Deep BOO: Automating Beam Orientation Optimization in Intensity Modulated Radiation Therapy.

Olalekan Ogunmolu*, Michael Folkerts*, Dan Nguyen*, Nicholas Gans†, and Steve Jiang*

*Department of Radiation Oncology, UT Southwestern Medical Center,
2280 Inwood Road, Dallas, Texas 75390-9303 , USA
†Department of Electrical Engineering, University of Texas at Dallas,
Richardson, TX 75080, USA
{olalekan.ogunmolu,dan.nguyen,steve.jiang@utsouthwestern.edu}
michael.folkerts@varian.com, ngans@utdallas.edu

Abstract. Intensity-Modulated Radiation Therapy (IMRT) is a cutting-edge tool for treating cancers by aiming radiation to cancer tumor(s) while minimizing radiation to critical structures from a robot's tool frame. Computationally finding the right treatment plan for a target volume is often an exhaustive combinatorial search problem where traditional optimization methods have not yielded real-time feasible results. Aiming to automate the beam orientation and intensity modulation process, we introduce a novel set of techniques leveraging on (i) pattern recognition, (ii) monte carlo evaluations, (iii) game theory, and (iv) neuro-dynamic programming. In this sentiment, we optimize a deep neural network policy which guides monte carlo simulations of promising beamlets. Seeking a saddle equilibrium, we let two fictitious neural network players, within a zero-sum markov game framework, alternatingly play a best response to their opponent's mixed strategy profile during episodes of a two-player markov decision game. During inference, the optimized policy predicts feasible beam angles on test patients' target volumes. This work merges the beam orientation and fluence map optimization subproblems in IMRT sequential treatment planning system into one pipeline. We formally introduce our approach, and present numerical simulation results for coplanar beam angles on prostate cases.

1 Introduction

Intensity Modulated Radiation Therapy (IMRT) is a state-of-the-art cancer treatment method that delivers high-precision x-rays or electron beams by modulating the intensity of the radiation beam from a robot's eye-view, and modifying the geometric field-shape of the resultant beam in order to produce conformal dose distributions around a tumor, typically located in a patient lying in a supine position on a linear accelerator (LINAC) table. Before treatment, a doctor contours the *critical structures* (or tumors) and *organs-at-risk* (or OARs) within a *target volume* (region of the patient's computed tomography (CT) or magnetic resonance (MRI) scan that contains the tumor and other organs), then prescribes doses that must be delivered. Each beam to be delivered consists of beamlets, aimed from the same angle, where each beamlet may be of a different intensity from that

of its neighbors. Radiation intensities are typically delivered from about $5-11$ different beam orientations with multiple collimator units. The process of choosing what beam angle is best for delivering beamlet intensities is termed *beam orientation optimization* (BOO) while the process of determining what intensity meets a prescribed fluence profile by a doctor is termed *fluence map optimization* (FMO).

A good radiation profile "influence" should simultaneously maximize and minimize the dose delivered to tumor(s) and OARs respectively, while producing sharp dose gradients at the transition between tumors and OARs. This problem has a conflicted objective because tumor(s) often intersect(s) with OARs, and the dose deposition's physics changes with each beam orientation so that BOO is a non-convex problem [1], [2] with myriads of possible beam combinations within a robot's phase space. Craft [1] locally tuned a beam angle set within the system's continuous state space using linear programming duality theory, and found that the BOO problem for a simple 2D pancreatic case has about 100 minima when the optimization is warm-started from 100 different beam angles. Building on Craft's work, Bertsimas et. al [3] resolved to a two meta-step algorithm: dynamically selecting beam angle candidates within the phase space via local minimum search with gradient descent, then exploring a different portion of the solution space by taking finite steps of simulated annealing. While [3] use global and local search with improvements in solutions obtained from equispaced angles, this method has the drawback of presenting the objective function as convex and assuming the doctor's preferences can be represented as a set of linear constraints. Jia et. al [4] split the problem into two subproblems: first progressively identifying non-consequential beam angles by evaluating a multi objective function, and then optimizing the fluence on beam angles that are assigned a high confidence by a dosimetric objective function. Heuristic search strategies have also previously developed e.g. [5–7]. Other lines of work have treated IMRT treatment planning as an inverse optimization problem, with techniques ranging from adaptive l_{21} optimization [4], mixed integer linear programming [8–10] and simulated annealing [11, 12].


IMRT TPS setup. ©radiologyinfo.org

The BOO problem can be mathematically formulated as a set cover problem, where given a universe of all candidate beam angles, $\mathcal{U}$, we seek to find from a subset family $\mathcal{S}$ of $\mathcal{U}$, a cover subfamily, $\mathcal{C} \subseteq \mathcal{S}$ whose union is an optimal beam set that meets the doctor's prescription. This is a combinatorics problem where the parameters to be optimized increase to the extent that classical optimization methods are not real-time feasible. When just the gantry (the robot's tool frame) is rotated with respect to the pose of the patient, we have *coplanar beams*, $i.e.$ all other joint angles of the robot are fixed while only the head rotates – resulting in a set of coplanar beams that are swept out by the gantry. In this work, we focus on finding good coplanar beams in BOO problems as more often than not, only coplanar beams are employed when delivering radiation [12]. We resort to an approximate dynamic programming (ADP) formulation to derive *approximately optimal*

policies in high dimensional state spaces. Particularly, we leverage on recent machine learning breakthroughs [13] to guide a Monte Carlo Tree Search (MCTS) [14–16] of promising beam angle candidates by rapidly exploring different parts of the beam space. We reckon that having an automated toolset for choosing desirable beam angles will eradicate the manual search process, and save treatment time.

Contributions: We provide pseudocode that transforms the BOO problem into a game planning strategy. Coupled with neural fictitious self-play, we continually refine the predictions from a neural network policy so as to drive the weights of the network policy to a **saddle equilibrium** [17]. To this end,

- we let a deep neural network model the nonlinear dynamical system and we generate policies that guide MCTS simulations for two players in a zero-sum Markov game
- an MCTS tree lookout strategy guides transition from one beam angle set to another during each markov decision process (MDP) setting for either player
- each player finds a best response strategy to its opponent's average strategy – driving the weights of the network policy toward an approximate **saddle equilibrium** [18].

Note that in contrast to simulated annealing approaches, our approach takes a very complicated nonlinear problem, solves it offline, saves the weights of the neural network, and then at test time, uses the saved weights to predict feasible beam angles. This is the first article, to the best of our knowledge, that provides a comprehensive description of MCTS within the framework of the classic BOO problem.

2 Methods and Materials

Our design philosophy draws inspiration from approximate dynamic programming, optimal control theory, and game theory. For games with perfect information, there is an optimal value function, $v^\star(s)$, that decides the game's outcome for every possible state, $s \in \mathcal{S}$, under perfect play. One could therefore devise a planning strategy that guides the search for optimistic beam angle configurations within the setup's phase space by using a probability distribution, $p(s, a)$, over a set of deterministic *pure strategies* for the tree.

The search for an *approximately* optimal beam angle set is performed by optimizing the parameters of a function approximator ψ, (here, a deep neural network, with multiple residual blocks [19]) which approximates a policy $\boldsymbol{\pi}$, that guides MCTS simulations of 'best-first' beam angle combinations for a sufficiently large number of iterations;. This MCTS performs a sparse lookout simulation, recursively expanding sub-nodes of a tree's root node, and selectively adjusting the beam angle(s) that contribute the least to an optimal fluence within a node's beam set. Successor nodes beneath a terminal node are **approximated** with a value, $v(s)$, to assure efficient selectivity. We maintain a probability distribution over possible states, based on a set of observations and observation probabilities for the underlying MDP. Let us formalize definitions and notations used throughout the rest of this document.

2.1 Notations and Definitions

The state of the dynamical system (setup), will be denoted by $s \in \mathcal{S},$; it is to be controlled by a discrete action $a \in \mathcal{A}$. States evolve according to the (unknown) dynamics $p(s_{t+1}|s_t, a_t)$, which we want to learn. The learning problem is posed within

a discrete-time finite-time horizon, T., while a beam angle combination search task in this setting can be defined by a reward function $r(s_t, a_t)$, that can be found by recovering a policy $p(a_t|s_t;\psi)$ which specifies a distribution over actions conditioned on the state and parameterized by the weights of a neural network, a tensor ψ. Without loss of generality, we will denote the action conditional $p(a_t|s_t,\psi)$ as $\boldsymbol{\pi}_\psi(a_t|s_t)$. Recovering the optimal weights may consist of the maximization problem, $\psi^\star = \arg\max_\psi \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t|\psi)}\left[r(s_t,a_t)\right]$, where the task is to determine an optimal neural network weight tensor $\psi^\star$ that maximize the sum of the respective instantaneous rewards $\sum_t r(s_t,a_t)$ of the policy, $\boldsymbol{\pi}_\psi(a_t|s_t)$. Let us now introduce some definitions.

Definition 1. *A **beam block** is a concatenation of beams, $\{\theta_1, \theta_2, \ldots, \theta_m\}$ as a tensor of dimension $m \times C \times H \times W$ (see Fig. 2) that together with the patient's ct mask form the state, s_t, at time step, t.*

Notation	Definition/Examples	Notation	Definition/Examples
m	dimensionality of a node's beam set, e.g. $m = 5$	n	dimension of discretized beam angles, e.g. $n = 180$ for 4° angular resoultion
$\boldsymbol{\Theta}$	discretized beam angle set e.g. equally spaced angles between 0° and 360°, spaced apart at 4°	$a_t \in \mathcal{A}$	control or action, $a_t \in \mathcal{A}$ at time step $t \in [1, T]$ used in determining the probability of transitioning from a beam angle subset to another within $\boldsymbol{\Theta}$
$\theta^j \subseteq \boldsymbol{\Theta}$	beam angles selected from $\boldsymbol{\Theta}$ e.g. $\theta_k \in \mathbb{R}^m$	$s_t \in \mathcal{S}$	markovian system state at time step, $t \in [1, T]$ e.g. patient contour, beam angle candidates; dimensionality $2,727,936$ to $3,563,520$
γ	discount factor e.g. 0.99	$\boldsymbol{f_\psi}$	parametric function approximator (deep neural network policy) for state s_t
$v_\psi(s)$	value estimate of state, s_t, as predicted by $\boldsymbol{f_\psi}$	$\boldsymbol{p(s)}$	probability distribution over current state, s generated by neural network policy
$Q(s,a)$	action-state values that encode the "goodness" of a beam-angle set, $\theta_k \in \mathbb{R}^m$, where m is the number of beams considered for a fluence generation, e.g. $m = 5$	$\boldsymbol{B}_{X_t}$	a concatenation of beams in consideration at time step, t, as a block of beams being fed to the neural network policy
$\mathcal{D}_{ij}(\theta_k)$	dose matrix containing dose influence to voxel i from beam angle, θ_k, $\forall\, k \in \{1, 2, \ldots, n\}$ where n is range of the beam set $\mathcal{B}$	$\boldsymbol{D}_t$	dose mask for target volume in consideration at time step , t

Table 1: Table of notations commonly used in this article

(a) angle 10° (b) angle 82° (c) angle 166°

Fig. 1: Example 2D gantry angle representations

Definition 2. *Every **node** of the tree, $\boldsymbol{x}$, has the following fields: (i) a pointer to the parent that led to it, $\boldsymbol{x}.p$; (ii) the beamlets, $\boldsymbol{x}_b$, stored at that node where $b = \{1, \ldots, m\}$; (iii) a set of move probabilities prior, $p(s, a)$; (iv) a pointer, $\boldsymbol{x}.r$, to the reward, r_t, for the state s_t; (v) a pointer to the state-action value $Q(s, a)$ and its upper confidence bound $U(s, a)$ (6) (vi) a visit count, $\mathrm{N}(s, a)$, that indicates the number of times that node was visited; and (vii) a pointer $\boldsymbol{x}.child_i$ to each of its children nodes.*

We want an adaptive allocation rule for determining the transition between states such that as the search process progresses, and achieves convergence as the simulation grows, the worst possible bias is bounded by a quantity that converges to zero. So we define the state broadly enough to capture all subjective unknowns that might influence the payoff/reward to be received by a rational decision-making agent, and then leverage on the *upper confidence bound* algorithm of Agrawal [20] to assure an asymptotic logarithmic regret behavior. Since we do not know what node may yield the best bandit, a player might be biased towards always selecting the beams set with the maximum value. Therefore, we attach a regret term $U(n(s))$ to the $Q(s, a)$-value so as to ensure the optimal beam does not evade the simulation $i.e.$ $Q(s, a) - U(n(s)) \leq Q(s, a) \leq Q(s, a) - U(n(s))$; the width of this confidence bound guides the exploration strategy for states that are momentarily unpromising in values but may later emerge as promising states as the number of games evolve in length. Other notations used in the article are delineated in Table 1.

2.2 Data Preprocessing

Here, we describe masks preprocessing before training the network.

Patient Mask. We obtained 77 anonymized patient CT scans and their associated dose matrices. The scans relate to prostate cases used in previous IMRT cancer TP. The prostate data contains scans of six organs, namely the patients' body, bladder, left and right femoral heads, rectum, and planning target volume (PTV) or tumor. Each patient's scan, $\boldsymbol{D}$, is represented in 3D as $N \times W \times H$, where N is the total number of slices, W is the slice width and H is the slice height. We resized each slice to a square-shaped 2D matrix of size 64×64. We generate 3D images that represent the orientation of the gantry with respect to the patient for each discretized beam angle as shown in (Fig. 1).

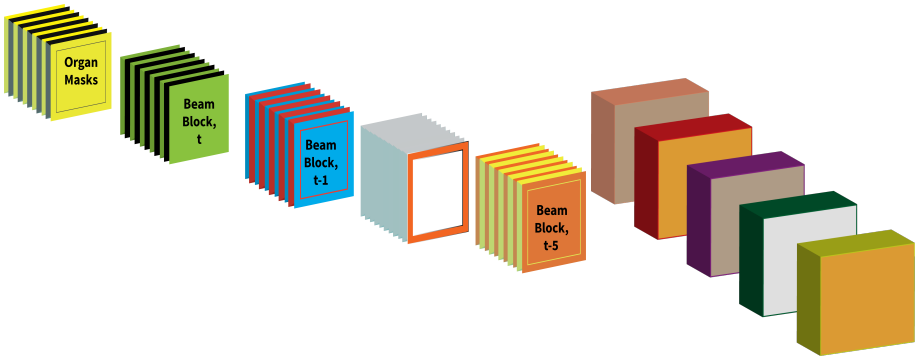


Fig. 2: [Left]: Concatenation of the target volume masks and the beam angles before feeding the input planes to the residual tower neural network. The first six planes (top-most mask of left figure) contain the delineated organs and the PTV. This is concatenated with a block of m beams from the current time step, regressed to the previous 5 time steps (here, 5 was heuristically chosen). [Right] Each beam angle in a beam block is represented as shown. Together with the target volume, these form an input plane of size $36 \times N \times W \times H$ to the policy/value neural network tower of residual blocks.

2.3 Neural Network Architecture

In addition to the resized masks, $\boldsymbol{D}$, we define five feature planes, $\boldsymbol{X}_t$ as a block of beam configurations: $\boldsymbol{B}_{\boldsymbol{X}_t}$, denotes the beam angles that generate the current fluence. For example, if we are considering five beams for the geometric shape for the fluence, $\boldsymbol{B}_{\boldsymbol{X}_t}$ would contain the 3D RGB images of the beams being considered at time step t. We augment the state with a memory of five previously used beam blocks, $\{\boldsymbol{B}_{\boldsymbol{X}_t}, \ldots, \boldsymbol{B}_{\boldsymbol{X}_{t-5}}\}$, during the search process; we make this augmentation due to the partial observability of the patient-gantry setup from a single set of features, in order to mitigate the uncertainty of decisions under this partially observable MDP formulation.

The dose masks and beam blocks are as shown in Fig. 2 so that the input planes to the network are sized as $T \times N \times H \times W$ where T is the total number of input planes ($T = 6$ structures $+ 5$ beams $+ 5 \times 5$ regressed beams $= 36$). Thus, the input to the network are arranged as: $s_t = [\boldsymbol{D}_t, \boldsymbol{B}_{\boldsymbol{X}_t}, \boldsymbol{B}_{\boldsymbol{X}_{t-1}}, \boldsymbol{B}_{\boldsymbol{X}_{t-2}}, \boldsymbol{B}_{\boldsymbol{X}_{t-3}}, \boldsymbol{B}_{\boldsymbol{X}_{t-4}}, \boldsymbol{B}_{\boldsymbol{X}_{t-5}}]$. To address robust learning of patient geometry and beam angle orientations, we use a modern neural network architecture with many residual blocks [21] so that each layer of the network fits a residual nonlinear mapping to its input data; we end up with a deeply stacked network whose input features, s_t, are processed by 34 residual blocks described as follows: (i) a 3D convolution with $64 \times l$ filters, a square kernel of width 7, and double strided convolutions, where l is the depth of the stack in the network; (ii) a 3D batch normalization layer [22]; (iii) nonlinear rectifiers [23]; (iv) a 3D convolution of $64 \times l$ filters; (v) a 3D batch normalization layer; (vi) a skip connection from the input to the block, in order to facilitate efficient gradients' propagation; and (vii) nonlinear rectifiers.

We split the output of the network into two heads: (i) the first head is a probability distribution over which angle in the current beam block contributes the least to an optimal fluence cost at the current time step, while (ii) the second head estimates the *value* (in an

ADP sense) of the subtree beneath the current node This probabilty head is comprised of two residual blocks, each containing: (i) a 3D convolution of $64 \times l$ filters, followed by a square kernel of size 1, and a single strided convolution; (ii) a 3D batch normalization layer; (iii) nonlinear rectifiers; (iv) a 3D convolution of $64 \times l$ filters (v) a 3D batch normalization layer; (vi) a skip connection from the input to the block (vii) nonlinear rectifiers (viii) a fully connected layer that maps the resulting output to the total number of discretized beam angle grids; and (ix) a softmax layer then maps the neuron units to logit probabilities $p_i(s|a)$ for all beam angles.

The value head applies the following transformation modules (i) a 3D convolution of $64 \times l$ filters, a square kernel of size 1, and a single strided convolution; (ii) a 3D batch normalization layer; (iii) nonlinear rectifiers; (iv) a second 3D convolution of $64 \times l$ filters; (v) a 3D batch normalization layer; (vi) a skip connection from the input to the block (vii) nonlinear rectifiers and a sequential layer consisting of:

- a linear layer that maps the output of the above connections to a 512-layer hidden unit
- a linear layer that maps the output of the above connections to a 256 hidden unit followed by rectified nonlinearities
- a linear layer that maps to a scalar value, and followed by rectified nonlinearities
- a tanh nonlinearity that maps the output to the closed interval $[-1, 1]$.

The parameters of the neural network were initialized using the initialization method proposed in [24]. The value and the probability distribution heads are inspired from Bayesian decision theory, where it is expected that a rational decision-maker's behavior is describable by a *utility function*, (or value function) – a quantitative characterization of the policy's preferences for an outcome – and a subjective probability distribution, which describes the policy's beliefs about all relevant unknown factors. When new information is presented to the decision-maker, the subjective probability distribution gets revised. Decisions about the optimal beam angle combination at the current time step are made under uncertainty; so we use a *probability model* to choose among **lotteries**, ($i.e.$ probability distributions over all discretized beam angles in the setup). Each state during our learning process is constructed by appending the beam block at the current time step to a history of beam blocks for the previous five time steps using a FIFO policy. Specifically, when we transition to a new state, the beam block that has been in the state set for the longest time ($i.e.$ at the **head** of the queue) is deleted first, and the new state's beam block is enqueued at the tail as in a **queue** data structure. This is so as to minimize the partial observability of the system.

2.4 Fluence Map Optimization

Suppose $\mathcal{X}$ is the total number of discretized voxels of interest (VOIs) in a target volume, and $\mathcal{B}_1 \cup \mathcal{B}_2 \cup \ldots \cup \mathcal{B}_n \subseteq \mathcal{B}$ represents the partition subset of a beam $\mathcal{B}$, where n is the total number of beams from which radiation can be delivered., if we let $\mathcal{D}_{ij}(\theta_k)$ be the matrix that describes each dose influence, d_i, delivered to a discretized voxel, i, in a volume of interest, VOI_h $(h = 1, \ldots, \mathcal{X})$, from a beam angle, θ_k, $k \in \{1, \ldots, n\}$, then one can compute the matrix $\mathcal{D}_{ij}(\theta_k)$ by calculating each d_i for every bixel, j, at

every φ°, resolution, where $j \in \mathcal{B}_k$. Doing this, we end up with a *sparse* matrix, $\mathcal{D}_{ij}(\theta_k)$, which consists of the dose to every voxel, i, incident from a beam angle, θ_k at every $360^\circ/\varphi^\circ$ (in our implementation, we set φ to 4°).

For a decision variable, x_j, representing the intensities of beamlets, it is trivial to find the dose influence, d_i, that relates the bixel intensities, x_j, to the voxels of interest, VOI_h. The fluence problem is to find the values of x_j for which d_i to the tumor is maximized, while simultaneously minimizing the d_i to critical structures. These objectives are naturally conflicted, and it is typical for dosimetrists to spend hours finding a good set of candidate beams. For the voxels in all OARs, a weighted quadratic objective minimizes the l_2 distance between a pre-calculated dose $\boldsymbol{Ax}$, and a doctor's prescribed dose, $\mathbf{b}$, while a weighted quadratic objective maximizes the l_2 distance between $\boldsymbol{Ax}$ and $\mathbf{b}$. The pre-calculated dose term is given by $\boldsymbol{Ax} = \{\sum_s \frac{w_s}{v_s}\mathcal{D}_{ij}^s \boldsymbol{x}_s \,|\, \mathcal{D}_{ij} \in \mathbb{R}^{n\times l}, n > l\}$, which is a combination of the dose components that belong to OARs and those that belong to PTVs. $w_s = \{\underline{w}_s, \bar{w}_s\}$ are the respective underdosing and overdosing weights for the OARs and PTVs, and v_s represents the total number of voxels in each structure. The cost function is

$$\frac{1}{v_s}\sum_{s\in\text{OARs}} \|(b_s - \underline{w}_s\mathcal{D}_{ij}^s\boldsymbol{x}_s)_+\|_2^2 + \frac{1}{v_s}\sum_{s\in\text{PTVs}} \|(\bar{w}_s\mathcal{D}_{ij}^s\boldsymbol{x}_s - b_s)_+\|_2^2 \tag{1}$$

where the underdosing weights are typically set as $\underline{w}_s = 0$ since it is desirable to deliver as little dose as possible to critical structures, while the overdosing weights are chosen to deliver the prescribed dose to the tumor; $(\cdot)_+$ denotes a Euclidean projection onto the nonnegative orthant $\mathbb{R}_+$. We can succintly write the above objective function, subject to nonnegative bixel intensity constraints, as

$$\min \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|_2^2 \quad \text{subject to } x \geq 0,$$

with the attraction of being differentiable and convex, penalizing dose volumes that exceed a doctor's prescribed constraint. The Lagrangian becomes

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = \min \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|_2^2 - \boldsymbol{\lambda}^T\boldsymbol{x}.$$

The above problem can be solved with dual gradient descent (DGD), but DGD has the drawback that the primal and dual updates are not robust to objective's constraints [25]. The alternating direction method of multipliers (ADMM) [25] tackles the robustness problem by adding a quadratic penalty term to the Lagrangian and alternatingly updating the $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ variables in a "broadcast and gather" process. This turns out to be attractive since we will be solving a large-scale learning problem for the optimal beam angle set combination. Introducing an auxiliary variable $\mathbf{z}$, the above is transformed into the following

$$\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{Ax} - \mathbf{b}\|_2^2, \quad \text{subject to } \boldsymbol{z} = \boldsymbol{x},\ \boldsymbol{z} \geq 0.$$

so that the Lagrangian can be written as,

$$\min_{\boldsymbol{x},\boldsymbol{z}} \frac{1}{2}\|\boldsymbol{Ax} - \mathbf{b}\|_2^2 - \boldsymbol{\lambda}^T(\boldsymbol{z} - \boldsymbol{x}) + \frac{\rho}{2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2, \tag{2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ is a multiplier and $\rho > 0$ is an ADMM penalty parameter. The $\boldsymbol{x}$ subproblem to (2) yields

$$\min_{\boldsymbol{x}} \quad \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{A}^T\boldsymbol{A} + \rho\boldsymbol{I})\boldsymbol{x} + (\boldsymbol{\lambda}^T - \boldsymbol{A}^T\boldsymbol{b} - \rho\boldsymbol{z}^T)\boldsymbol{x},$$

so that the $\boldsymbol{x}$-update (due to the convex quadratic nature of the problem) becomes,

$$\boldsymbol{x}^{k+1} = \left(\boldsymbol{A}^T\boldsymbol{A} + \rho\boldsymbol{I}\right)^{-1}\left(\boldsymbol{A}^T\boldsymbol{b} + \rho\boldsymbol{z}^k - \boldsymbol{\lambda}^k\right). \tag{3}$$

Similarly, the $\boldsymbol{z}$-update for (2) can be found by the z-minimization subproblem

$$\min_{\boldsymbol{z}} -\boldsymbol{\lambda}^T\boldsymbol{z} + \frac{\rho}{2}\|\boldsymbol{z} - \boldsymbol{x}\|_2^2 := \min_{\boldsymbol{z}} \frac{\rho}{2}\|\boldsymbol{z} - \boldsymbol{x} - \frac{1}{\rho}(\boldsymbol{\lambda})\|_2^2.$$

Using the soft-thresholding operator, $S_{\boldsymbol{\lambda}/\rho}$, we find that

$$\boldsymbol{z}^{k+1} = S_{\boldsymbol{\lambda}/\rho}\left(\boldsymbol{x}^{k+1} + \boldsymbol{\lambda}^k\right), \tag{4}$$

where $S_{\boldsymbol{\lambda}/\rho}(\tau) = (x - \boldsymbol{\lambda}/\rho)_+ - (-\tau - \boldsymbol{\lambda}/\rho)_+$. $\boldsymbol{\lambda}$ is updated as

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k - \gamma(\boldsymbol{z}^{k+1} - \boldsymbol{x}^{k+1}), \tag{5}$$

where γ is a parameter that controls the step length. The inverse operation in (3) can be carried out with any iterative solver e.g. conjugate gradient. We use an over-relaxation parameter, $\alpha^k = 1.5$, and set the quadratic penalty to $\rho = 1.5$, in the $\boldsymbol{z}$ and $\boldsymbol{\lambda}$ updates: $\alpha^k\boldsymbol{A}\,\boldsymbol{x}^{k+1} - (1-\alpha^k)\boldsymbol{z}^k$. The stopping criterion is met when the primal and dual residuals are sufficiently small, $i.e.$,

$$r^k = \|\boldsymbol{x}^k - \boldsymbol{z}^k\|_2 \leq \epsilon^{\text{pri}} \quad \text{and} \quad s^k = \| - \rho\left(\boldsymbol{z}^{k+1} - \boldsymbol{z}^k\right)\|_2 \leq \epsilon^{\text{dual}},$$

with,

$$\epsilon^{\text{pri}} = \sqrt{\rho}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}\max\{\|\boldsymbol{x}^k\|_2, \| - \boldsymbol{z}\|_2\}, \text{ and}$$
$$\epsilon^{\text{dual}} = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}(\rho\boldsymbol{\lambda}^{\boldsymbol{k}}),$$

where $\epsilon^{\text{pri}} > 0$, $\epsilon^{\text{dual}} > 0$ are the primal and dual feasibility tolerances for the primal and dual feasibility conditions (see [25, §3.3]). We set $\epsilon^{\text{abs}} = 10^{-4}$ and $\epsilon^{\text{rel}} = 10^{-2}$.

2.5 Game Tree Simulation

Consider b^d possible move sequences of a gantry-patient setup where b are the beam angles chosen to construct a fluence and d is the total number of discretized gantry angles. Suppose $b = 180$ and $d = 5$, we have 180^5 possible search directions, rendering exhaustive search infeasible. We leverage on Monte Carlo simulations, encouraged by their recent success in large games [26–28], to break the curse of dimensionality [29]. To this end, we let a neural network roll-out policy guide the search for a *best-first* beam angle set, where search transitions are governed by probabilities obtained from a two-player zero-sum game of neural fictitious self-play (to be discussed shortly).

Our MCTS proceeds as follows, we iteratively sample beam angles within the setup's phase space by carrying out a lookahead search from the tree's root node at a fixed depth, restricting samples to 90 discretized beams in $\boldsymbol{\Theta}$. At the first iteration, the subset of beam angles are randomly sampled from $\boldsymbol{\Theta}$. We then progressively add children nodes using an **expand_policy** (alg. 1), guided by *move probabilities* $p(s,a)$, generated by a network policy $\boldsymbol{f_\psi}$, that either recursively **expand**s the current node or rolls out the current simulation to completion. Actions $\{a_0,\ldots,a_t\}$ guide transitions in beam changes through the tree's sub-nodes as a **back-up** operator (alg. 1) progressively averages the outcome of many random simulations to min-max through the evolution of the simulation length.

As we recursively traverse the edges of the tree, we need to prevent "angle collisions" $i.e.$ angles overlap during exploration; we therefore introduce a minimum pairwise distance, $\bar{d}_i \in \mathbb{R}^+$ between the beamlets in a beam block, defined as $\|\theta_i - \theta_j\| \geq \bar{d}_i, \forall\{j \in m \setminus i\}$, with $\bar{d}_i = 20^\circ$. Repeatedly performing roll-outs, we maintain a history of state-action value pairs along the tree'e edges – aiding faster convergence if the same state is encountered more than once, because we can bias an action selection based on old actions that were chosen. We compute the mean outcome of every simulation through state s in which action a is selected, $i.e.$ the tree's $Q(s,a)$-value, as $Q(s,a) = \dfrac{1}{\mathrm{N}(s,a)} \sum_{i=1}^{n(s)} \mathrm{I}_i(s,a)\zeta_i$, where $\mathrm{N}(s,a) = \sum_{i=1}^{n(s)} \mathrm{I}_i(s,a)$ is the total number of simulations in which action a was selected in state s, $n(s)$ is the total number of times a game is played through state s, and ζ_i is the outcome of the ith simulation played out from s. Specifically,

$$\mathrm{I}_i(s,a) = \begin{cases} 1, & \text{if } a \text{ was selected on the } i\text{'th policy rollout} \\ 0, & \text{otherwise.} \end{cases}$$

During simulation, each state and action in the search tree are updated as: $n(s_t) \leftarrow n(s_t)+1$; $N(s_t,a_t) \leftarrow N(s_t,a_t)+1$; $Q(s_t,a_t) \leftarrow Q(s_t,a_t)+\pm r(s_t,a_t)$. where $r(s_t,a_t)$ is the reward/cost gained or incurred by the agent after action a in state s_t. After each simulation, a 'best move' for the current beam block is selected; we exponentiate the move probabilities by a temperature slightly larger than unity to encourage diversity in early play; specifically, we compute $p(a|s_0;\psi) = \frac{N(s_0,a)^{1/\tau}}{\sum_b N(s,b)^{1/\tau}}$, where τ is the temperature factor that diversifies the move probabilities. The modified UCT algorithm applied to optimal beam angle selection is presented in algorithm 1.

Definition 3. *A minimizer player is at a **terminal state** if the FMO cost for the beams at a leaf node is greater than the cost of the beams at its direct ancestor node.*

Definition 4. *A maximizer player has reached a **terminal state** if the FMO cost for the beams at a leaf node is less than the FMO cost for the beams at its direct ancestor node.*

Definition 5. *We define an **upper confidence bound**, $U(s,a)$, on $Q(s,a)$ that adds an exploration bonus that is highest for seldomly visited state-action pairs so that the tree*

Algorithm 1 Deep BOO Monte Carlo Tree Search

function **MCTS**(s_0)
  $s_0 \leftarrow \boldsymbol{x}_0(s_0)$
  while `search_time` $<$ `budget` **do**
    $\bar{\boldsymbol{x}} \leftarrow$ **EXPAND_POLICY**($\boldsymbol{x}_0$)
    $\bar{\boldsymbol{x}}.r \leftarrow$ **FMO_POLICY**
    BACKUP($\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}.r$)
  end while
  return **BEST_CHILD**($\boldsymbol{x}_0, c$)
end function

function **SELECT_MOVE**($\boldsymbol{x}, c$)
  if p_1 `to play` **then**
    return $\text{argmax}_{\bar{\boldsymbol{x}} \in \boldsymbol{x}} Q(\bar{\boldsymbol{x}}) + \mathcal{K}(\bar{\boldsymbol{x}})$
  else
    return $\arg\min_{\bar{\boldsymbol{x}} \in \boldsymbol{x}} Q(\bar{\boldsymbol{x}}) - \mathcal{K}(\bar{\boldsymbol{x}})$
  end if
end function

function **EXPAND_POLICY**($\boldsymbol{x}$)
  while $\boldsymbol{x}$ `nonterminal` **do**
    if $\boldsymbol{x}$ `not fully expanded` **then**
      return **EXPAND** ($\boldsymbol{x}$)
    else
      $\boldsymbol{x} \leftarrow$ **BEST_CHILD**($\boldsymbol{x}, c$)
    end if
  end while
  return $\boldsymbol{x}$
end function

function **FMO_POLICY**($\boldsymbol{x}$)
  return $r = -h^{\star}(\boldsymbol{x}(s)|\cdot)$
end function

function **FULLY_EXPANDED**($\boldsymbol{x}, \boldsymbol{d}$)
  $d_i \leftarrow$ `pairwise_distance`($\boldsymbol{x}.s$)
  `min_elem` $\leftarrow \min(\boldsymbol{d})$
  if `min_elem` $< \boldsymbol{d}$ **then**
    return `True`
  else
    return `False`
  end if
end function

function **EXPAND**($\boldsymbol{x}$)
  $\bar{a}$ = **SELECT_MOVE**($\boldsymbol{x}$, c)
  `sample` $\bar{\theta}$ `with` $\boldsymbol{x}.p(s, a)$
  update $\bar{\theta} \leftarrow \bar{\theta} + \bar{a}$
  with $\boldsymbol{\pi}_{t-1}$, `create` $\bar{\boldsymbol{x}}.p(\bar{s}, \bar{a})$
  while `not` $\bar{\boldsymbol{x}} \in \boldsymbol{x}_0$ **do**
    `add` $\bar{\boldsymbol{x}}$ `to` $\boldsymbol{x}$
  end while
  return $\bar{\boldsymbol{x}}$
end function

function **BACK_UP**($\boldsymbol{x}, c$)
  while $\bar{\boldsymbol{x}}$ `not null` **do**
    $N(\bar{\boldsymbol{x}}) \leftarrow \bar{\boldsymbol{x}} + 1$
    $Q(\bar{\boldsymbol{x}}) \leftarrow Q(\bar{\boldsymbol{x}}) + \bar{\boldsymbol{x}}.r$
    $\bar{\boldsymbol{x}} =$ `parent of` $\bar{\boldsymbol{x}}$
  end while
end function

function **BEST_CHILD**($\boldsymbol{x}_0$)
  if p_1 `to play` **then**
    return $\boldsymbol{x}_0[\arg\min$ `children of` $\boldsymbol{x}_0.r]$
  else
    return $\boldsymbol{x}_0[\arg\max$ `children of` $\boldsymbol{x}_0.r]$
  end if
end function

where $\mathcal{K}(\bar{\boldsymbol{x}}) = c\sqrt{\dfrac{2 \ln n(\bar{\boldsymbol{x}}.s)}{N(\bar{\boldsymbol{x}}.s, a)}}$ and $\bar{\boldsymbol{x}} \in \boldsymbol{x}$ implies $\bar{\boldsymbol{x}} \in$ children of $\boldsymbol{x}$

expansion policy selects the action $a^\star$ that maximizes the augmented value:

$$\bar{Q}(s,a) = Q_j(s,a) + c\sqrt{\frac{2\ ln\ n(s)}{\mathrm{N}(s,a)}}, \quad \textit{where } a^\star = \arg\max_a \bar{Q}(s,a). \tag{6}$$

$Q(s,a)$ is the highest average observed reward from node j – encouraging exploitation of the current node, and ln $n(s)$ is the natural logarithm of the total number of roll-outs through state s. The second term in (6) encourages exploration of other beam angles and c is a scalar exploration constant.

Note that (6) is a version of the **UCB1** algorithm [16] and definitions 3 and 4 preserve the neighborhood search for the beam tuning as specified earlier. We continually update the weights of the neural network policy in a separate thread, writing the weights to a shared memory buffer for the MCTS to read from, $i.e.$ the search thread uses the previous iteration of the trained network policy to to run the policy improvement procedure. When angles are at the edges $i.e.$ 0° or 360° and an angle change outside the range $0 \leq \theta \leq 360$ is recommended, we "wrap" around to enforce cyclicity. Note that the **`EXPAND_POLICY`** and **`FMO_POLICY`** procedures of alg. 1 can be seen as a form of Add/Drop simulated annealing as described in [12]. While the **`FMO_POLICY`** procedure returns the node with the optimal fluence cost, the **`BEST_CHILD`** procedure compares the quality of all beam angle sets in the children of the tree's root node.

2.6 Self-Play Neuro-Dynamic Programming

We consider an approximate dynamic programming (ADP) task where we seek to discover the mapping between patient's geometry and a good beam angle combination by maximizing an extrinsic reward signal that is stage-wise informed by the quality of a fluence profile. To this end, we apply optimal control of incompletely known **MDP**s [30]. Our MDP consists of states $s \in \mathcal{S}$, actions, $a \in \mathcal{A}$, transition probability, $\mathcal{P}^a_{ss'}$, and a reward function $\mathcal{R}^a_s$. $\mathcal{P}^a_{ss'}$ governs the evolution of states e.g. from $s \to s'$, while $\mathcal{R}^a_s$ determines a reward after a transitions occur.

In order to find a good saddle-point for the optimization problem, we applied weakened fictitious self-play (FSP) [18] in order to find an *approximate* best response strategy to an opposing agent's mixed strategy in Markov decision games of self-play. Here, an agent continually plays a game against its opponent and the game's outcome is given by averaging the outcome of all individual plays. In our formulation, each player does not necessarily know the strategy of its opponent ahead of time $i.e.$ their security levels do not necessarily coincide. To ensure that equilibrium can be found within *pure strategies*, we let one player act after observing the decision outcome of the other player.

The network, $\boldsymbol{f_\psi}$, predicts a probability distribution over all beam angle configurations, $p_a = p(s,a)$, and a *value*, $v_\psi(s)$ – an estimate that the current beam angle set θ is the optimal beam set among all a tree's nodes. For a game Γ, suppose that $y = \{y_1, \ldots, y_m | \sum_{i=1}^m y_i = 1\}$ and $z = \{z_1, \ldots, z_n | \sum_{i=1}^n z_i = 1\}$ are the respective probability distributions for players p_1 and p_2, defined on the n and $m-$dimensional simplices respectively, the average value of the game will correspond to player p_1 minimizing a cost $\mathcal{J}(y,z) = y^T\ \Gamma z$ and player p_2 maximizing $\mathcal{J}(y,z)$. Each player's action

is governed by a mixed strategy – obtained by adding a Gaussian random walk sequence with standard deviation 2 to the prior probability distribution predicted by the neural network policy or computed by the tree policy; this is then normalized by the sum of the resulting noised distribution. We do this to encourage further exploration and to transform each player's pure strategy to a mixed strategy. As such, a player's mixed strategy is simply a random variable whose values are the player's pure strategies. Players $p_1,$ and p_2's mixed strategies are independent random variables, repeatedly implemented during game simulations such that their actual decisions are based on chance events. However, as the number of times the game is played gets larger, the frequency with which different actions for p_1 and p_2 are chosen will converge to the probability distribution that characterize their random strategies [31, pp.24].

The fictitious self-player samples from previous episodes of the game of self-play at each iteration k. For each ***pure strategy***, $p(s, a)$, predicted by the neural network policy, $\boldsymbol{f_\psi}$, each player makes a decision governed by a probability distribution on the space of its pure strategies. The opposing agent's policy is randomly sampled from a previous checkpoint of the training scheme. The goal of the two-player zero-sum neural fictitious self-play is to drive the network to an approximate saddle equilibrium, via experiential learning, thus guaranteeing that the realization of an approximate optimal value function for the optimal beam angle set that generates a desired fluence/DVH profile.

2.7 End-to-End Optimization

A head of the tower residual block produces a probability distribution over all discretized angles, which is transformed into a mixing strategy by a random walk Gaussian. Each tree node is formed with the prior probability of the network's policy and the patient's state, s_t (including the current and previous five beam block configuration). When a node is expanded and we transition to other nodes, a full FMO is carried out to evaluate the quality of the chosen beamlets. The FMO cost scores the "goodness" of transitive beam blocks. Each training episode involves randomly sampling from the CT dataset, a patient per episode, concatenating the sampled geometry to the previous beam blocks as outlined in Fig. 2, and eliminating the beam that has been longest in the block as in a *FIFO* scheme.

The network policy, $\pi(\cdot|\psi_{t-1})$, and search tree, $\Gamma(\boldsymbol{\pi}_\psi(\cdot))$, were optimized in separate concurrent threads; the network's weights were written to a shared memory map – asynchronously updated via gradient descent – while the tree search thread ran in a parallel thread from a previous iteration of the network policy. The weights' mean-square error are optimized to minimize the partial derivatives of the predicted value with respect to the network weights, ψ as follows:

$$\Delta\psi_{\boldsymbol{v}} = \frac{\partial \boldsymbol{v}_\psi}{\partial \psi}(\boldsymbol{\zeta} - \boldsymbol{v}_\psi(s)).$$

while the network's value head is optimized to minimize the squared average between winner of self-play games, ζ, and the network's predicted value, $v^p_\psi(s)$. At the end of each episodic simulation, we compare the value predicted by either player, average their mixing strategies and update the gradients of the loss with respect to the values based on

the above equation. The value head and its predecessor modules predict the outcome games of self-play by measuring an estimate of the value function$v_\psi(s)$, for winning from the current beams state, s, of self-play games.

At each terminal node of the tree, new search probabilities are computed, exponentiated by the inverse of a temperature parameter (set to 0.98, see § 2.5). Altogether, we minimize the combined loss,

$$l = (\boldsymbol{\zeta} - \boldsymbol{v})^2 - \boldsymbol{\pi}^T \mathbf{log}(\boldsymbol{p}) + \lambda \|\boldsymbol{\psi}\|_2^2, \tag{7}$$

where λ (set to 1.5) controls regularization of the network weights to avoid overfitting. We take the network weights' induced tensor norm (last term of (7) given its robustness to the assymetrical network modular weights), and we weight the cross entropy term by 0.9, and the mean square error (mse) loss by 0.01 to keep the overall loss in the direction of persistent reduction in training error. The cross-entropy and mse loss were weighted based on empirical evaluations of the loss surface. Similarly, we train the probability distribution over current beams by maximizing the similarity between the computed search probabilities $\boldsymbol{\pi}$ and the predicted distribution p (by the search process) with the cross-entropy loss:

$$\Delta\psi_p = \frac{\log \partial \boldsymbol{p}_\psi(a|s)}{\partial \psi}(\boldsymbol{\pi}^T \mathbf{p})).$$

We use 4 search threads, 8 CPUs, and 4 GPUs for the final version of this algorithm.

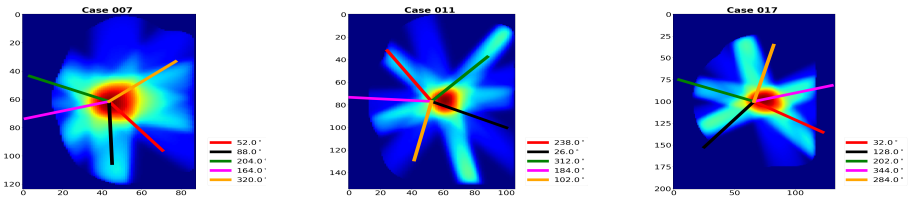
3 Results and Discussion

This section presents results on 3D prostate cases. The results highlighted are for the respective patients' central slices. We provide a visualization of the results in terms of the dose wash plots and dose-volume histograms in Tables 2 and 3. The first results (top half of each of the figures in Tables 2 and 3), are highlights on the dose wash plots and dose-volume histograms for the prostate cases we train the policy on. The bottom-half of the two tables illustrate the results on test data after training. Note that for all of these cases, the tumor is at the middle of the patient's target volume and this is reflected in the dose washes. We use the fractional DVH curves to evaluate the quality of the plan found by our network oracle. The DVH gives a summary of the distribution of simulated doses within a volume of interest in a 2D format – indicating what volume of a structure in consideration received a particular amount of dose.

From Table 2, we see that the algorithm does select beam angles that are fairly equidistant from one another, thanks to the pairwise distance operator that constrained the selectivity of beamlets in during the search strategy; it generally yields dose wash plots that with good dosimetric concentration on the tumor and sharp gradients at the transition between tumors and OARs; it also largely avoids strong dose to OARs. This is desirable in clinical settings. The dose volume histogram plots in Table 3 shows the quantitative dosimetric efficacy of the resulting beams selected by the deep neural network in terms of the volume of dose delivered to tumor volume and the sparing provided to organs-at-risk volumes. We do acknowledge that there edge cases where a strong dose is given to certain OARs before a sharp drop-off in the dose to the particular

Table 2: **Dose wash plots for select patients during training/testing of the self-play network**

Training Regime



Testing Regime

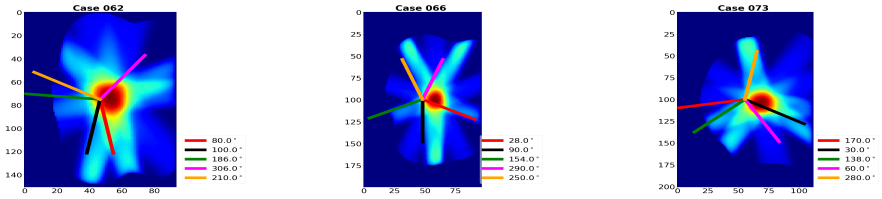
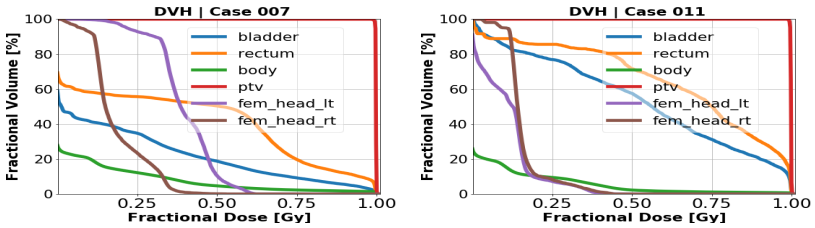
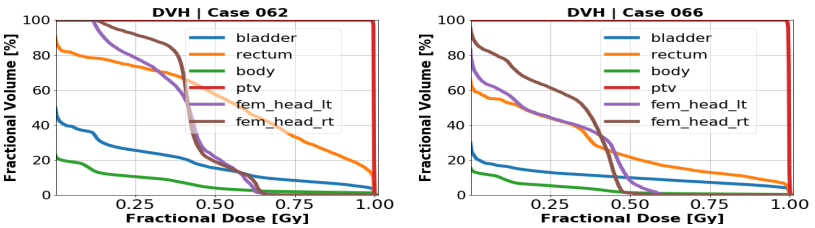


Table 3: **Dose-Volume Histogram For training/testing data**

Training Regime



Testing Regime



organs (e.g. the hot dose to the tumor in case 065). We conjecture that this is due to the network settling on beam blocks with two or more angles that are lfairly close together – hence resulting in this hot dose. For cases such as this, some manual weighting of the

relevant terms in the FMO cost could be introduced after running the network to in order to yield a satisfactory wah plot. While the DVH curves do show a consistent amount of dose to each of the tumor in all of the cases for both training and testing data, we notice for example for cases 7, 65 and 62, the high dose to the femoral heads, femoral heads and rectum, and femoral heads and rectum. Examining carefully the wash plots for these cases, we notice fairly close angles in some of the beamlets that were chosen by the deep policy. It becomes clear that mitigating beamlets that are close in angle to one another is an important challenge in taking this proposal from a simulation setting to the real-world. We are currently investigating this and we will send out our findings as soon as we find a solution.

We conjecture that this could that using the optimal FMO cost as a criteria in judging the goodness of beam angle candidates is not adequate enough as a criterion in the search for best beam angles. In a future work, these limitations will be addressed.

However, the advantage of this neural network policy is that finding the beam angles for a simulated TPS is orders of magnitude faster than the current way these angles are found in the clinic. At test time, we pick a saved checkpoint from a previous iteration of training the neural network, we use it to search with the robot's beam-eye-view. The search process tpically takes between 2-3 minutes before we settle on a good candidate beam angle set. This is a lot of time saved compared to manually tuning beam angles by dosimetrists or radiation therapists in clinics – significantly saving time in treatment time incurred in radiation oncology treatment planning systems.

4 Conclusion

Beam orientation optimization is a key component of conformal IMRT radiotherapy TPS. It has a nonconvex solution surface given the way the dose coefficients can significantly alter based on the gantry angle from which beams are aimed to a target volume. As such, in modern clinics, this problem is typically solved with many hours of planning, hand-tuning and refinement – usually by experienced dosimetrists and radiation therapists.

Given the long time traditional optimization approaches take in solving this problem, we adopt a hybrid approach, leveraging on monte carlo simulation of games in high dimensional state-spaces, neuro-dynamic programming, convex optimization of fluence profiles of a target radiation, and game theory to arrive at good candidate beamlets. Our monte carlo tree search formulation is the first, to the best of our knowledge, that transforms the BOO problem into a monte carlo tree search strategy; and provides a pseudocode that shows how we implement the tree search to navigate the complex state space of respective beamlets. In the part II of this paper, we elucidate on the end-to-end optimization pipeline and provide our initial results based on a set of numerical simulations for select anonymized patients data in our clinic.

References

1. Craft, D.: Local beam angle optimization with linear programming and gradient search. Physics in Medicine & Biology **52**(7), N127 (2007) 2

2. Södertröm, S., Brahme, A.: Optimization of the Dose Delivery In A Few Field Techniques Using Radiobiological Objective Functions. Medical physics **20**(4), 1201–1210 (1993) 2
3. Bertsimas, D., Cacchiani, V., Craft, D., Nohadani, O.: A Hybrid Approach To Beam Angle Optimization In Intensity-modulated Radiation Therapy. Computers & Operations Research **40**(9), 2187–2197 (2013) 2
4. Jia, X., Men, C., Lou, Y., Jiang, S.B.: Beam Orientation Optimization For Intensity Modulated Radiation Therapy Using Adaptive L2,1-minimization. Physics in Medicine and Biology **56**(19), 6205–6222 (2011) 2
5. Bortfeld, T., Schlegel, W.: Optimization of beam orientations in radiation therapy: Some theoretical considerations. Physics in Medicine & Biology **38**(2), 291 (1993) 2
6. Djajaputra, D., Wu, Q., Wu, Y., Mohan, R.: Algorithm and Performance Of A Clinical Imrt Beam-angle Optimization System. Physics in Medicine & Biology **48**(19), 3191 (2003) 2
7. Pugachev, A., Xing, L.: Computer-assisted selection of coplanar beam orientations in intensity-modulated radiation therapy. Physics in Medicine & Biology **46**(9), 2467 (2001) 2
8. Wang, C., Dai, J., Hu, Y.: Optimization of beam orientations and beam weights for conformal radiotherapy using mixed integer programming. Physics in Medicine & Biology **48**(24), 4065 (2003) 2
9. Lim, G.J., Ferris, M.C., Wright, S.J., Shepard, D.M., Earl, M.A.: An optimization framework for conformal radiation treatment planning. INFORMS Journal on Computing **19**(3), 366–380 (2007) 2
10. D D'Souza, W., Meyer, R.R., Shi, L.: Selection of beam orientations in intensity-modulated radiation therapy using single-beam indices and integer programming. Physics in Medicine & Biology **49**(15), 3465 (2004) 2
11. Hou, Q., Wang, J., Chen, Y., Galvin, J.M.: Beam orientation optimization for imrt by a hybrid method of the genetic algorithm and the simulated dynamics. Medical Physics **30**(9), 2360–2367 (2003) 2
12. Aleman, D.M., Kumar, A., Ahuja, R.K., Romeijn, H.E., Dempsey, J.F.: Neighborhood Search Approaches to Beam Orientation Optimization in Intensity Modulated Radiation Therapy Treatment Planning. Journal of Global Optimization **42**(4), 587–607 (2008) 2, 12
13. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature **521**(7553), 436–444 (2015) 3
14. Gelly, S., Silver, D.: Monte-Carlo tree search and rapid action value estimation in computer Go. Artificial Intelligence **175**, 1856–1875 (2011) 3
15. Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. International Conference on Computers and Games (2006) 3
16. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo Planning. European Conference on Machine Learning (2006) 3, 12
17. Ogunmolu, O., Gans, N., Summers, T.: Minimax Iterative Dynamic Game : Application to Nonlinear Robot Control. IEEE International Conference on Intelligent Robots and Systems (2018) 3
18. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International Conference on Machine Learning, pp. 805–813 (2015) 3, 12
19. Huang, G., Liu, Z., Weinberger, K.Q., van der Maaten, L.: Densely Connected Convolutional Networks. arXiv preprint arXiv:1608.06993 (2016) 3
20. Agrawal, R.: Sample mean based index policies by o (log n) regret for the multi-armed bandit problem. Advances in Applied Probability **27**(4), 1054–1078 (1995) 5
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning For Image Recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778 (2016) 6
22. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training By Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015) 6

23. Hahnloser, R.H., Sarpeshkar, R., Mahowald, M.A., Douglas, R.J., Seung, H.S.: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. Nature **405**(6789), 947 (2000) 6
24. He, K., Zhang, X., Ren, S., Sun, J.: Delving Deep into Rectifiers: Surpassing Human-level Performance on Imagenet Classification. In: Proceedings of the IEEE international conference on computer vision, pp. 1026–1034 (2015) 7
25. Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J.: Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers. Foundations and Trends in Machine learning **3**(1), 1–122 (2011) 8, 9
26. Chung, M., Buro, M., Schaeffer, J.: Monte carlo planning in rts games. In: CIG. Citeseer (2005) 9
27. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. nature 529, no. 7587: 484-489. (2016) 9
28. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the Game Of Go Without Human Knowledge. Nature **550**(7676), 354 (2017) 9
29. Bellman, R.: Dynamic programming. Princeton University Press (1957) 9
30. Astrom, K.: Optimal Control of Markov Processes with Incomplete State Information **10**, 174–205 (1965) 12
31. Basar, T., Olsder, G.J.: Dynamic noncooperative game theory, vol. 23. Siam (1999) 13