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 method for treating cancers by aiming radiation to cancer tumor while minimizing radiation to organs-at-risk. Usually, radiation is aimed from a particle accelerator, mounted on a robot manipulator. Computationally finding the correct treatment plan for a target volume is often an exhaustive combinatorial search problem, and 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 (i) pattern recognition, (ii) monte carlo evaluations, (iii) game theory, and (iv) neuro-dynamic programming. We optimize a deep neural network policy that 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 inference, the optimized policy predicts feasible beam angles on test 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 results for coplanar beam angles on prostate cases.

1 Introduction

In this paper, we will present the preliminary results of a multi-disciplinary research project to design a real-time feasible treatment planning optimization in intensity-modulated radiation therapy (IMRT). IMRT is a cancer treatment method that delivers geometrically-shaped, high-precision x-rays or electron beams to tumors by modulating the intensity of the radiation beam. A multileaf collimator shapes a conventional geometrical field, and the intensity of the geometric field shape is varied bixel-wise in order to modulate the "fluence" profile around a tumor. This is done while the patient lies in a supine position on a treatment table. Before treatment, a doctor contours the **critical structures** (or tumors) and **organs-at-risk** (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) and 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 may be delivered from about $5 - 15$ 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). Commonly, the set of beams is constrained to lie in a single plane passing through the critical structures.

When just the robot's tool frame is used to adjust the fluence intensity, we have **coplanar beams**. In this work, we focus on finding good coplanar beams in BOO problems as commonly, only coplanar beams are employed [1]. We consider fictitious self-play as a practical application for developing an effective beam orientation selection strategy in a scenario involving two rational decision-making agents that: i) do not communicate their policies to each other (i.e. the game is non-cooperative), and ii) behave reactively in order to adequately explore the state space.

Contributions: We transform the BOO problem into a game planning strategy. Coupled with neural fictitious self-play, we refine the predictions from a neural network policy to drive the weights of the policy to a **saddle equilibrium** [2]. To this end,

- we devise a tree lookout strategy for games with large state spaces to guide transition from one beam angle set to another;
- the sparse tree lookout strategy is guided by a deep neural network policy, which produces a utility (or value) function that characterizes the policy's preference for an outcome, and a *subjective probability distribution*, which describes the policy's belief about all relevant unknown factors at each time step;
- in a zero-sum two-player Markov decision game of perfect information, either player finds an alternating best response to the current player's average strategy; this is to drive the policy's weights toward an approximate **saddle equilibrium** [3];
- this aids the network in finding an approximately optimal beam angle candidate set that meets the doctor's dosimetric requirements.

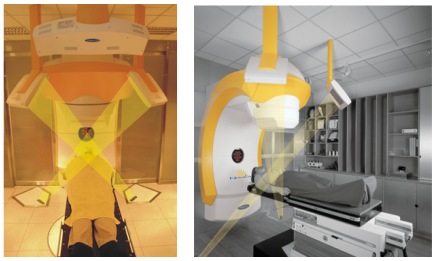
This work presents the first comprehensive description of Monte-Carlo Tree Search (MCTS) within the framework of BOO. It adds new pseudocode that transforms BOO into a game planning strategy. The game recovers an *approximately optimal* set of beamlets and an optimized fluence during TP.

There has been related work on beam orientation optimization. Craft [4] 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 [5] 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 [5] 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 [6] 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. [7–9]. Li et al. [10] used a genetic algorithm fitness metric to select a beams subset from a discrete candidate set of beam angles; a conjugate-gradient dose objective then optimized the intensity of the chosen beams. Other work treat IMRT treatment planning (TP) as an inverse optimization problem, with techniques ranging from adaptive l_{21} optimization [6], mixed integer linear programming [11–13] and simulated annealing [1, 14].

2 Methods and Materials

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 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.



Noninvasive RT. ©Novalis

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 as in [15]) that approximates a policy $\boldsymbol{\pi}$. The policy guides simulations of 'best-first' beam angle combinations for a sufficiently large number of iterations – essentially a sparse lookout simulation that

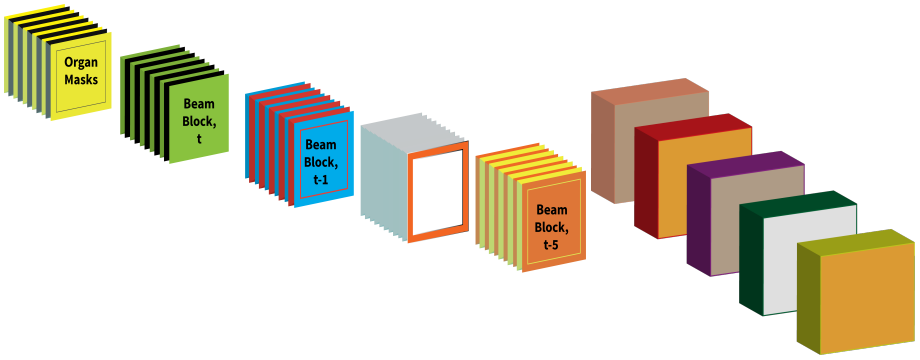


Fig. 1: [Left]: Concatenation of the target volume masks and the beam angles before feeding the input planes to the residual tower neural network. The first six planes (top-most mask of left figure) contain the delineated organs and the PTV. This is concatenated with a block of m beams from the current 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.

selectively adjusts beams that contribute the least to an optimal fluence. 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 observation probabilities for the underlying Markov Decision Process (MDP). Let us now formalize definitions and notations used throughout the rest of this document.

2.1 Notations and Definitions

The state of the dynamical system 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 finite-time horizon, T, while a beam angle combination search task can be defined by a reward function, $r(s_t, a_t)$, which can be found by recovering a policy, $p(a_t|s_t; \psi)$, that specifies

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. $m = 5$	$\boldsymbol{B}_{\boldsymbol{X}_t}$	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 distribution over actions conditioned on the state, and parameterized by the weights of a neural network, a tensor ψ. Without loss of generality, we 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

$$\psi^\star = \arg\max_{\psi} \sum_{t=1}^{T} \mathbb{E}_{(s_t,a_t)\sim p(s_t,a_t|\psi)} \left[r(s_t, a_t) \right].$$

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

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 $U(s, a)$ (7) (vi) a visit count, $\mathbb{N}(s, a)$, 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 since we do not know what node may yield the best bandit, as a player might be biased towards always selecting the beams set with the maximum value. Therefore, 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; we then leverage the *upper confidence bound* algorithm of Agrawal et al. [16] to assure an asymptotic logarithmic regret behavior. We attach a regret term $U(n(s))$ to the Q-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. Other notations used in the article are delineated in Table 1.

2.2 Data Preprocessing

Patient Mask. We obtained 77 anonymized patient CT scans and their associated dose matrices. The scans relate to prostate cases used in previous treatment plans. Six organs are present within the scans: the patients' body, bladder, left and right femoral heads, rectum, and the 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 and H are the respective slice width and 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 robot with respect to the patient for each discretized beam angle.

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, $\boldsymbol{B}_{\boldsymbol{X}_t}$, where $\boldsymbol{B}_{\boldsymbol{X}_t}$ denotes the beam angles that generate the current fluence.

For five beams for the fluence's geometric shape for example, $\boldsymbol{B}_{\boldsymbol{X}_t}$ would contain the 3D images of the beams being considered at time step t. We augment the state with a memory of five previously used beam blocks, $\{\boldsymbol{B}_{\boldsymbol{X}_t}, \ldots, \boldsymbol{B}_{\boldsymbol{X}_{t-5}}\}$, in order to mitigate the uncertainty of decisions under this incomplete MDP formulation.

The dose masks and beam blocks are as shown in Fig. 1. 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: $s_t = [\boldsymbol{D}_t, \boldsymbol{B}_{\boldsymbol{X}_t}, \boldsymbol{B}_{\boldsymbol{X}_{t-1}}, \boldsymbol{B}_{\boldsymbol{X}_{t-2}}, \boldsymbol{B}_{\boldsymbol{X}_{t-3}}, \boldsymbol{B}_{\boldsymbol{X}_{t-4}}, \boldsymbol{B}_{\boldsymbol{X}_{t-5}}]$. We use a modern neural network architecture with many residual blocks [15] 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 [17]; (iii) nonlinear rectifiers [18]; (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; (ii) the second head estimates the *value* of the subtree beneath the current node. The probabilty head is comprised of two residual blocks, each containing the following modules: (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 mapping the output of the above connections to a hidden 512-layer
- followed by a linear layer mapping to a 256 hidden unit, then rectified nonlinearities
- followed by a linear layer mapping to a scalar value, then rectified nonlinearities
- and a `tanh` nonlinearity that maps the output to the closed interval $[-1, 1]$.

The network's parameters were initialized using the proposal in [19]. The value and 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 belief 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. 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, $k \in \{1, \ldots, n\}$. 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 an ill-conditioned *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. For the voxels in the target volume, a weighted quadratic objective minimizes the l_2 distance between a pre-calculated dose $\boldsymbol{Ax}$ (where $\boldsymbol{x}$ represents the vectorized bixels, x_j), 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. Let $w_s = \{\underline{w}_s, \bar{w}_s\}$ represent the respective underdosing and overdosing weights for OARs and PTVs, and v_s represent the total number of voxels in each structure. We define the following cost

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

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

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

The Lagrangian thus becomes

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

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ is a multiplier. This 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 [20]. The alternating direction method of multipliers (ADMM) [20] 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 $\mathbf{z}$, we have

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

so that the Lagrangian can be written as,

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

where $\rho > 0$ is an ADMM penalty parameter. Minimizing (2) w.r.t $\boldsymbol{x}$, the $\boldsymbol{x}$ subproblem of (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

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

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

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

and γ 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,

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

$$\epsilon^{\text{dual}} = \sqrt{n}\epsilon^{\text{abs}} + \epsilon^{\text{rel}}(\rho\boldsymbol{\lambda}^{\boldsymbol{k}}), \tag{6}$$

where $\epsilon^{\text{pri}} > 0$, $\epsilon^{\text{dual}} > 0$ are the primal and dual feasibility tolerances for the primal and dual feasibility conditions (see [20, §3.3]). In this work, 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 robot-patient setup, where b is the number of beam angles chosen to construct a fluence, and d is the total number of discretized beam angle set. Suppose $b = 180$ and $d = 5$, we have 180^5 possible search directions, rendering exhaustive search infeasible. Therefore, we leverage Monte Carlo simulations, encouraged by their recent success in large games [21–23], to break the curse of dimensionality [24]. MCTS combines traditional min-max evaluation of a tree's leaves with Monte Carlo evaluations [25]. MCTS iteratively runs random simulations from a tree's root position through its children nodes by randomly choosing actions until arrival at a terminal state. A back-up operator progressively averages the outcome of many random simulations to min-max as the amount of simulation expands. Thus MCTS solves a min-max [2] problem by default.

We iteratively sample beam angles – performing a lookahead search from the current state at a fixed depth. We restrict samples to 90 discretized beams in $\boldsymbol{\Theta}$. 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.

As we recursively traverse the edges of the tree, we need to prevent "angle collisions". To avoid this, we introduce a minimum pairwise distance, $\bar{d}_i \in \mathbb{R}^+$ between beamlets, defined as $\|\theta_i - \theta_j\| \geq \bar{d}_i, \forall \{j \in m \setminus i\}$, with $\bar{d}_i = 20°$. Repeatedly performing roll-outs, a history of state-action value pairs along the tree's edges is kept. This ensures we can bias an action selection based on old actions that were chosen – 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 $Q(s, a) = \dfrac{1}{\mathbb{N}(s, a)} \sum_{i=1}^{n(s)} \mathbb{I}_i(s, a)\zeta_i$, where $\mathbb{I}_i(s, a)$ is an indicator function given by

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

and $\mathbb{N}(s, a) = \sum_{i=1}^{n(s)} \mathbb{I}_i(s, a)$ 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,

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 as follows, $p(a|s_0; \psi) = \frac{N(s_0,a)^{1/\tau}}{\sum_b N(s,b)^{1/\tau}}$, 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.

Algorithm 1 Deep BOO Monte Carlo Tree Search

function **MCTS**(s_0, c)
  $s_0 \leftarrow \boldsymbol{x}_0(s_0)$
  while search_time $<$ budget **do**
    $\bar{\boldsymbol{x}} \leftarrow$ **EXPAND_POLICY**($\boldsymbol{x}_0$, c)
    $\bar{\boldsymbol{x}}.r \leftarrow$ **FMO_POLICY**($\bar{\boldsymbol{x}}$)
    BACKUP($\bar{\boldsymbol{x}}, \bar{\boldsymbol{x}}.r$)
  end while
  return **BEST_CHILD**($\boldsymbol{x}_0$)
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**($\boldsymbol{x}$, c)
  while $\boldsymbol{x}$ nonterminal **do**
    if $\boldsymbol{x}$ not f.expanded **then**
      return **EXPAND** ($\boldsymbol{x}$, c)
    else
      $\boldsymbol{x} \leftarrow$ **BEST_CHILD**($\boldsymbol{x}$)
    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**($\boldsymbol{x}$, c)
  $\bar{a}$ = **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 $\bar{\boldsymbol{x}} \in \boldsymbol{x}$ **do**
    add $\bar{\boldsymbol{x}}$ to $\boldsymbol{x}$
  end while
  return $\bar{\boldsymbol{x}}$
end function

function **BACK_UP**($\boldsymbol{x}, \bar{\boldsymbol{x}}.r$)
  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**($\boldsymbol{x}$)
  if p_1 to play **then**
    return $\arg\min$ children of $\boldsymbol{x}$
  else
    return $\arg\max$ children of $\boldsymbol{x}$
  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 $\boldsymbol{x}$.

Definition 3. *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*

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

$$\bar{Q}(s,a) = Q_j(s,a) + c\sqrt{\frac{2\ ln\ n(s)}{\mathbb{N}(s,a)}}, \quad \textit{where } a^\star = \arg\max_a \bar{Q}(s,a). \tag{7}$$

$\bar{Q}(s,a)$ 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 (7) encourages exploration of other beam angles and c is a scalar exploration constant.

Note that (7) is a version of the **UCB1** algorithm [26]. 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 Algorithm 1 can be seen as a form of Add/Drop simulated annealing as described in [1]. 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 continually play a zero-sum FSP game between two neural networks. Without loss of generality, we will call the first player, p_1, and the second player, p_2. Player p_1 chooses its action under a (stochastic) strategy, $\pi^{p_1} = \{\pi_0^{p_1}, \pi_1^{p_1}, \ldots, \pi_T^{p_1}\} \subseteq \mathit{\Pi}^{p_1}$ that seeks to minimize the outcome ζ, while p_2's actions are governed by a policy $\pi^{p_2} = \{\pi_0^{p_2}, \pi_1^{p_2}, \ldots, \pi_T^{p_2}\} \subseteq \mathit{\Pi}^{p_2}$ that seeks to maximize ζ in order to guarantee an equilibrium solution for a game without saddle point. $\mathit{\Pi}^{p_i}$ is the set of all possible non-stationary Markovian policies. Each player bases its decision on a random event's outcome – obtained from a **mixed strategy** determined by averaging the outcome of individual plays. Together, both players constitute a two-player **stochastic action selection strategy**, $\boldsymbol{\pi}(s,a) = \{\boldsymbol{\pi}^{p_1}, \boldsymbol{\pi}^{p_2}\}$ that gives the probability of selecting moves in any given state. Suppose the game simulation starts from an initial condition s_0, one may write the optimal reward-to-go value function for state s in stage t, with horizon length T as

$$V_t^*(s) = \inf_{\pi^{p_1} \in \mathit{\Pi}^{p_1}} \sup_{\pi^{p_2} \in \mathit{\Pi}^{p_2}} \mathbb{E}\left[\sum_{i=t}^{T-1} V_t(s_0, f(s_t, \pi^{p_1}, \pi^{p_2}))\right],$$
$$s \in S,\ t = 0, \ldots, T-1$$

where the terminal value $V_T^*(s) = 0,\ \forall\, s \in S$; π^{p_1} and π^{p_2} contain the action/control sequences $\{a_t^{p_1}\}_{0 \leq t \leq T}$ and $\{a_t^{p_2}\}_{0 \leq t \leq T}$. The **saddle point strategies** for an optimal control sequence pair $\{a_t^{p_1^*}, a_t^{p_2^*}\}$ can be recursively obtained by optimizing a state-action

value cost, $Q_t(s,a)$, as follows

$$V_t^*(s) = Q_t^*(s_t, \pi_t^{p_1}, \pi_t^{p_2}) = \min_{\pi^{p_1} \in \Pi^{p_1}} \max_{\pi^{p_2} \in \Pi^{p_2}} Q_t^\star(s_t, \pi^{p_1}, \pi^{p_2}) \\ \forall s_t \in \mathcal{S}, \pi^{p_1} \in \Pi^{p_1}, \pi^{p_2} \in \Pi^{p_2}. \tag{8}$$

such that

$$v_{p_1}^\star \leq v^\star \leq v_{p_2}^\star \quad \forall \{\pi_t^{p_1}, \pi_t^{p_2}\}_{0 \leq t \leq T}.$$

where $v_{p_i}^\star$ are the respective optimal values for each player. $Q(s,a)$ can be recovered from the cummulative reward function, $R(s,a)$ and probability transition function, $P(s,a)$ as

$$Q_t^*(s_t, \pi_t^{p_1}, \pi_t^{p_2}) = R(s,a) + \gamma \sum_{x \in S} P(s,a)(x) V_{t+1}^\star(x).$$

Under ideal conditions, it is desirable to determine the optimal value function under perfect play; however, given the curse of dimensionality for BOO problems, the best we can hope for is an approximately optimal value $v_\psi^\star(s)$ by continually estimating the value function $v_\psi^p(s)$, e.g. , using a policy parameterized by a large function approximator such as deep neural networks $\boldsymbol{f_\psi}$ to approximate the optimal value so that $v_\psi(s) \approx v_\psi^p(s) \approx v^\star(s)$. Here ψ are Lipschitz basis functions that are parameters of the function approximator.

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 may be the optimal beam set. 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. Players $p_1,$ and p_2's strategies are independent random variables, repeatedly implemented during game simulations. 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 [27, pp.24].

The network policy, $\pi(\cdot|\psi_t)$, and search tree, $\Gamma(\boldsymbol{\pi}_\psi(\cdot))$, are optimized in separate concurrent threads; to assure non-blocking of search and network optimization processes, the network's weights were written to a shared memory map, where they are asynchronously updated by gradient descent, while the tree search thread ran in a parallel thread from a previous iteration of the network policy, $\pi(\cdot|\psi_{t-1})$. At the end of a full MDP iteration, we compare the value predicted by either player, average their mixing strategies and update the gradients of the loss function with respect to the *values*.We train the probability distribution over current beams by maximizing the similarity between the computed search probabilities $\boldsymbol{\pi}$ and the predicted distribution $\boldsymbol{p}$ (by the search process) with the cross-entropy loss: $\Delta\psi_p = \dfrac{\log \partial \boldsymbol{p}_\psi(a|s)}{\partial \psi}(\boldsymbol{\pi}^T \mathbf{p})$, and we take the network

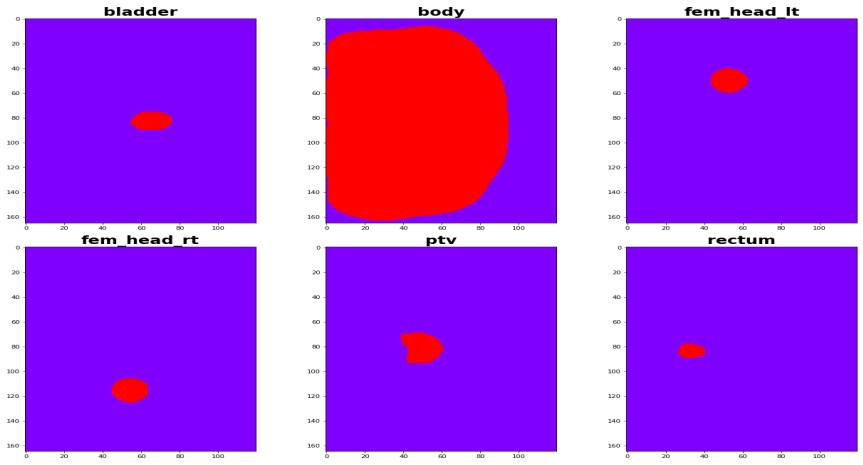


Fig. 2: Separated target volume organs and PTV masks.

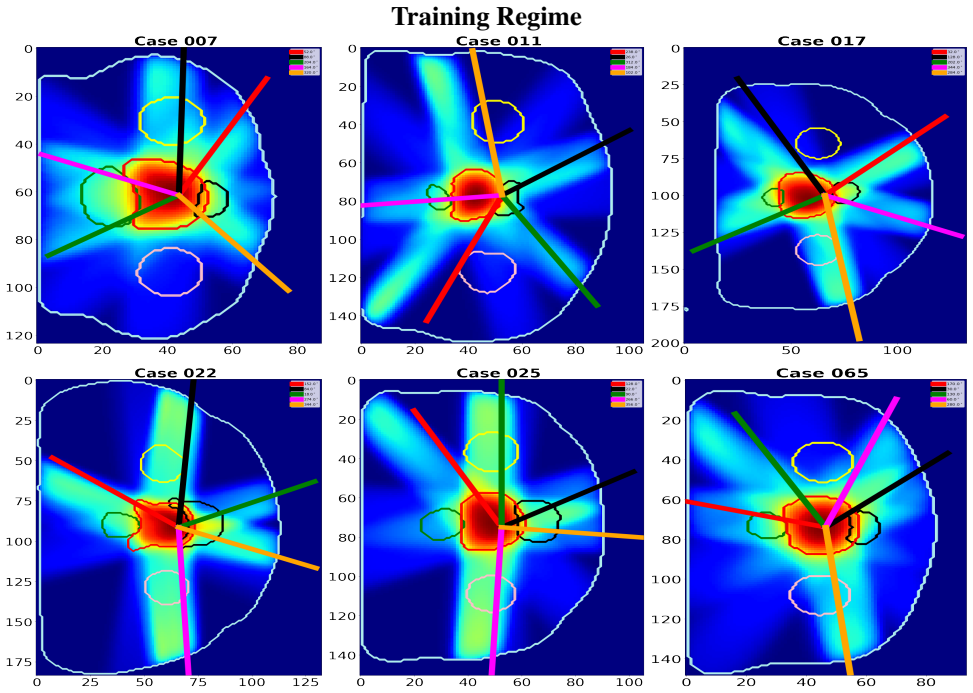
weights' induced tensor norm (given its robustness to the assymetrical network modular weights). Altogether, we minimize the combined loss, $l = (\boldsymbol{\zeta} - \boldsymbol{v})^2 - \boldsymbol{\pi}^T \mathbf{log}(\boldsymbol{p}) + \lambda \|\boldsymbol{\psi}\|_2^2$, where λ (set to 1.5) controls regularization of the network weights to avoid overfitting. The cross entropy term was weighted 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. These values were found by empirical evaluations of the loss surface.

3 Results

This section presents results on 3D prostate cases. We start the training process by randomly adding five beam blocks to the state queue as described in §2. The input planes are then passed through the tower residual network, from which probability distributions and a value are predicted. We add a random walk sequence to this pure strategy, generating a mixed strategy, and subsequently construct the tree. This mixed strategy guides search for approximately optimal beam angles. When a move is selected, the current node is expanded, resulting in a new set of beamlets. The fluence map is found via (1), and the reward for maximizing player, or the cost of the minimizing player are updated. We continue expanding the leaves of the tree until we reach a terminal leaf – when more than one angle are at the same node, in our implementation. As new beam angle combinations are added to the tree, the FIFO queue is updated until the tree search terminates. We then compute new search probabilities and propagate the outcome of the game through the ancestors of the terminal node.

Fig. 2 depicts the arrangement of the respective structures within a prostate case that we consider. Notice that the PTV is at the center of the tumor, and this is consistent with all cases that we do consider. What follows in Table 2 are the dose washes obtained by superimposing the calculated dose (based on the beams that the network selects) on the CT scan map. They provide a qualitative evaluation of the computed dose for a particular case. Represented as a heat map., regions with minimal or no radiation are dark blue,

Table 2: **Dose wash plots for select patients after training the self-play network**



while regions that receive higher doses increase in intensity from green through yellow to red. As seen, the intersection of beams delivers heavy dose to the tumors (center of the slices) while it largely spares surrounding tissues. The line overlays on the plots are the angles of incident radiation.

The policy selects fairly equidistant beams, thanks to the pairwise distance operator that constrained the collision of beamlets during the search process. Hence, we obtain a good dosimetric concentration on the tumor (center of the slices shown) and sharp gradients at the transition between tumors and OARs, while largely sparing OARs.

The advantage of this policy is that finding the right beam angles is orders of magnitude faster than the current way these angles are found in the clinic. At test time, we pick the last checkpoint during training and use it to find feasible beam angles. The search process typically takes between 2-3 minutes before we settle on a good candidate beam angle set. This is significant time saved compared to manually tuning beam angles by dosimetrists or radiation therapists in clinics.

4 Conclusions

Beam orientation optimization is a key component of treatment planning optimization. It has a nonconvex solution surface given the way the dose coefficients can significantly

alter based on the angle from which radiation are aimed to a target volume. As such, in modern clinics, solving this problem involves many hours of planning, tuning and refinement – usually by experienced treatment planners. Given the long time traditional optimization approaches take in solving this problem, we adopt a hybrid approach, leveraging Monte Carlo simulation of games in high dimensional state-spaces, neuro-dynamic programming, convex optimization of fluence profiles, and game theory to arrive at good candidate beamlets. Our work is the first, to the best of our knowledge, that transforms the BOO problem into a MCTS strategy with pseudocode.

References

1. 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, 3, 11
2. Ogunmolu, O., Gans, N., Summers, T.: Minimax Iterative Dynamic Game : Application to Nonlinear Robot Control. IEEE International Conference on Intelligent Robots and Systems (2018) 2, 9
3. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International Conference on Machine Learning, pp. 805–813 (2015) 2
4. Craft, D.: Local beam angle optimization with linear programming and gradient search. Physics in Medicine & Biology **52**(7), N127 (2007) 2
5. 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
6. 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, 3
7. Bortfeld, T., Schlegel, W.: Optimization of beam orientations in radiation therapy: Some theoretical considerations. Physics in Medicine & Biology **38**(2), 291 (1993) 3
8. 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) 3
9. Pugachev, A., Xing, L.: Computer-assisted selection of coplanar beam orientations in intensity-modulated radiation therapy. Physics in Medicine & Biology **46**(9), 2467 (2001) 3
10. Li, Y., Yao, J., Yao, D.: Automatic beam angle selection in imrt planning using genetic algorithm. Physics in Medicine & Biology **49**(10), 1915 (2004) 3
11. 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) 3
12. 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) 3
13. 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) 3
14. 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) 3

15. 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) 3, 6
16. 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
17. Ioffe, S., Szegedy, C.: Batch Normalization: Accelerating Deep Network Training By Reducing Internal Covariate Shift. arXiv preprint arXiv:1502.03167 (2015) 6
18. 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
19. 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) 6
20. 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
21. Chung, M., Buro, M., Schaeffer, J.: Monte carlo planning in rts games. In: CIG. Citeseer (2005) 9
22. 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
23. 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
24. Bellman, R.: Dynamic programming. Princeton University Press (1957) 9
25. Coulom, R.: Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. International Conference on Computers and Games (2006) 9
26. Kocsis, L., Szepesvári, C.: Bandit based Monte-Carlo Planning. European Conference on Machine Learning (2006) 11
27. Basar, T., Olsder, G.J.: Dynamic noncooperative game theory, vol. 23. Siam (1999) 12