lfloor \tilde{N}_K + 0.5 \rfloor \tag{A.14}$$

Dazu ist es notwendig den exakten Wert $\tilde{N}_K$ auf die am nächsten liegende ganze Zahl $\hat{N}_K$ zu runden, was durch Addition von 0.5 und Gaußklammern erreicht wird. Man erhält so nach (A.10) den durch $\hat{N}_K$ für $\hat{\pi} = 4 \cdot \frac{\hat{N}_K}{\hat{N}_{Ges}}$ resultierenden, minimalen relativen Fehler

$$\hat{\xi} = \frac{\hat{\pi} - \pi}{\pi} \tag{A.15}$$

Gleichung (A.16) liefert gemäß (A.11) für diesen relativen Fehler die Anzahl der Dezimalstellen, in denen die Näherung $\hat{\pi}$ mit dem tatsächlichen π übereinstimmt.

$$N_{\hat{\xi}} = -log_{10}(|\hat{\xi}|) \tag{A.16}$$

Bei sehr guter Homogenität eines Zufallsgenerators sind für $\tilde{\pi}$ also höchstens $N_{\hat{\xi}}$ richtige Dezimalstellen nach insgesamt $\hat{N}_{Ges}$ Punkt-Würfen ($= 2 \cdot \hat{N}_{Ges}$ einzelne Zahlen-Würfe) zu erwarten. Aus programmatischer Sicht ist es daher ausreichend den relativen Fehler nur gegenüber $\pi_{N_{\xi_{min}}}$ (= die ersten $N_{\xi_{min}}$ Dezimalstellen von π) zu bestimmen. Die dazu notwendigen Berechnungen sind in (A.17), (A.18) und (A.19) definiert.

$$N_{\xi_{min}} := \lceil N_{\hat{\xi}} \rceil \tag{A.17}$$

$$\pi_{N_{\xi_{min}}} := \lfloor \frac{\pi}{10^{N_{\xi_{min}}}} \rfloor \cdot 10^{N_{\xi_{min}}} \tag{A.18}$$

$$\tilde{\xi} := \frac{\tilde{\pi} - \pi_{N_{\xi_{min}}}}{\pi_{N_{\xi_{min}}}} = \frac{\frac{N_K}{N_{Ges}} - \pi_{N_{\xi_{min}}}}{\pi_{N_{\xi_{min}}}} \tag{A.19}$$

Zur Auswertung des relativen Fehlers ist es hier (wie auch bei der Standardabweichung) sinnvoll, anhand der Umrechnungsformel (A.11) die Anzahl übereinstimmender Dezimalstellen zu ermitteln. Diese Anzahl kann schließlich als Maß für die Homogenität eines getesteten Zufallsgenerators dienen. Die normierte Kenngröße K_π für den PI-Test wird somit durch folgende Gleichung definiert, wobei $N_{K_{min}} \in [1, \hat{N}_K)$ die minimal geforderte Anzahl an Übereinstimmungen in den Dezimalstellen von $\tilde{\pi}$ und $\pi_{N_{\xi_{min}}}$ angibt.

$$K_\pi := \frac{N_{\xi_{min}} + log_{10}(|\tilde{\xi}|)}{N_{\xi_{min}} - N_{K_{min}}}$$

A.2.2.8 Visuelle Homogenitätsprüfung

Der PI-Test des vorherigen Abschnitts liefert zwar ein Maß der Homogenität eines Zufallsgenerators für zweidimensional verteilte Punkte, doch anhand der Schätzung für π kann noch nicht auf Homogenität geschlossen werden. Denn da lediglich gezählt wird welcher Anteil der Punkte im Einheitsviertelkreis liegt, ist ein systematisches Verhalten zunächst nicht auszuschließen. Wenn keine gesicherten Informationen über einen Zufallsgenerator vorliegen ist es also notwendig und hinreichend eine grafische Darstellung der 2D-Verteilung augenscheinlich zu überprüfen um Systematiken auszuschließen. Dazu kann es sinnvoll sein, das Einheitsquadrat in Bins zu unterteilen. Man erhält so ein Rauschbild welches eine möglichst gleichmäßige und nicht systematische Struktur aufweisen sollte.

A.2.2.9 Laufzeit

Ein Zufallsgenerator benötigt zur Erzeugung einer Stichprobe vom Umfang n, also zum Generieren von n Zufallszahlen, eine gewisse Zeit t, die vollständig von der Hard- und Software des verwendeten Computersystems abhängt. Es ist deshalb sinnvoll für die Laufzeit eines Generators eine weitere Kenngröße K_t einzuführen. Für deren Normierung ist es sehr wichtig eine geeignete, rechnerspezifische Sollzeit $t_{max} > 0$ vorzugeben, die

z.B. anhand von Testläufen bestimmt werden kann. Als Kenngröße der Laufzeit wird dann definiert

$$K_t := \frac{t}{t_{max}}$$

A.2.2.10 Weitere Tests

Die Güteeigenschaften eines Zufallsgenerators werden noch über vielerlei weitere Tests geprüft. Da wären der klassische χ^2 Anpassungstest zur Überprüfung der Verteilung, diverse kombinatorische und nicht parametrische Tests [17]. Bei kommerziellen und etablierten Zufallszahlengeneratoren kann die Erfüllung der Anforderungen solcher Tests jedoch vorausgesetzt werden und bedarf keiner expliziten Überprüfung. Allgemein sind weitergehende Tests, wie die in den vorherigen Abschnitten erläuterten, nur dann notwendig, wenn Generatoren untereinander verglichen werden oder die Qualität spezieller Eigenschaften untersucht werden sollen.

A.2.3 Ergebnisse der Gütetests

Für die Realisierung eines gleichverteilten Zufallsgenerators stehen verschiedene Möglichkeiten zur Verfügung. In der verwendeten Programmiersprache sind solche Generatoren meist standardmäßig integriert (wie beispielsweise *rand()* in C++) oder können direkt als fertige C++ Quellcodes erworben und ins eigene Programm eingebunden werden. Darüber hinaus gibt es auch ganze Sammlungen (Bibliotheken) an diversen Zufallsgeneratoren, die teils auch als Freeware angeboten und verwendet werden können (hier bietet sich die kostenlose GNU Scientific Library (*GSL*) (siehe [10]) an).

A.2.3.1 Testumfang

Während der Entwicklung von DAO wurden Generatoren aus allen drei Bereichen ausgewählt und auf ihre Tauglichkeit hin untersucht. Das waren im Einzelnen:

- *rand()* (g++-2.95)
- Diverse Generatoren aus der GSL Lib
- C++ Quellcode eines RANMAR Zufallsgenerators (CERNLIB, adaptiert)
- C++ Quellcode für einen Standard Kongruenz-Generator

Die Rückgabewerte der GSL Zufallsgeneratoren können mittels spezieller Methoden als *unsigned long* oder *double* zurückgegeben werden. Dementsprechend unterscheiden sich auch die Intervalle in denen die Rückgabewerte liegen. Mit der Standardmethode *uniform_pos* erhält man double Werte im Intervall $(0, 1)$. Ganzzahlige Werte ([INT]) können mit *uniform_int* auf das Intervall $[0, n-1]$ abgebildet werden, wobei n eine beliebige ganze Zahl > 1 sein kann. Mit *get* erhält man dagegen einen algorithmusabhängigen (ganzzahligen) Wertebereich, der für den taus-Generator $[0, 4294967295]$ beträgt. Für die Generatortests wurden aus der GSL Lib die in der folgenden Liste aufgeführten Generatoren verwendet. In Klammern ist die jeweilige Rückgabemethode angegeben. Sämtliche Informationen bezüglich der Generatoren der GSL Lib entstammen dem zugehörigen Reference Manual [1].

- taus (uniform_pos)
- ranmar (uniform_pos)
- ranlxd1 (uniform_pos)
- mt19937 (uniform_pos)
- taus[INT] (get)
- taus[INT] (uniform_int)

Der Programmcode (Listing A.2) für den Standard Kongruenz-Generator entstammt einer unbekannten Quelle und wurde einem C-Programm entnommen.

```
unsigned long random (unsigned long init) {
  static unsigned long n;
  if (init>0) n = init;
  return n = 1664525 * n + 1013904223;
}
```

Listing A.2: Standard Kongruenz Generator (C++)

Insgesamt wurden also sieben verschiedene Generatoren und drei Varianten des GSL taus Generators getestet. Die Testdurchführungen erfolgten gemäß der folgenden Parameter:

- Anzahl der Tests N pro Generator = 50 (direkt aufeinanderfolgend)
- Anzahl der Testwürfe $n = 10^7$
- Anzahl der Ereignisse $M = 10^3$ (Wertebereich: $x \in [0, 999]$)
- Von der Systemzeit abhängiger Zufalls-Seed pro Testlauf eines Generators

Problematisch ist hier der Sollwertebereich, der von denen der Zufallsgeneratoren abweicht. Deshalb mussten diese jeweils auf das ganzzahlige Intevall $[0, 999]$ abgebildet werden. Die mit uniform_pos erhaltenen Werten lassen sich dabei einfach durch Diskretisierung abbilden: $(0, 1) \rightarrow [0, 999]$. Bei ganzzahligen Methoden wie get und uniform_int ist noch eine Umwandlung in Gleitpunktzahlen ($x \rightarrow y^*$) als Zwischenschritt notwendig: $[a, b] \rightarrow [c^*, d^*] \rightarrow [0, 999]$. Falls $a = 0$ und $b \geq 999$ lässt sich dieser Zwischenschritt durch Modulorechnung $x \in [0, b] \rightarrow \hat{x} = x \bmod 1000 \rightarrow \hat{x} \in [0, 999]$ umgehen. Tests haben hierbei ergeben, dass letztere Methode fast um den Faktor 4 kleinere Spannweiten der Häufigkeitsverteilung erzeugt und deshalb zunächst eher in Frage kommt. Modulorechnung hat allerdings den schwerwiegenden Nachteil, dass nur die letzten Bitstellen (low-order Bits) der generierten Zufallszahlen verwendet werden. Dieser Umstand führt dazu, dass die volle Qualität des Zufallsgenerators nicht ausgenutzt werden kann. Die Folge können systematisches Verhalten und der Verlust der Gleichverteilung sein, vor allem bei der Abbildung auf sehr kleine Intervalle. Am Beispiel des Intervalls $[0, 999]$ ergibt sich, das von den vollen 32 Bit der durch die ganzzahligen Methoden gelieferten Zufallszahlen nur die ersten 10 Bits verwendet werden ($\log_2 1000 = 9.9658$). Da bei realen Optimierungen selten mehr als 2000 zu bestimmende Parameter auftreten, kann grundsätzlich mit einem Verlust von etwa 20 Bits gerechnet werden. Programmatisch darf die Abbildung daher nur per Divisionsoperator erfolgen, wodurch immer die volle Anzahl an Bits genutzt wird.

A.2.3.2 Anforderungen

Die in Abschnitt A.2.2 definierten Kenngrößen hängen jeweils von einem Parameter ab, der den gewünschten Sollwert der jeweiligen Testgröße enthält. Diese wurden wie in Tabelle A.1 aufgeführt festgelegt. Für den PI-Test ergab sich dabei mit $\hat{N}_{Ges} = 5 \cdot 10^6$

Kenngröße	**Testgröße**	**Typ**	**Sollwert**
K_Δ	$N_{\Delta_{max}}$	Differenz	650
K_{Std}	$-log_{10}(\lvert\xi_s\rvert)$	Dezimalstellen	4
K_π	$-log_{10}\left(\left\lvert\tilde{\xi}\right\rvert\right)$	Dezimalstellen	3.5
K_t	t_{max}	Zeit (in s)	0.5

Tabelle A.1: Sollwerte der Testgrößen

für $N_{\xi_{min}}$ gemäß A.21 der Wert 8.

$$\begin{aligned} \hat{N}_k &= \lfloor \frac{\pi}{4} \cdot 5 \cdot 10^6 + 0.5 \rfloor = 3926991 \\ \Rightarrow \hat{\pi} &= 4 \cdot \frac{\hat{N}_k}{\hat{N}_{Ges}} = 3.1415928 \\ \Rightarrow N_{\hat{\xi}} &= 7.3 \overset{\lceil\ \rceil}{\Longrightarrow} N_{\xi_{min}} = 8 \end{aligned} \tag{A.20}$$

A.2.3.3 Testergebnisse und Auswertung

Nach der Testdurchführung erhält man pro Generator für jede Testgröße entsprechend der Anzahl der Stichproben 50 Werte. Die Mittelwerte über diese Stichproben liefern für jede Testgröße eine Näherung des tatsächlichen Wertes. Die Mittelungen sind in Abbildung A.4 aufgetragen. Die einzelnen Näherungen der Testgrößen können hier direkt verglichen werden. Dabei fällt sofort der starke Ausreißer des GSL ranlxd1 Generators bei den Zeittestwerten auf. Dieser Generator benötigt für die Berechnung von 10^7 Zufallszahlen auf dem Testrechnersystem (siehe Abschnitt 2.1, Seite 10) etwa drei Sekunden. Damit liegt diese Zeit weit unter dem Niveau aller anderen Generatoren, die höchstens bei etwa einer Sekunde liegen. Die Anforderung von 0.5 Sekunden hat jedoch nur der

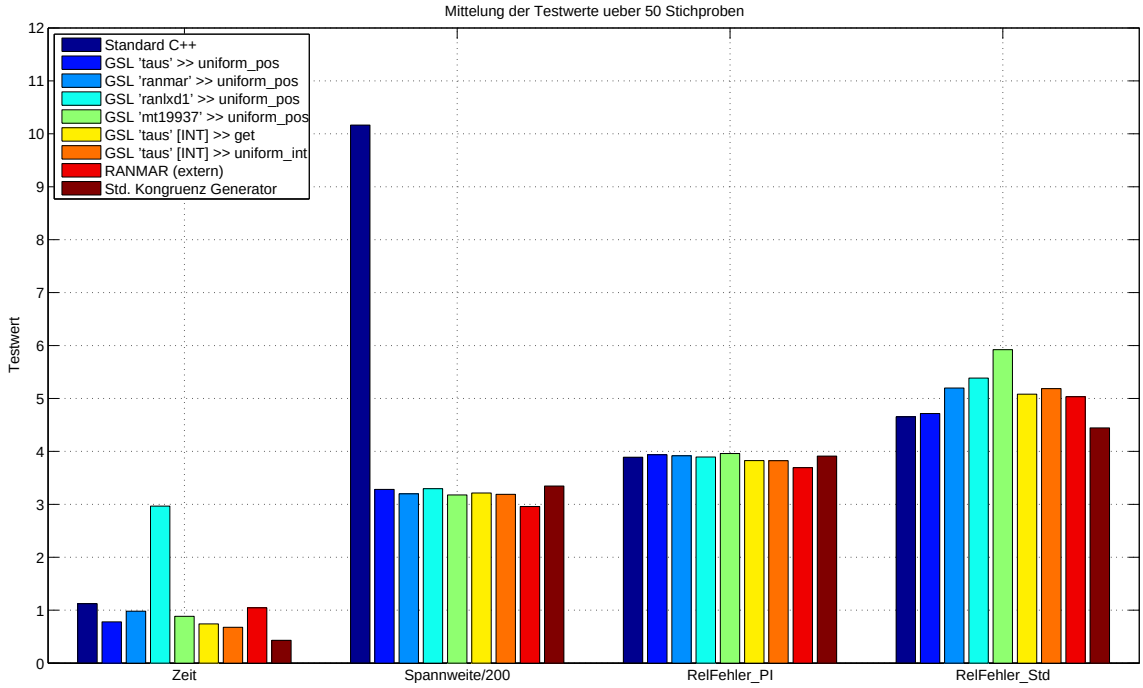

Abbildung A.4: Mittelung der Testwerte über alle 50 Stichproben; die Spannweite wurde mit 200 normiert

Standard Kongruenz Generator erfüllt.

Die Spannweiten liegen bis auf die des *C++ rand()* Generators interessanterweise alle sehr nahe beieinander, wurden in dieser Zeichnung allerdings zwecks Übersichtlichkeit mit 200 normiert. Die tasächlichen Schwankungen sind also höher als es die Zeichnung vermuten lässt, aber dennoch unwesentlich. Das beste Ergebnis lieferte hier der adaptierte CERN Ranmar Generator. Im Vergleich zu allen anderen Generatoren zeigt der *C++ rand()* Generator hier einen extrem hohen Wert. Genauere Betrachtungen haben gezeigt, dass dieser Generator den Wert 0 stark bevorzugt, wohingegen die 1 (bzw. wegen der Abbildung auf das Intervall $[0, 999]$ die 999) entsprechend stark benachteiligt wird. Die Berechnungsmethode wurde von M. Kompf [11] übernommen und entsprechend angepasst. Auch weitere Modifikationen konnten dieses Problem nicht beheben, Programmierfehler konnten ausgeschlossen werden. Der *C++ rand()* Generator muss an dieser Stelle also als untauglich klassifiziert werden.

Die Anzahl der (durch den PI-Test bestimmten) mit π übereinstimmenden Dezimalstellen ergibt nur geringe Unterschieden zwischen den Generatoren. Das Kriterium der Homogenität wird also offenbar von allen Generatoren ähnlich gut erfüllt. Gewissheit muss jedoch die Begutachtung der Rauschbilder liefern, da beim PI-Test eine perfekte Homogenität vorgetäuscht werden kann, wenn insgesamt nur zwei verschiedene Punkte im richtigen Verhältnis zueinander gewürfelt werden. Freilich ist dieser Fall in der Praxis kaum denkbar.

Bei der Standardabweichung zeigten sich sehr hohe Übereinstimmungen mit nur geringen Schwankungen unter den Generatoren. Überraschend ist das Ergebnis jedoch dahingehend, dass sich laut diesem Ergebnis die Häufigkeitsverteilungen der Generatoren untereinander offenbar kaum unterscheiden. Das Auftragen der mittleren Häufigkeitsverteilungen aller Generatoren in einem Diagramm (Abbildung A.5) bestätigt diese Vermutung. Das beste Ergebnis lieferte der *GSL mt19937 [uniform_pos]* Generator. Die Differenz zu den anderen Generatoren beträgt hier zum Teil fast eine ganze Dezimalstelle. Eine deutlich bessere Verteilung ist dennoch nicht zu erwarten, da vor allem der Vergleich der mittleren Häufigkeitsverteilungen zeigt, dass die Wahl des Generators durch andere Kriterien entschieden werden muss.

Genauere Aussagen verlangen daher die Betrachtung aller Stichproben und Kennwerte. In Abbildung A.6 sind für jede Stichprobe und nach Generator gruppiert die einzelnen Kennwertsummen und Anteile aufgetragen (blau: Spannweite, cyan: Zeit, gelb: π Näherung, braun: Standardabweichung). Die Stichproben sind pro Generator jeweils nach der Spannweite sortiert. Die ersten fünf Stichproben des adaptierten Ranmar Generators, deren Spannweite jeweils bei 6 liegt, fallen hier besonders auf. Für die zugehörigen Seeds wurden diesen Daten zufolge also nahezu exakte Gleichverteilungen erzeugt! Zusätzliche Tests ergaben jedoch, dass diese scheinbar hohe Güte aus einem Hyperebenenverhalten resultiert (siehe auch Abbildung A.10). Dadurch erklärt sich auch das (im Mittel) beste Ergebnis der Spannweite für diesen Generator. Da die Seeds zufällig gewählt werden ist die Verwendung des *RANMAR* Generators gefährlich, weil nicht ohne Weiteres bestimmt werden kann, welche Seedwerte ein solches Verhalten verursachen. Die Verwendung an-

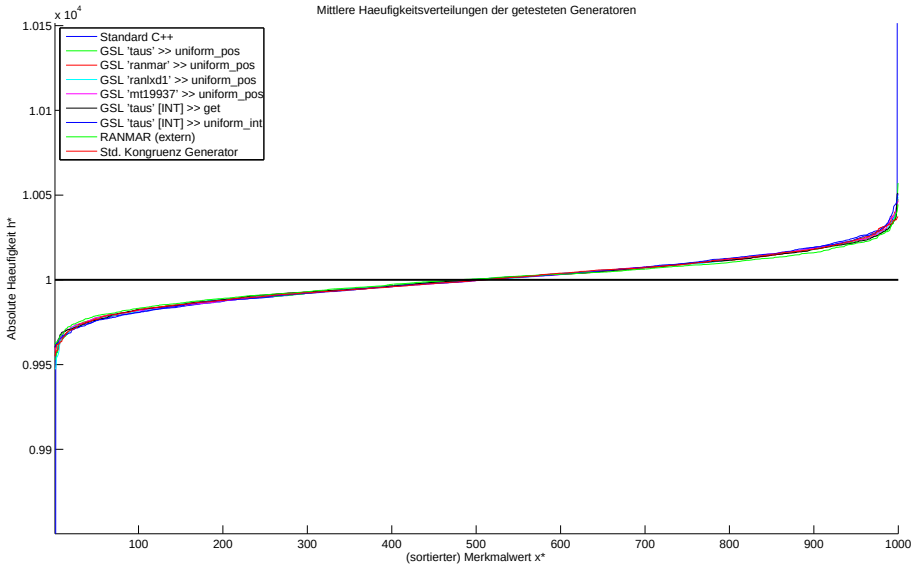

Abbildung A.5: Mittlere Häufigkeitsverteilungen der Generatoren; die Ausreißer des C++ Generators sind treten an den Rändern deutlich hervor

derer Generatoren hat also in jedem Fall Vorrang.

Durch die Sortierung der Kennwertsummen nach der Spannweite erhält man in den Grafiken eine relativ homogene Grundlinie. Dadurch lassen sich wegen den recht konstanten Rechenzeiten Schwankungen im Kennwert für den PI-Test recht gut ablesen (gelb). Hier zeigen alle Generatoren ein ähnlich stabiles Homogenitätsverhalten, wobei unter anderem der Standard-Kongruenzgenerator besonders gut abschneidet. Weil über diesen jedoch keinerlei gesicherte Informationen bezüglich statistischer Tests vorliegen, kommt eher die Verwendung von GSL Generatoren in Betracht. Die Ergebnisse zeigen hier, dass eine Auswahl anhand des Homogenitätsverhaltens nicht erfolgen kann, sofern die Rauschbilder als akzeptabel eingestuft werden können.

A.2.3.4 Rauschbild Auswertung

Beim PI-Test wurden $5 \cdot 10^6$ Zahlenpaare im Einheitsquadrat gewürfelt. Bei der Auswertung wurden die Koordinaten auf drei Nachkommastellen gerundet und die absoluten Häufigkeiten der so diskretisierten Punkte auf eine 1000×1000 Matrix abgebildet. Zur

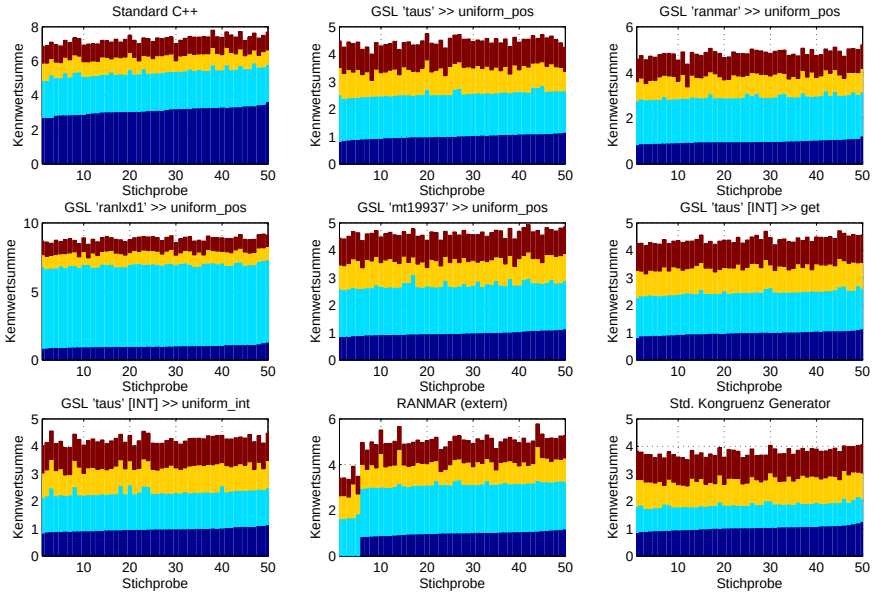

Abbildung A.6: Kennwerte der Stichproben: Spannweite (blau), Rechenzeit (cyan), PI-Test (gelb), Standardabweichung (braun); die Kennwertsummen sind nach der Spannweite sortiert

besseren Auswertung wurde diese Matrix in 10×10 Teilmatrizen zerlegt und deren Mittelwerte in eine 100×100 Verteilungsmatrix übetragen. Sollten Anhäufungen (Cluster) auftreten können diese so besser erfasst werden. Ein Vergleich mit den exakt verteilten Punkten lässt sich dann durch Überlagerung mit der Verteilungsmatrix erzeugen. In Abbildung A.7 sind beide Matrizen in Form von Rauschbildern zu sehen. Trotz der schlechten Spannweite lässt sich im Rauschbild keine Systematik oder auffallende Cluster erkennen. Die Abbildungen A.8 und A.9 zeigen selbiges für den GSL taus [uniform_pos] und den Standard Kongruenzgenerator. Hier zeigen sich, wie auch bei allen übrigen GSL Generatoren, die gleichen Bilder. Die Homogenität ist bei allen getesteten Generatoren also gemäß den Kennwerten sehr gut. Jedoch bildet der adaptierte RANMAR Generator für bestimmte Seeds eine Ausnahme und zeigt stark ausgeprägte Systematiken (Abbildung A.10).

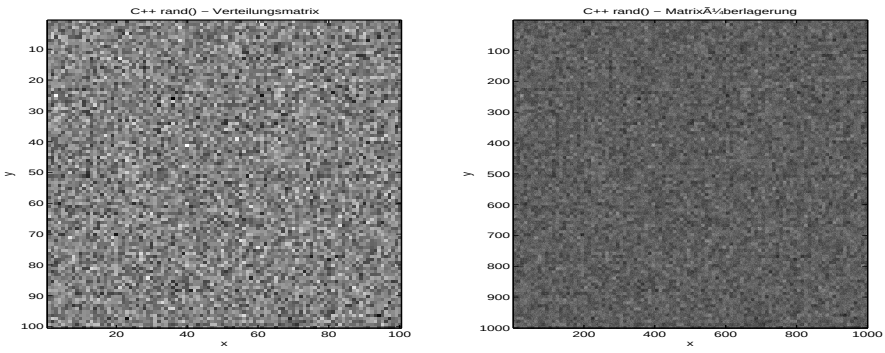

Abbildung A.7: Rauschbild einer Homogenitätsverteilung des C++ rand() Generators; links: Verteilungsmatrix; rechts: Überlagerung

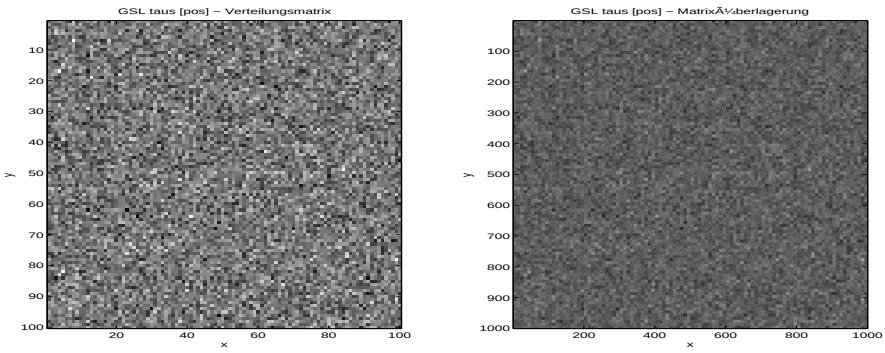

Abbildung A.8: Rauschbild einer Homogenitätsverteilung des GSL taus [uniform_pos] Generators; links: Verteilungsmatrix; rechts: Überlagerung

A.2.3.5 Zusammenfassung der Testergebnisse

Tabelle A.2 enthält eine kurze Zusammenfassung der Testergebnisse. Zu erwähnen ist einerseits, dass für den C++ und den Standard Kongruenzgenerator jeweils keine weiteren Informationen bezüglich ihrer statistischen Eigenschaften gefunden werden konnten. Deshalb kommt deren Verwendung nur im Falle herausragender Testergebnisse gegenüber den GSL Generatoren, für die solche Eigenschaften laut Manual [1] gesichert sind, in Frage. Diese Anforderung wurde jedoch nicht erfüllt, da sich die Kennwerte der verschiedenen Generatoren sehr stark gleichen. Weiterhin wurde vor der endgültigen Test-

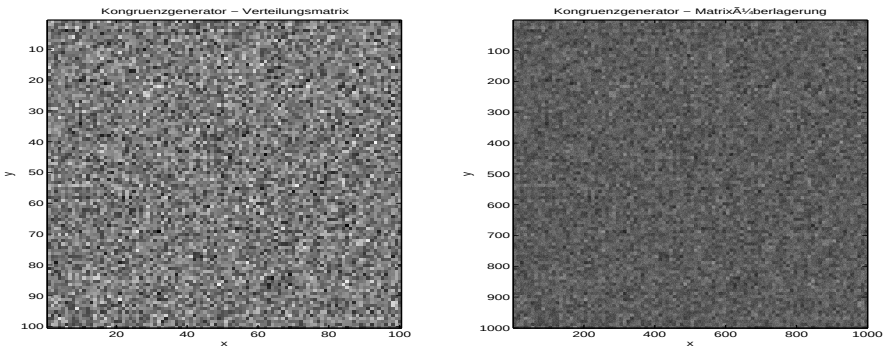

Abbildung A.9: Rauschbild einer Homogenitätsverteilung des Standard Kongruenzgenerators; links: Verteilungsmatrix; rechts: Überlagerung

auswertung in DAO bereits der GSL taus Generator verwendet, da dieser (laut Manual) eine sehr schnelle Berechnung der Zufallszahlen verspricht. Die Zeitkennwerte der Tests bestätigen diese Aussage. Da zwischen den akzeptierten GSL Generatoren keine mar-

Generator	**Akzeptiert**	**Details**
Standard C++	Nein	Spannweite zu groß; keine gesicherten Informationen bezüglich weiterer statistischer Tests
GSL taus [u_pos]	Ja	Gute Rechenzeit; keine Mängel
GSL ranmar [u_pos]	Ja	Nichts zu beanstanden
GSL ranlxd1 [u_pos]	Nein	Rechenzeit zu hoch
GSL mt19937 [u_pos]	Ja	Gute Rechenzeit; gute Häufigkeitsverteilung
GSL taus [int_get]	Ja	Gute Rechenzeit; keine Mängel
GSL taus [u_int]	Ja	Gute Rechenzeit; keine Mängel
RANMAR (extern)	Nein	Gefahr der Systematiken bei zufälligen Seeds
Kongruenzgenerator	Nein	Sehr gute Eigenschaften, aber keine Informationen bezüglich weiterer statistischer Tests vorhanden

Tabelle A.2: Zusammenfassung der Generatortests

kanten Unterschiede festzustellen sind, wurde in DAO die Verwendung des **GSL taus Generators** (mit der Methode **uniform_pos**) beibehalten. Auch für die Simulation gaußverteilter Zufallszahlen versprechen andere Generatoren keine nennenswerten Ver-

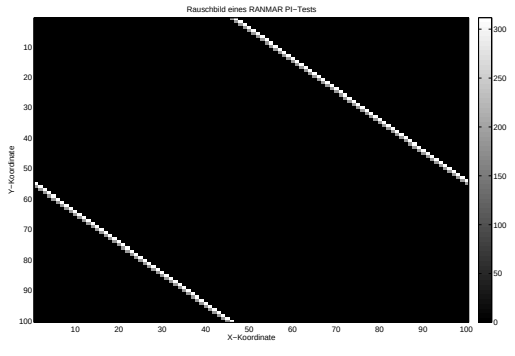

Abbildung A.10: Verteilungsmatrix einer Homogenitätsverteilung des RANMAR-Generators; die Helligkeit der Bildpunkte gibt deren absolute Häufigkeit an

besserungen.

A.3 DAO Programmstruktur

DAO besteht aus einer Vielzahl an Klassen. Die Auflistung und Kurzbeschreibung der von IKO adaptierten Klassen sind Tabelle A.3 zu entnehmen, entsprechend enthält Tabelle A.4 die DAO spezifischen Klassen. Zur Bereitstellung von Parameterwerten und wichtigen Variablen werden Datencontainer verwendet (Tabelle A.5).

A.4 DAO Bildschirmausgabe

In den folgenden Listings (A.3 bis A.7) sind die Bildschirmausgaben von DAO zu sehen. Details dazu sind Abschnitt 2.3.6 zu entnehmen. Dateinamen wurden entfernt.

Bezeichnung	Beschreibung
soIkoKernel	Liest Kernel-Dateien ein und ermöglicht den Zugriff auf deren Daten
soKernel	Ist die Elternklasse von soIkoKernel und enthält dessen virtuelle Funktionsdefinitionen
soRoi	Liest ROI-Definitionsdateien ein und ermöglicht so die Zuordnung der Voxel
soMainDAO	Eine Minimalversion des IKO Hauptprogramms; erstellt die benötigten Klasseninstanzen und steuert den Ablauf des Programms; enthält die `main`-Funktion
dmatrix	Eine auf den Datentyp *double* festgesetzte Kopie der Klasse lmatrix, die mit dynamischen Datentypen arbeitet und die Darstellung von Volumenmatrizen ermöglicht; wurde um einen Konstruktor zum Import von lmatrix Objekten erweitert

Tabelle A.3: Adaptierte IKO Klassen

```
==================================
DAO StandAlone Version 1.3.8 - IKO
==================================

Read Kernel-File:
-----------------
*** NOTE: Big Kernels will be used ! ***
soIkoKernel: Kernel file [...].ker ready for reading: done, found 2082 pixels
Pixel number reset: 2082
soIkoKernel: Kernel file [...].ker ready for reading: done
*** Loading ROI data ... done. ***

Initialize DAO:
===============
Read DAO setup:
---------------
Load DAO setup from file [Std_Prostata.dao]: Successful!

Read VMC file:
--------------
VMC file: [...]
Successful!

Check VMC data:
--------------
Successful!

Read constraint informations:
-----------------------------
Load config for PRIM[Standard]: Successful!
```

Listing A.3: DAO Bildschirmausgabe (Daten einlesen)

Bezeichnung	Beschreibung
DAO	Die Hauptklasse des Programms; enthält den Optimierungs-algorithmus und Methoden zur Berechnung der Zielfunktion und Aufbereitung der eingelesenen Daten
DAO_Calc	Enthält Funktionen zur Berechnung der durch den Optimierungs-Algorithmus bestimmten Datenänderungen (Dosismatrix und Bixelmatrizen)
DAO_Output	Stellt Methoden zur Bildschirmausgabe zur Verfügung, erstellt und füllt Ausgabe- und Protokolldateien
DAO_Rand	Enthält alle zur Erzeugung von gleich- und normalverteilten Zufallszahlen notwendigen Funktionen
DAO_SetupReader	Liest die DAO Konfigurationsdatei ein
ioVMC	Stellt Methoden zum Einlesen und Erstellen von VMC Dateien zur Verfügung; enthält Funktionen zur Umrechnung der Leafkoordinaten
MLCcon	Liest einen per Übergabeparameter definierten MLC-Constraint Datensatz ein; enthält Funktionen zur Überprüfung der MLC-Constraints für per Referenz übergebene Segment-Objekte

Tabelle A.4: DAO Klassen

Bezeichnung	Beschreibung
RadiationPlan	Enthält Informationen wie die Anzahl der Segmente, Beams, etc.
OptimisationSettings	Enthält die vom DAO_SetupReader eingelesenen Paramter
OptimisationValues	Enthält Informationen über die laufende Optimierung (z.b. Akzeptanzrate, spezifische Wärme, etc.); wird von DAO_Output benötigt um Bildschirm- und Dateiausgaben machen zu können

Tabelle A.5: Datencontainer

```
Create segments:
----------------
Create user-defined segment objects ... done.
Number of beams: 6
Number of segments per beam: 10
Number of weight parameters per segment: 2
Number of parameters: 2160
Create non user-defined segment objects:
[..................................................]
Finished.
(Segment weights have been set to 0.10)
Set BixleIDs: ...
Check segments: 49 of 2082 bixels will be ignored.
finished!
```

Listing A.4: DAO Bildschirmausgabe (Segment-Objekte erzeugen)

```
Calculate initial values:
-------------------------
Calculate full dose ... done.
Calculate OF value ... done.
OFval = 59354.4076
Dref  = 324.3355
Time for dose and OF calculation: 2.94 seconds

==================
DAO Init finished!
```

Listing A.5: DAO Bildschirmausgabe (Startwerte berechnen)

```
Processing SA-Initialisation
============================

Iterate(8): [........]
Calculated OF: 59354.4
Real OF: 59354.4

Results:
--------
dH_mean : 735.736
dH_max : 17257
T_start : 29060
Start OFvalue : 78634.9

========================
Initialisation finished!
```

Listing A.6: DAO Bildschirmausgabe (SA Initialisierung)

```
Running SA optimisation:
========================

Data will be recorded !

Iterate(4): [....]

[...]

Current status:
---------------
SegmentWeightZero Ratio = 0
*** New minimal OF: 24394.417758 *** ( >> -1568.09 )
OFmin:  24394.4
OFref:  24394.4

Temperature: 62.5497 > 0.000010
Loops      : 105 of 325
NoImprove  :   0 of 150
NoChange   :   0 of 25

Leaf mod by OF          (Accp|Mod) : 3463 | 4445  Rate: 0.78
Weight mod by OF        (Accp|Mod) :  27 | 136  Rate: 0.20
Unaccepted parameter modifications : 1091  Accp: 0.76
*** Calc OFvalue :   24394.4 ***
*** Real OFvalue :   24394.4 ***
OK!!

Iterate(4): [../
```

Listing A.7: DAO Bildschirmausgabe (SA Optimierung)

```
======================
Optimisation finished!

### Abort because of no change ! ###

Results:
========
OFvalue before: 59354.4
OFvalue after : 20299.2
Current (real) OFvalue  : 20299.2
=> Minimized to 34.2 %
========

Export data:
------------
MLC data exported to file:
[...]/Result.vmc

ROI data exportet to file:
[...]/RoiData.rec

Merge kernels:
==============
soIkoKernel: Kernel file [...].ker_big ready for reading: done, found 2107 pixels
Pixels: 2107
soIkoKernel: Kernel file [...].ker_big ready for reading: done

Calculate big kernel dose parts ... done.

Dose data (ASCII) exportet to file:
[...]/DAO_dose_ascii.d3d
Dose data (BINARY) exportet to file:
[...]/DAO_dose_bin.d3d

Results stored in file:
[...]/Result.rec


Program finished. Total time: 14 h 56 min 27 sec.
```

Listing A.8: DAO Bildschirmausgabe (Ergebnis)

A.5 MLC Constraint Konfigurationsparameter

Tabelle A.6 beschreibt die verschiedenen Constraint-Typen. Die *bordercheck* und *crash* Constraints sind trivial, können jedoch über Parameter deaktiviert werden falls die Nachrüstung mit weiteren Constraint-Typen dies erfordert. Listing A.9 zeigt alle Konfigurationsparameter wie sie in der Constraint-Datei `MLCcon.dao` hinterlegt sind und für sämtliche Optimierungen verwendet wurden.

Constraint	Beschreibung
CON_BORDERCHECK	Leafs dürfen den durch die Startkonfiguration definierten Bereich nicht verlassen
CON_CRASH	Leafs eines Leafpaares dürfen nicht ineinander fahren
CON_PASSING	Benachbarte und gegenüberliegende Leafs dürfen nicht oder nur um eine bestimmte Länge gegeneinander verfahren werden
CON_MIN_AREA	Löcher in den Aperturen dürfen eine minimale Flächenöffnung nicht unterschreiten; die Bildung von Löchern kann verboten werden

Tabelle A.6: Verwendete MLC Hard-Constraints

```
***CONSTRAINT_DEFINITION
-DEVICE_NAME       | PRIM
-CONFIG_NAME       | Standard
*CON_BORDERCHECK
-ENABLED           | 1
*CON_CRASH
-ENABLED           | 1
*CON_PASSING
-ENABLED           | 1
-MAX_PASS_LENGTH   | 0
*CON_MIN_AREA
-ENABLED           | 1
-AREA_LIMIT        | 4
-MUST_LINK_UP      | 1
```

Listing A.9: Konfigurationsdatensatz für MLC Constraints

A.6 Gewicht Constraint Konfigurationsparameter

Die Gewicht Constraints wurden einmalig und empirisch bestimmt. Tabelle A.7 enthält die Beschreibung der Constraint-Parameter, Listing A.10 zeigt die verwendeten Werte.

```
***DAO_PLAN_CONFIG
-ALLOW_ZERO_WEIGHT| F
-WEIGHT_MOD_RATE  | 2
-WEIGHT_PRECISION | 0.01
-MAX_SEG_WEIGHT   | 0.75
-MAX_BEAM_WEIGHT  | 1.25
```

Listing A.10: Konfigurationsdatensatz für Gewicht Constraints

Parameter	Beschreibung
ALLOW_ZERO_WEIGHT	Bestimmt ob Segmentgewichte auf 0 gesetzt werde dürfen
WEIGHT_MOD_RATE	Bestimmt die Vervielfachung der Segmentgewicht Parameter in der Parameterliste
WEIGHT_PRECISION	Auflösung δ der Segmentgewichte
MAX_SEG_WEIGHT	Maximales Einzelsegmentgewicht
MAX_BEAM_WEIGHT	Maximales Beamgewicht (Summe aller Segmentgewichte eines Beams)

Tabelle A.7: Verwendete Gewicht Hard-Constraints

A.7 Optimierungsparameter

Listing A.11 zeigt den Datensatz an Optimierungsparametern, der primär für Prostatafälle verwendet und in der Datei `Std_Prostata.dao` hinterlegt wurde.

```
***DAO_SA_CONFIG
-SA_OF_CHECK_RATE | 50
-SA_MAGIC_FACT    | 2
-SA_LOOP_ITERATE  | 4
-SA_INIT_ITERATE  | 8
-SA_COOL_EXPO     | 0.95
-SA_START_TEMP_LOG| 0.975
-SA_TRIANGLE_HIGH | -1
-SA_TRIANGLE_LOW  | -1
-SA_MAX_ITERATIONS| 325
-SA_MAX_NO_IMPROVE| 150
-SA_MAX_NO_CHANGE | 15
-SA_CHANGE_LIMIT  | 0.0001
-SA_MIN_TEMP      | 0.00001
-SA_SIGMA_WEIGHT  | 1.0
-SA_SIGMA_LEAF    | 5.5
```

Listing A.11: Beispiel für Optimierungsparameter bei Prostatafällen

Parameter	**Beschreibung**
SA_OF_CHECK_RATE	Rate der vollständigen Dosisaktualisierungen (in Loops)
SA_MAGIC_FACT	Faktor der die Vervielfachung der Iterationsschleifen-Durchläufe im Temperaturbereich des magischen Dreiecks angibt
SA_LOOP_ITERATE	Anzahl der Iterationsschleifen-Durchläufe
SA_INIT_ITERATE	Anzahl der Iterationsschleifen-Durchläufe bei der SA Initialisierung (einmalig)
SA_CHANGE_LIMIT	Toleranzbereich λ für Verbesserungen der Zielfunktion (Abbruchbedingung)
SA_COOL_EXPO	Abkühlrate α
SA_TRIANGLE_HIGH	Oberere Grenze des *magischen* Temperaturbereichs
SA_TRIANGLE_LOW	Untere Grenze des *magischen* Temperaturbereichs
SA_MAX_ITERATIONS	Maximal zugelassene Anzahl an Loops (Abbruchbedingung)
SA_MAX_NO_IMPROVE	Maximale Anzahl an zugelassenen Loops, die zu keiner Verbesserung der Zielfunktion führen (Abbruchbedingung)
SA_MAX_NO_CHANGE	Maximal zugelassene Anzahl an Loops innerhalb des Toleranzbereichs für Verbesserungen (Abbruchbedingung)
SA_START_TEMP_LOG	Akzeptanzrate P_{acc} zur Bestimmung der Starttemperatur
SA_MIN_TEMP	Untere Temperaturgrenze (Abbruchbedingung)
SA_SIGMA_WEIGHT	Sigma der Normalverteilung für Gewichtänderungen
SA_SIGMA_LEAF	Sigma der Normalverteilung für Leafänderungen

Tabelle A.8: Verwendete Optimierungsparameter

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis und Glossar

Apertur ..... Die durch den MLC bestimmte Form der Strahlerkopföffnung, Seite 4

Beamlet ..... Der durch einen Bixel der BMP definierte Teilstrahl, Seite 12

Bixel ..... Element der BMP Matrix, Seite 12

Bixelmatrix ..... Binäre 2D-Matrix mit BMP Dimensionierung zur Darstellung einer Apertur, Seite 20

BMP ..... Beam Modulation Plane; Strahlmodulationsebene, Seite 12

CT ..... Computer Tomographie, Seite 11

DAO ..... Direkte Apertur Optimierung, Seite 8

DV-Constraint ..... Gibt für ein ROI Bedingungen vor die dessen Dosis-Volumen Verhältnis betrifft und der Qualitätsverifikation dient, Seite 5

DVH ..... Dosis-Volumen Histogramm, Seite 5

Energie ..... Bezeichnung für den Wert der Zielfunktion bei SA-Optimierungen, Seite 39

Entscheidungsvariablen Die zu optimierenden Eigenschaften eines Systems; werden hier auch als Parameter bezeichnet, Seite 21

Fluenzmatrix ..... 2D-Matrix mit BMP dimensionierung; enthält die Bixelgewichte einer BMP, Seite 14

Fluenzverteilung .... Beschreibt die Intensitätsmodulation des Photonenstrahls einer Einstrahlrichtung, Seite 3

Grundsegment ..... Das in der VMC-Datei für eine Einstrahlrichtung zuerst definierte Segment; wird zur Definition der maximalen Leafpositionen für

alle Segmente dieser Einstrahlrichtung verwendet, Seite 27

GSL ..... GNU Scientific Library, Seite 98

IK ..... Inverser Kernel, Seite 12

IKO ..... Inverse Kernel Optimierung [2], Seite 7

IMRT ..... Intensitätsmodulierte Strahlentherapie, Seite 3

inverser Kernel ..... Eine Volumenmatrix die den Anteil der Dosisdeponationen durch ein einzelnes Beamlet enthält, Seite 12

Leaf ..... Einzelne verfahrbare Komponente des MLCs, Seite 4

Loop ..... Fasst die bei einer konstanten Temperatur durchgeführten Teil-Iterationsschritte zusammen; wird als einzelner Iterationsschritt der DAO Optimierung angesehen, Seite 33

Margin ..... Randbereich um ein ROI als Hilfskontur, Seite 6

MC ..... Monte-Carlo, Seite 11

MLC ..... Multi Leaf Collminator, Seite 4

OAR ..... Organ At Risk (Risikoorgan), Seite 6

Observable ..... In der Physik der formale Name für eine Messgröße; Bei der SA Optimierung eine Variable die während des Optimierungsverlaufs ausgewertet (beobachtet, observiert) wird, Seite 41

OF ..... Objective Function/Zielfunktion, Seite 14

Parameter ..... Wird in Bezug auf die DAO Optimierung äquivalent zu dem Begriff 'Entscheidungsvariablen' verwendet, da diese als Parameter der Zielfunktion betrachtet werden können., Seite 35

Penaltyfaktor ..... Bestrafungs- bzw. Gewichtungsfaktor für die Zielfunktionskomponente einer ROI, Seite 14

PTV ..... Planning Target Volume (Zielvolumen), Seite 6

ROI ..... Region Of Interest (Betrachtete Volumenstruktur), Seite 5

SA ..... Simulated Annealing; stochastisches Optimierungsverfahren, Seite 8

Seed ..... random seed (engl.) bezeichnet einen oder ggf. mehrere Zahlen-

werte die einem Zufallsgenerator als Paramter für dessen Initialisierung übergeben werden, Seite 86

Segment ..... Einzelne zu bestrahlende Aperturen werden jeweils als Segment bezeichnet, Seite 5

Segmentgewicht ..... Beschreibt die Zeitdauer, für die das zugehörige Segment bestrahlt wird, Seite 5

UT ..... Unspecified Tissue (Hilfskontur), Seite 6

Voxel ..... Element einer 3D-Matrix; Volumenelement, Seite 11

Zielfunktion ..... Eine Funktion die ein Gütemaß für die zu optimierende Systemeigenschaft liefert und im Laufe der Optimierung entweder minimiert oder maximiert werden soll, Seite 14

Literaturverzeichnis

[1] *GNU Scientific Library*, 1.10 edition, Sep 2007. Reference Manual.

[2] L. Bogner et al. Application of an inverse kernel concept to Monte Carlo based IMRT. *Medical Physics*, 33(12), Dec 2006.

[3] L. Bogner, J. Scherer, and M. Herbst. An inverse Monte-Carlo optimization algorithm for conformal radiotherapy. *Med. Phys.*, (15):111–119, 1999.

[4] T. Bortfeld. Optimized Planning Using Physical Objectives and Constraints. *Seminars in Radiation Oncology*, 9(1):20–34, Jan 1999.

[5] G. Ehlert. *Simulation und Optimierung*. AFW Wirtschaftsakademie Bad Harzburg GmbH, 2002. Studienbrief.

[6] M. Fippel. Fast Monte Carlo dose calculation for photon beams based on the VMC electron algorithm. *Med. Phys.*, (26):1466–1475, 1999.

[7] S.B. Gelfand and S.K. Mitter. Analysis of Simulated Annealing for Optimization. *Proc. 24th conf. on Decision and Control*, (779), 1985.

[8] S. Gillis, C. De Wagter, J. Bohsung, B. Perrin, P. Williams, and BJ. Mijnheer. An inter-centre quality assurance network for IMRT verification: results of the ESTRO QUASIMODO project. *Radiother Oncol.*, 76(3):340–53, 2005.

[9] M. Hartmann. *IKO - ein Monte-Carlo basiertes inverses Bestrahlungsplanungssystem für die IMRT*. PhD thesis, Universität Regensburg, 2004.

[10] Free Software Foundation Inc. `http://www.gnu.org/software/gsl/`, Jan 2008. [Online; Stand 2. Januar 2008].

[11] M. Kompf. `http://cplus.kompf.de/artikel/random.html`, Dez 2007. [Online; Stand 6. Dezember 2007].

[12] C. Lawrence, J.L. Zhou, and A.L. Tits. *User's Guide for CFSQP Version 2.5*. University of Maryland, College Park, MD 20742, USA, 1997.

[13] N Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *J. Chem. Phys.*, 21:1087, 1953.

[14] D. Mitra, F. Romeo, and A. Sangiovanni-Vincentelli. Convergence and Finit-time Behaviour of Simulated Annealing. Proc. 24th Conf. on Decision and Control, 1985. 761.

[15] Prof. Dr. Ingo Morgenstern. Persönliche Mitteilung, 2008.

[16] M. Rickhey. Untersuchungen zur Anwendbarkeit des inversen Monte-Carlo-Planungssystems IKO zur gezielten Dosismodulation im Tumor und deren dosimetrische Verifikation. Diplomarbeit, Universität Regensburg, Mai 2005.

[17] N. Schmitz and F. Lehmann. *Monte-Carlo-Methoden I.* Mathematical systems in economics. Meisenheim am Glan: Hain, 28 edition, 1976. Erzeugen und Testen von Zufallszahlen.

[18] D.M. Shepard et al. Direct aperture optimization: A turnkey solution for step-and-shoot IMRT. *Medical Physics*, 29(6):1007–1018, June 2002.

[19] S.V. Spirou and C.-S. Chui. A gradient inverse planning algorithm with dose-volume constraints. *Med. Phys.*, 25:321–333, Mar 1998.

[20] M. Wannenmacher, J. Debus, F. Wenz, and J. Bahnsen. *Strahlentherapie*. Springer Verlag, 2006.

[21] R. Zieliński. *Erzeugung von Zufallszahlen.* Deutsch-Taschenbücher. Harri Deutsch, 27 edition, 1978. Programmierung und Test auf Digitalrechnern.

[22] M. Zizler. Einführung in die physikalische optimierung. Vorlesungsskript: Prof. Dr. Ingo Morgenstern, Mai 2007. Fakultät für Physik, Universität Regensburg.

Danksagung

An dieser Stelle möchte ich mich bei allen Mitarbeitern der medizinphysikalischen Arbeitsgruppe der Klinik und Poliklinik für Strahlentherapie der Universität Regensburg bedanken. Insbesondere danke ich meinem Betreuer, Herrn Prof. Dr. Ludig Bogner für seine Unterstützung und Ratschläge. Ich danke ebenso Herrn Prof. Dr. Oliver Kölbl für die Möglichkeit in dieser Abteilung arbeiten zu können. Herrn Prof. Dr. Ingo Morgenstern danke ich für seine Unterstützung und Zusammenarbeit auf dem Gebiet der stochastischen Optmierungsmethoden.

Außerdem möchte ich mich bei meinem Kollegen, Herrn Dipl. Phys. Mark Rickhey für seine Hilfsbereitschaft und Gedult bedanken, die er mir besonders in der heißen Phase meiner Diplomarbeit entgegengebracht hat. Meiner Kollegin, Frau Dipl. Phys. Judith Alvarez-Moret danke ich ebenfalls für ihre Hilfsbereitschaft und Unterstützung.

Besonderen Dank möchte ich auch meinem Professor, Herrn Dr. Hans-Jürgen Wagner ausdrücken, der sich um organisatorische Probleme und meine fachspezifischen Fragen engagiert gekümmert hat. Herrn Prof. Dr. Roland Hornung danke ich im Weiteren für die Bereitschaft, die Zweitkorrektur dieser Arbeit zu übernehmen.

Für die mentale Unterstützung und hilfreichen Ratschläge danke ich außerdem meinen Eltern, die es mir ermöglichen konnten studieren zu dürfen, sowie meinen Freunden und Kommilitonen.

Die Arbeit in der Abteilung für Strahlentherapie hat mir sehr viel Spaß gemacht und ich konnte tiefe Einblicke im Bereich der modernen Onkologie gewinnen. Mir wurde vor allem verdeutlicht, wie sehr die Mathematik in der modernen Medizin ein wichtiges Fundament darstellt.

E R K L Ä R U N G

1. Mir ist bekannt, dass dieses Exemplar der Diplomarbeit als Prüfungsleistung in das Eigentum des Freistaates Bayern übergeht.

2. Ich erkläre hiermit, dass ich diese Diplomarbeit selbständig verfasst, noch nicht anderweitig für andere Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen und Hilfsmittel benützt sowie wörtlich und sinngemäße Zitate als solche gekennzeichnet habe.

Regensburg, den 28.03.2008

.....

Unterschrift