Deep Q-learning for simultaneous Beam Orientation and trajectory optimization for Cyberknife

Peyman Kafaei[1,3], Quentin Cappart[2,3], Marc-Andre Renaud[1,3], Nicolas Chapados[1] and Louis-Martin Rousseau[1,3]

[1] Department of Mathematics and Industrial Engineering, Polytechnique Montreal, Montreal, Canada,

[2] Department of Computer Engineering and Software Engineering, Polytechnique Montreal, Montreal, Canada.

[3] CIRRELT-Interuniversity Research Center on Enterprise Networks, Logistics and Transportation, Montreal, Canada

E-mail: peyman.kafaei@polymtl.ca

Abstract. In this paper, we propose a Deep Reinforcement Learning algorithm to find the best beam orientations for radiosurgery treatment planning and particularly the Cyberknife system. We present a Deep Q-learning algorithm to find a subset of the beams and the order to traverse them. A reward function is defined to minimize the distance covered by the robotic arm while avoiding the selection of close beams. Individual beam scores are also generated based on their effect on the beam intensity and are incorporated in the reward function. The algorithm and the quality of the treatment plan are evaluated on three clinical lung case patients. Computational results show a reduction in the treatment time while maintaining the quality of the treatment in comparison with the plan using all of the beams. This results in a more comfortable treatment for the patients and creates the opportunity to treat a higher number of patients in the clinics.

Keywords deep reinforcement learning, Cyberknife, beam orientation optimization, radiation therapy, trajectory

Submitted to: *Phys. Med. Biol.*

1. Introduction

A high-quality radiation therapy treatment requires delivering a prescribed dose to the target volume while sparing normal tissues and organs-at-risk (OARs). The deposited dose needs to closely follow the clinical prescriptions without having a large dose gradient within the tumor. To this end, selection of the best possible set of beams through which the patient is treated with radiation is considerably important.

The problem targeted in this work is to find the best beam orientations for the Cyberknife radiosurgery system (Accuray Inc., Sunnyvale, CA) with a multileaf collimator (MLC). This system delivers noncoplanar beams towards the patient and is used for Stereotactic Radiosurgery (SRS) or Stereotactic Body Radiation Therapy (SBRT). While the Cyberknife system can deliver high-quality treatment plans in terms of dose conformity, the long delivery times of certain Cyberknife plans have been cited as a risk factor [1]. It should be pointed out that more than half of the overall treatment time corresponds to the robotic arm movement between beams and not the beam-on time (actual treatment) during which inadvertent patient movements diminish the treatment quality. The total treatment time might take about one hour [2]. Finding the optimal beam trajectory is challenging as it necessitates considering the total treatment time as well as the final dose distributions.

Non-coplanar plans delivered on standard linear accelerators such as Volumetric Modulated Arc Therapy (VMAT) or Intensity Modulated Radiation Therapy (IMRT) tend to have reasonable treatment times because the treatment field is larger, and there are not as many degrees of freedom as on the Cyberknife. However, the high flexibility of the Cyberknife system requires the consideration of treatment time as a critical factor through meticulously selecting beams and optimizing the trajectory.

In this paper, we propose to use Deep Reinforcement Learning (DRL) to optimize beam orientations for the treatment plans delivered with the Cyberknife system. This method has the benefit to exploit different geometric and dosimetric features to pick the best beams and to simultaneously minimize the delivery time. This directly enables us to consider numerous patient-specific features for beam selection, unlike the common practice that uses a fixed trajectory [3]. The contributions of this paper are as follows:

(i) We propose a Deep Reinforcement Learning algorithm, namely deep Q-learning, to optimize the beam orientations. Experimental results obtained show that this approach can obtain a high-quality solution in a shorter amount of time compared to a standard mixed-integer programming formulation. The final treatment quality is also on par with the treatment using all the possible beams.

(ii) Our proposed solving process leverages both dosimetric and geometric features at the same time. This directly leads to a realistic objective compared with individual beam score methods. In addition, we consider another measure to distribute the selected beams around the patient and avoid having clusters of beams in specific positions around the patient.

This paper is organized as follows. The next section introduces the related works to this problem. Section 3 describes the methods used and the algorithm proposed in this work. The characteristics of the Cyberknife systems and the deep Q-learning method are also discussed in this section. The experimental results and the performance of our proposed method are represented in Section 4 where the implementation, results, and conclusion are discussed.

2. Related Works

Treatment Planning problems can be represented by two main subproblems: Beam Orientation Optimization (BOO) problem and Fluence Map Optimization (FMO) problem. The former finds the optimal subset of the beams to treat the patient through them. The latter computes the amount of the dose delivered at each beam in different organs.

Two major approaches exist to select the most favorable subset of beams. The first approach optimizes the beam orientations and beam fluence maps simultaneously. The beams having positive intensities (positive fluence) in the solution are thus selected in the trajectory. This approach leads to an NP-hard Combinatorial Optimization (CO) problem, meaning there are no known algorithm able to solve it in polynomial time [4]. Although this results in an optimal solution concerning the prescribed dose objectives [5], it entails a major drawback: The computation time may be prohibitive [6]. A partial solution to these issues is to find an approximate solution, instead of the optimal one. This can be attained using local-search methods, such as simulated annealing [7], or by controlling the execution time of mixed integer linear programming models [8]. In addition, some researchers propose methods such as Benders Decomposition [9] or Column Generation [10] to solve the mixed integer mathematical programming formulation of this problem.

A second option is to decouple optimizing beam orientation and beam fluence maps and solve them sequentially. Following this strategy, researchers can exploit some information about the beams before the optimization of the beam intensities. To solve the beam orientation problem, various quality measures of an individual or subsets of beams can be generated [11, 12]. This reduces the computation time despite some approximation errors. The selected beam orientations are then forwarded to another optimization problem to find their respective fluence [13]. Numerous scores based on geometric features of organs and dosimetric characteristics of the candidate beams are defined. Beams are then ranked and selected based on their score. The focus of the methods considering geometric measures is based on the premise that avoiding OARs is essential for an acceptable plan [7, 11]. The overlap between the volumes of the target and the OARs from every beam's-eye-view gives a such metric [14] as well as the position of the OARs with respect to the target value (background or foreground) due to the physical characteristics of the photon beams [14]. The latter assumption neglects the cases where the target volume is surrounded by the OARs as in lung tumors.

Ranking beam orientations based on dosimetric effects have also been considered. Bangert and Oelfke [13] generate locally ideal beam orientations for each target voxel using radiobiological features and conclude that beam orientations typically cluster around distinct positions. They use k-means clustering to select k beams from the potential locally ideal set of beams. In addition to this metric, Yuan et al. [12] define a similarity score for pairwise beam orientations to select more scattered beams iteratively, starting from an empty set of selected beams.

Several researchers have incorporated beam fluence map optimization (FMO) problems into the scalar beam scores to develop more advanced measures of quality. Bangert and Unkelbach [6] have shown that an early stopping mechanism before reaching the optimal solution of this problem is a surrogate for the optimal solution and can thus be used as individual beam metrics. In another study, after selecting beam orientations using solely geometric scores, Smyth et al. [15] run the beam fluence optimization for a few iterations. They then perturb the couch angle for each selected beam and evaluate the FMO to find the best objective function corresponding to the best beams. Starting from every possible beam orientation around the patient, referred to as the 4π plan [16], Langhans et al. [17] solve the FMO problem and eliminate the beams with the total dose of less than the average and repeat this process until $\mathcal{N} = 20$ beams are selected. By assigning different geometric scores to every selected beam orientation, they solve a pathfinding algorithm with a variation of A^* algorithm and output a trajectory to be traversed during the treatment delivery. Nonetheless, one disadvantage is that the angular separation of the selected beams is such that the delivery would not be feasible due to machine restrictions. Lyu et al. [18] follow a similar approach and activate a subset of dosimetrically promising beams from an initial 4π plan. They assign FMO-based individual and pairwise beam costs to find the optimal trajectory.

In this paper, we apply *deep reinforcement learning* (DRL) to solve the beam orientation optimization (BOO) problem. We consider a number of measures in order to optimize beam selection and thereby reduce the treatment time. Even though we decouple BOO and FMO problems, our proposed method uses measures from both geometric features of the beams and dosimetric features related to each patient. Therefore, dose information is taken into account without further complicating the problem. The subset of beams selected by our approach is then forwarded to the *direct aperture optimization* (DAO) module [19] to obtain the intensities of the beams and delivery sequence.

3. Materials and Methods

3.1. The Cyberknife system

The extra degrees of freedom in the Cyberknife system enable a non-coplanar delivery with limited restrictions. The robotic arm can deliver radiation at a discrete set of positions around the patient. Such positions are referred to as *nodes*. In this work, a

node and a *beam* are used interchangeably as we assume that there is only one beam orientation per node. These nodes are uniformly distributed on a sphere centered at the imaging center of the Cyberknife system. Starting from the resting point, the robotic arm can move from the current node to any other node that does not entail a collision with the patient's body. The Cyberknife equipped with a Multileaf Collimator (MLC) allows more flexibility in field shaping and delivers fewer monitor units in comparison with cone collimators [20]. The InCise MLC mounted on the Cyberknife system has 26 leaf pairs, each with a width of 3.85 mm and a maximum field size of 115 $mm \times 100$ mm at 800 mm source-to-axis distance. The radiation field from each beam is discretized into beamlets with the size of twice the MLC leaf width by 5 mm [20] and a dose distribution is generated for each beamlet.

This work assumes a step-and-shoot delivery [21]. Delivery of the radiation is only allowed at nodes. Unlike IMRT and VMAT, the dose rate of the Cyberknife system is constant during delivery and cannot be increased or decreased as a parameter of the device; therefore, the dose deposited in the patient is proportional to the delivery time.

3.2. Deep Reinforcement Learning

Reinforcement learning is a sub-field of machine learning that considers the decision-making dynamics of an agent that interacts with an environment. The goal is to find an optimal sequence of actions that the agent must follow in order to accomplish a given task. Reinforcement learning problems are commonly modeled as a *Markov Decision Process* (MDP) [22].

Let $\langle S, A, T, R\rangle$ be the tuple representing the agent-environment interactions in the proposed RL structure. S is the set of states and A denotes the set of all possible actions. By taking an action the state of the environment changes to a new one following the transition function T. The environment sends a signal to the agent by a deterministic reward function R as a result of selected action. At time step t, the agent receives a representation of the state of the environment $s_t \in S$. Using the current information at s_t, the agent takes an action $a_t \in A$ which gives the state-action pair (s_t, a_t). At the next time step, the state of the environment is updated to $s_{t+1} = T(s_t, a_t)$ and the agent receives a scalar reward of $r_t = R(s_t, a_t) \in \mathbb{R}$. In a deterministic environment, the selection of an arbitrary action in a specific state always results in the same next state. The goal of reinforcement learning is to learn the agent's behaviour policy $\pi : S \to A$, indicating the action to be taken at each state, such that it optimizes the sum of the rewards. The agent visits a sequence of states $s_t \in S$ at each time step $t \in [1, \Theta]$ creating an *episode*. State s_Θ is referred to as the *terminal state*. The expected return after time step t is $G_t = \sum_{k=t+1}^{\Theta} R(s_k, a_k)$. The problem is to find the optimal policy $\pi^* = \text{argmax}_\pi\, Q^\pi(a, s)\ \forall s \in S, \forall a \in A$ where $Q^\pi(a, s)$ denotes the quality of selecting action a at state s under policy π. It is referred to as the Q-values, and, for this environment, we have $Q^\pi(a, s) = G_t$.

For relatively small environments, the optimal Q-values can be computed by an

exhaustive exploration with Dynamic Programming. However, when the problem size increases, Dynamic Programming suffers from the so-called *curse of dimensionality* [23], making an exhaustive exploration not tractable anymore. In order to deal with this issue, Q-learning [24] provides an estimation of the optimal Q-value function for every action selected at specific states. It is computed by successive updates. This value is updated by $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[R_t(s_t, a_t) + \max_{a \in A_t} Q(s_{t+1}, a) - Q(s_t, a_t)\right]$ where α is the learning rate. The aim is thus to find the optimal policy such that the expected total reward gained through successive states is maximized.

In practice, the environments are comprised of an exponential number of states and many of those may not be visited during the previous updates. Neural fitted Q-learning [25] deals with this issue and uses neural networks to approximate the Q-value function. By learning a weight vector $\mathbf{w}$, the model provides an estimator such that $\widehat{Q}(s, a, \mathbf{w}) \approx Q(s, a)$. To this end, an optimizer, such as Adam [26] is used to minimize the squared loss between the current Q-value and the approximated Q-value and update the weight vector: $\mathbf{w} \leftarrow \mathbf{w} - \frac{1}{2}\alpha \nabla L(\mathbf{w})$ where the squared loss is $L(\mathbf{w}) = \left[R(s_t, a_t) + \max_{a \in A_t} \widehat{Q}(s_{t+1}, a, \mathbf{w}) - \widehat{Q}(s_t, a_t, \mathbf{w})\right]^2$. We use prioritized experience replay [27] to stabilize the training.

3.3. Graph Neural Networks

The Beam Orientation Optimization problem can be formulated as a CO problem. Recently, Graph Neural Networks (GNNs) [28] emerged as a machine learning architecture to help solve CO problems efficiently [29]. For any problem that can be represented as a graph, the idea of the GNNs is to compute a vectorial representation of each node by aggregating features of the neighboring nodes [29]. The learned vector representation encodes crucial and latent structures that help to solve challenging CO problems. We formulate the BOO problem as a graph to exploit the benefits of GNNs. As shown in Figure 1, a state graph represents the partial trajectory with the selected beams depicted by full circles. Features are assigned to each node (represented by boxes with colors with different shades). The state graph passes through several layers in the GNNs and at each layer, a new representation of the node features is generated by exploiting the information of the neighbour edges and nodes. At the end, the final node features of the last layer are combined to create a vectorized representation of the state graph.

3.4. Problem Representation

An instance of the BOO problem can be represented by a simple, undirected, and fully-connected graph. Let $G = (V, E)$ be a graph representing a problem instance, where the set of vertices V corresponds to the set of all possible nodes (beams) around the patient for the Cyberknife system. The edges denote the direct path the robotic arm needs to traverse from one node to another. To fully represent the problem, we add distance-

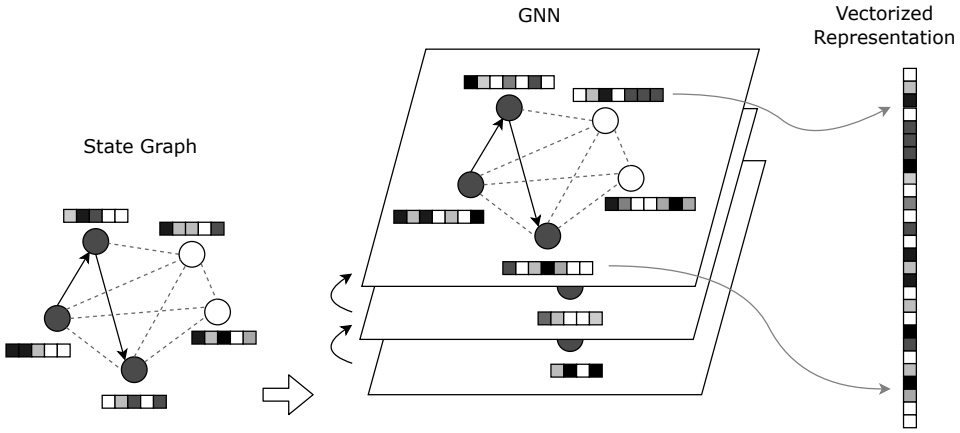

Figure 1. Vectorized representation of the state graph by applying GNNs.

driven and dose-driven features to edges and nodes of the graph, respectively. At each time step, using Graph Neural Networks [30], we obtain an vectorial representation of the current state. This vector is then forwarded to the next steps to learn the Q-values.

3.5. Reinforcement Learning environment for BOO

The RL environment formalizes the problem we want to solve using the aforementioned learning algorithm. Let us consider an initial set of nodes around the patient. Due to the flexibility of the Cyberknife system, the robotic arm can move to an arbitrary node in the following step to irradiate the patient. The movements prone to collision of the robotic arm with the patient can be manually excluded by setting the cost of traversing their path to an arbitrary large value. As such, without loss of generality, collisions are not considered in our implementation. Therefore, at each time step, the set of available nodes includes the ones that have not been selected yet. An episode starts when the robotic arm is at the resting point, and at each time step, it moves towards the next node. The episode finishes when a predefined number of beams ($\mathcal{N}$) are selected [20, 17]. Each movement incurs a reward. Following the structure described by Cappart et al. [31], our RL environment is formally defined as follows:

State At each time t, the state $s_t \in S$ contains the ordered sequence of selected beams denoted by δ_t. A state can be represented as an undirected acyclic graph. The nodes of this graph include following features: (1) the nodes coordinates (x, y, z), (2) the average dose deposited in the OARs, d_{oar} and, (3) the average dose deposited in the target volume d_{tar} at unit intensity. For every structure (OARs, target volumes) that each beam passes through, we have computed the dose deposited in each voxel at unit intensity. We then average this value over the voxels for each individual structure to obtain the average dose deposited in each OAR (d_i, $i \in OARs$) and the target volumes (d_{tar}) separately corresponding to each beam. Each state is transformed into a d-dimensional vector of features ($d = 128$) using GNNs, which serves as the input of a fully connected neural network.

Action At each state s_t, the action $a_t \in A_t$ is defined as the next node to be visited. An action is available only if it is not already included in the trajectory (i.e., $a_t \notin \delta_t$).

Transition The state is updated according to the action performed. By selecting a new action, a new beam is appended to the sequence of the already selected beams. Therefore, $\delta_{t+1} = \delta_t \cup \{a_t\}$.

Reward At each time step t, the agent receives the reward $r_t = R(s_t, a_t)$. Let m be the last beam added to the trajectory in state s_t and the next beam selected is $a_t = n$. The reward is defined as follows:

$$R(s_t, a_t) = -(r_{dist} + r_{dose}), \quad t \neq \Theta \tag{1}$$

$$R(s_\Theta, a_t) = -(r_{dist} + r_{dose} + r_{spread})$$

r_{dist} is the euclidean distance between m and n. r_{dose} for beam n is the ratio $(\sum_{i \in OARs} \omega_i \ d_i)(\omega_{tar} \ d_{tar})^{-1}$ where ω_j is the importance of each structure j. For simplicity we have considered a similar weight for all the structures. r_{spread} is the beam-spread score. The latter accounts for the beam-spread measure among n and all previously selected beams in the trajectory denoted by $\sum_{i,j} K(1-\cos\alpha_{ij})^{-1}$ and $i, j \in \delta_\Theta$. This ensures a maximum separation between selected nodes to produce a plan with higher quality [12, 32].

The importance of each part of the reward can be determined by weighting them; however, in our experiments, we set an equal weight on all terms except for the beam separation which is set by the choice of the parameter K.

3.6. Learning Algorithm

The learning algorithm relies on Deep Q-networks (DQN) presented in Algorithm 1. An agent is used to learn the weight vector ($\mathbf{w}$) of a fully connected neural network to output Q-values. At each iteration of the training phase, a random instance of the problem represented as a graph G is created. Each instance resembles a hypothetical patient. It must be noted that we generate the features to create the state graph. Therefore, for each instance, we generate random coordinates for the nodes from which the beams are delivered towards the patients. These values are normalized to be in the range of $(0, 1)$. The other important measure to shape random instances during training is the ratio of the dose delivered to OARs to the dose delivered in the target volume for each beam. To compute this, for each beam, we need the information on the dose deposited in every OAR ($d_i, \ i \in OARs$) and the target volume (d_{tar}) that the beam passes through. Prior to the training, for all the real patient data we calculated these values as follows. For every structure (OARs, target volumes) that each beam passes through, we compute the dose deposited in each voxel at unit intensity. We then average these values over the voxels of each OAR and the target volume to obtain the values of $d_i, \ i \in OARs$ and d_{tar}. We then add white Gaussian noise with zero mean and square coefficient of variation equal to 0.25 to augment the dataset of dose to OARs and dose to target. Following this, new instances has been added to the dataset for the training purpose [33]. Then

during the training phase, the values of $d_i,\ i \in OARs$ and d_{tar} are randomly selected from this augmented dataset. We thereby compute the ratio of $(\sum_{i \in OARs} d_i)(d_{tar})^{-1}$ for each possible beam. An episode is then constructed by selecting one beam at each time step t using the model's deep neural network architecture with the current weight vector. To make the training more robust, we exploit Prioritized Experience Replay [27]. At each time step, an experience tuple $e_t = (s_t, a_t, r_t, s_{t+1})$ and its corresponding significance are added to the memory for further use in the training.

Always greedily selecting the action with the maximum immediate reward generally leads to inferior solutions due to the lack of exploration. Even following the action with the best approximated Q-value could lead to local minima if no exploration is used. To ensure a balanced exploitation-exploration trade-off, actions are chosen following a softmax action selection strategy. This is also superior to the ϵ-greedy strategy where it chooses equally among all available actions while exploring [31]. In the softmax action selection strategy, the greedy action still has the highest probability while others are weighted according to their value estimates. By adjusting a temperature hyper-parameter, the learning begins with higher exploration followed by increasingly favoring exploitation. At the beginning of the training, the temperature is set to 0, and it increases to a predefined maximum value as the training goes on.

Gradient-based optimizers tend to have difficulties if the rewards are large, sparse, or too small. To avoid such cases, we use scaling to map the reward space into an interval close to zero [31]. Let $\gamma \in \mathbb{R}$ be the scaling factor which is dependant on the value of the coordinates of the nodes. The rescaled reward at time step t is $\gamma\, r_t$.

We sample a batch from the stored experiences to learn the model for estimating optimal Q-values for every state-action pair. For each experience comprising the batch, we obtain an embedding (vectorized representation) of the state as illustrated in Figure 1 and then pass the embedding into a neural network called the *policy network* as an input, shown in Figure 2. The aim of the policy network is to approximate the optimal policy by finding the optimal Q-values for state-action pairs. We have such a network associated with each possible action from the input given state. The output of these networks is the estimated Q-value for each available action from the state s_t. It must be pointed out the output layer of this neural network presents the action space of the proposed RL algorithm. Each output unit (action) corresponds to a node of the state graph (a beam). The beams that are currently included in the partial trajectory are masked, illustrated by full circles in Figure 2, and cannot be chosen. Once the next beam is selected, the trajectory will be updated the state transitions to a new one as in Figure 2 (b).

At this point, the loss needs to be calculated by comparing the Q-value estimated from the policy network for the action a_t of the experience tuple in the batch and the corresponding (target) Q-value for the same action denoted by $q^*(s_t, a_t) = E[R_{t+1} + \max_{a'} q^*(s_{t+1}, a')]$ and $a' \in A_{t+1}$. The $\max_{a'} q^*(s_{t+1}, a')$ needs to be approximated. The target Q-values are obtained from a completely separate network cloned from the policy network named the target network with its weights are frozen with the original

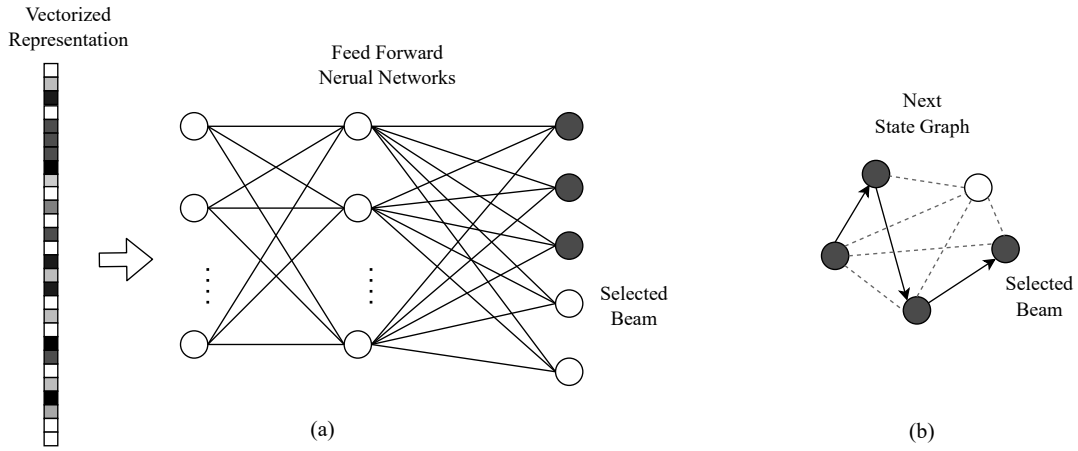

Figure 2. Action Selection. (a) Passing the embedding of the state through a Feed Forward Neural Network. (b) Updated trajectory after the action (selected beam) is performed. The state graph is transitioned into a new state.

policy network's weights. The weights of the target network are updated to the policy network's new weights every certain amount of the time steps. This target network enables us to estimate the maximum Q-value for the next state s_{t+1} in the experience tuple to get the target Q-value $q^*(s_t, a_t)$.

3.7. Neural Network Architecture

Selecting the beams to incorporate in the final trajectory for the robotic arm movement of the Cyberknife system depends on the nodes of the underlying graph G. Therefore, to capture all the node and edge features, we employ a Graph Attention Network architecture to embed the graphs [34]. The policy network is a fully connected feed-forward network with three hidden layers, consisting of 128, 64, and 32 hidden units, respectively. The input of this network is the embedded current state. At each stage, we have one such network for every available action. The output of the policy network is the prediction of the Q-value for the current state-action pair. We use the *Rectified Linear Units* (ReLu) as the activation function of the hidden layers and no activation at the output layer.

3.8. Solving Algorithm

Once trained, the model can be reused to obtain a trajectory for unseen instances. First, the instance is represented as a graph, and the relevant features are extracted. The features of each state s are inputted to the deep neural networs architecture of the model to estimate the value of the state-action function $\widehat{Q}$ for all feasible actions. The next node is selected following the greedy policy $\pi = \text{argmax}_{a \in A} \widehat{Q}(s, a)$. The node incurring the maximum Q-value is inserted into the list of selected nodes until a predetermined number of nodes are selected.

Algorithm 1 Training

▷ $\mathcal{I}$ is the number of iterations
▷ Θ is the length of the episode
▷ N is the mini-batch size
▷ e_t is the experience tuple (s_t, a_t, r_t, s_{t+1})
▷ α is the learning rate

$\mathcal{M} \leftarrow$ initializeMemory(m) ▷ Creating the replay memory with size m
for i **from** 1 **to** $\mathcal{I}$ **do**
    $G \leftarrow$ generateRandomInstance
    $(S, A, T, R) \leftarrow$ initializeEnvironment(G)
    $s_1 \leftarrow$ initializeState(G)
    for t **from** 1 **to** Θ **do**
        $a_t \leftarrow$ softmaxSelection(s_t)
        $r_t = \gamma\, R(s_t, a_t)$
        $s_{t+1} = T(s_t, a_t)$
        $\mathcal{M} \leftarrow$ updateMemory(e_t)
        for j **from** 1 **to** N **do**
            $e \leftarrow$ getSampleFrom$(\mathcal{M})$
            $\mathcal{L}_j(\mathbf{w}) \leftarrow$ squared loss of e
        end for
        $\mathbf{w} \leftarrow \mathbf{w} + \frac{\alpha}{2N} \sum_{j=1}^{N} \mathcal{L}_j(\mathbf{w})$ ▷ Update the weight vector
    end for
end for
return $\mathbf{w}$

4. Experiments

4.1. Dataset

To evaluate the performance and efficiency of the proposed algorithm, we consider three challenging cases suffering from lung cancer. The data is collected from anonymous patient data who underwent radiation therapy using the Cyberknife system at *Centre Hospitalier de l'Université de Montréal* (CHUM). For the lung tumor, a choice of a few predetermined paths developed by the manufacturer is selected regardless of the innate differences among various patients. The first case consists of 28,800 beamlets and 62,41,184 voxels where the tumor is in the right lung. There are 26,208 beamlets and 7,736,670 voxels for the second case for which the tumor lies in the left lung. The third plan has 3,440,112 voxels and 29,952 beamlets, and the tumor lies in the right lung.

To this end, we evaluated the results of the proposed method with the current clinical treatment plan for the lung tumor including 100, 91, and 104 non-coplanar 800

source-to-axis distance (SAD) nodes configured by the manufacturer scattered around the centroid of the target volume for patient 1, 2, and 3 respectively.

4.2. Setup

Our model is implemented in Python 3.7. Training is carried out on one GPU (NVIDIA V100 Volta, 32GB memory) for 24 hours on randomly generated instances and Adam optimizer is used for training. To create an instance in the training phase, at any node, values for different features are selected from the augmented database. Therefore, we can compute the rewards namely distance-driven, dose-driven, and beam-spread measure. It should be pointed out that all of these randomly generated values are normalized to have similar ranges for all of the features during the learning.

Before initiating the training phase, we generate a set of 100 held-out validation instances to track the performance of the learned model and the baselines as the training goes on. The model resulting in the best average reward on the held-out validation set is then selected as the final one and tested on another set of randomly generated graphs with the same configuration as the training set to evaluate the generalization ability of the model.

Three baselines are developed to compare the performance of our proposed algorithm. They select a subset of beams and the best trajectory in a single pass. Firstly, for each instance in the validation set, we randomly create 50 trajectories and output the the best, the worst, and the average objective values. In addition, a greedy heuristic is developed as follows. Starting from the resting point of the robotic arm, at each time step, we select n nearest points as the candidates to be selected for the next node to visit. For each potential next node and the partial trajectory created so far, we compute the dose-driven and the beam-spread score and add the node yielding the maximum total reward to the tour. We also implemented and solved the problem (with the same instances used in the RL and the heuristic) in Gurobi 9.0.0 [35] with time limits. The mathematical programming model is represented in the appendices. Restricting the time to converge prevents finding the optimal solution using the exact solver and results in some negligible optimality gap. Nonetheless, for practical cases, the optimizer takes a long time to converge, which limits clinical applicability.

Once the subset of beams is selected, we follow the algorithm represented by Renaud et al. [19] for only photon particles. This algorithm is implemented in C++ and the quadratic mathematical programming is solved by IPOPT 3.13‡ which is based on interior-point methods. This outputs the weight of the every selected beam and how to deliver the treatment.

‡ `https://coin-or.github.io/Ipopt/`

4.3. Training

We initialize the parameters of the policy network (weight vector $\mathbf{w}$) according to *Xavier* initialization [36] sampled uniformly from $(-x, x)$ where $x = g\left[6 \,/\, (dim_{in} + dim_{out})\right]^{1/2}$ where dim_{in} and dim_{out} refer to the input and output dimensions of the current layer and $g = \sqrt{2}$ is the scaling factor used in this work. The capacity of the replay memory is set to 50 and at each epoch, a batch of 32 experiences are sampled from it. We train for 24 hours on the training data generated randomly on-the-fly as described in Section 4.2. The learning rate of $\alpha = 4 \times 10^{-4}$ is set following different experiments.

Using the weight vector $\mathbf{w}$ learned in the training phase, at each time step the next node to be visited (next action) based on the current state s_t is selected by a greedy policy following $a_t = \text{argmax}_{a \in A_t} \hat{Q}(s_t, a, \mathbf{w})$. This process continues until the end of the episode where the predetermined number of beams are selected. With the way that the reward function is defined, the selected beams will not be clustered around some points in the space around the patient.

4.4. Results

In this section, we evaluate the performance of the proposed algorithm based on the quality and efficiency of the treatment on the lung cases. We use the Dose-Volume Histograms (DVH) to evaluate the quality of the plans, a method widely used in practice. These histograms depict the percentage of the organs capturing a certain amount of dose. In order to attain an acceptable treatment plan regarding DVH constraints, we change the structures underdose and overdose weights during the optimization of the dose intensities. We increase the penalty for the structures whose DVH measures are worse than the critical dosimetry references or the proposed values by the physicians. A challenging issue is the voxels that could be identified in more than one structure. In this case, based on the important factor of individual structures, the aforementioned voxels are assigned to the structures with higher priority. All the structures are considered throughout the algorithm, to create individual beam scores and beamlets for the beam intensity optimization. It should be pointed out that fewer multi-structure voxels results in higher quality plans. The figures and tables representing the results for patient 2 and 3 are shown in the appendices.

Figures 3, A1, and A2 compare the DVH diagrams for the cases of $\mathcal{K} = 25$ beams (DQN-Plan) and the clinical plan which uses all the beams (Full-Plan). Healthy tissues are protected in both plans but the dose to the tumor is relatively lower in DQN-Plan. All treatment plans meet clinical needs. As for the healthy tissues for patient 1, the right and left lungs, and the ribs receive lower radiation in high percentage of their volume using the DQN-plan. The heart and trachea attain lower dose in high volumes using the proposed algorithm. In all these cases, the dose deposited in the structures are within the acceptable thresholds.

The heart, right lung, esophagus receive lower dose throughout the structure with the DQN-methods compared to the Full-Plan for patient 2. For the high percentage of

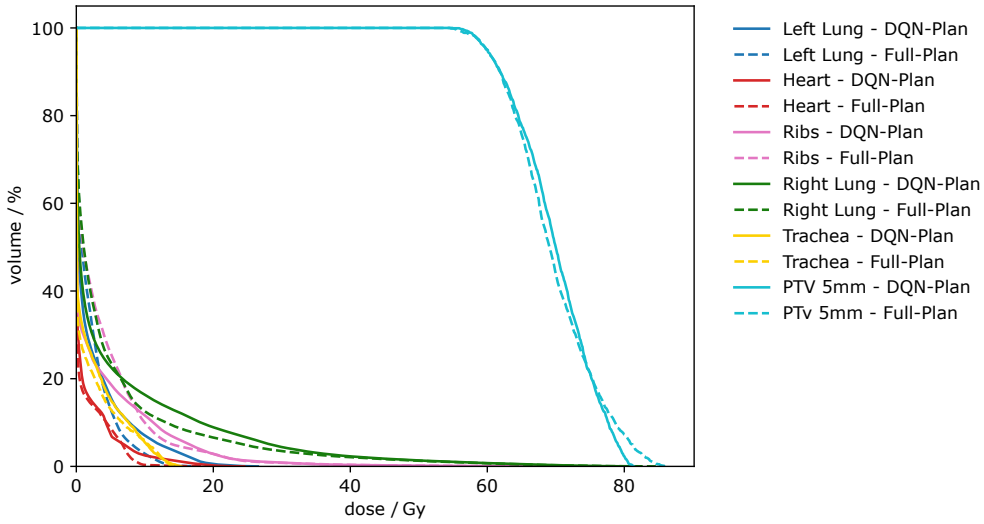

Figure 3. Comparison between DVH diagrams for patient 1.

the volumes of the ribs and the left lung (where the tumor is in), the DQN-Plan gets lower dose deposited. The dose deposited in tumor however is less in the majority of the volume when following the DQN-Plan.

In addition, the robotic arm traverses only 25 beams in the DQN-plan whereas the Full-Plan requires all the beams in the original treatment plan, 100, 93, and 104 for patients 1, 2, and 3. In general, the DQN-Plan maintains the quality of the Full-Plan treatment within the clinically accepted thresholds, while delivered in shorter time. Table 1 compares the total reward (objective), execution time, and the total distance traversed by the robotic arm, and time for different methods. It should be pointed out that the value of the objective function and the execution time are irrelevant for the clinical method which is currently used at CHUM. The arm travels the distance while the beam is off. The majority of the treatment time is spent on moving from one beam to the next one. The total reward (the objective) is comprised of three parts, total distance traversed, total dose-driven score collected, and the level of the sparsity of the selected beams around the patient. Therefore, merely having the lowest distance covered will not necessarily lead to a more advantageous treatment.

As illustrated, although DQN-Plan does not yield the minimum distance traversed in comparison with the heuristic and random selection, it attains a better total reward and hence a higher treatment quality shown in Tables 2, A1, and A2. In these tables, $\mathrm{D}_{X\%}$ represents the amount of dose deposited in at least $X\%$ of the structure, D_{max} is the maximum dose delivered to the structure, and D_{mean} is the mean dose to the structure. V_Y Gy percentage of the structure volume receiving Y Gy dose.

Table 1 also demonstrates the treatment time comparison between the clinical plan and the DQN-Plan. Using the DQN-Plan results in a 35%, 33%, and 40% reduction of the treatment time for patients 1, 2, and 3, respectively.

We represent the performance of the proposed Deep Q-learning algorithm during

Table 1. Solution comparisons between different methods of solving the BOO.

	Method	Obj.	Execution time (s)	Total arm Distance	Time (min)
Patient 1	DQN	3.53	1.09	5,529	35
	Gurobi	3.27	3600.00	6,350	37
	Heuristic	4.50	2.47	5,494	35
	Random	4.65	0.15	2,696	31
	Clinical	NA[a]	NA[a]	17,726	54
Patient 2	DQN	4.18	1.08	4,656	35
	Gurobi	3.18	3600.00	4,836	36
	Heuristic	4.26	2.89	4,128	34
	Random	4.88	0.24	6,730	39
	Clinical	NA[a]	NA[a]	12,936	51
Patient 3	DQN	1.80	1.38	4,068	30
	Gurobi	1.53	3600.00	5,851	31
	Heuristic	4.26	1.26	3,696	29
	Random	2.41	0.09	8,166	35
	Clinical	NA[a]	NA[a]	22,553	50

[a]Objective function and execution time are not defined for the Clinical-path method as this is the approach that is used at the clinic and uses all of the possible nodes.

the training phase in Figure 4 for both patients. At certain points in time (every 100 episodes), we apply the learned model against the instances in the validation set and show it in the plot. The same results are also illustrated for each baseline (random selection, heuristic, and the results of the implementation in Gurobi). Figure 5 illustrates a trajectory generated by the DQN-method compared with one of the Clinical method.

4.5. Discussion

In this work, we proposed a method based on Deep Q-learning to solve the BOO problem for the Cyberknife system. It entails a reduction in treatment time while maintaining plan quality. The selected subset of the beams can be forwarded to any formulation of the Fluence Map Optimization or Direct Aperture Optimization problem (with different objective functions and constraints) to realize beam intensities and leaf sequencing.

Compared to previous methods, using a deep Q-learning approach enables us to include dose considerations related to each patient as well as the geometric characteristics of beams. The proposed algorithm exploits the benefits of both approaches of solving the BOO problem. Firstly, capturing the dose-related scores results in more realistic treatment planning. Methods assigning scores to beams by geometric characteristics generally suffer from the lack of dose-relevant rewards as they do not directly capture the information of the deposited dose of individual beams. However, we showed that including the dose-driven part in the rewards compensates this issue. Furthermore, similar to heuristics, using the learned models on new patients

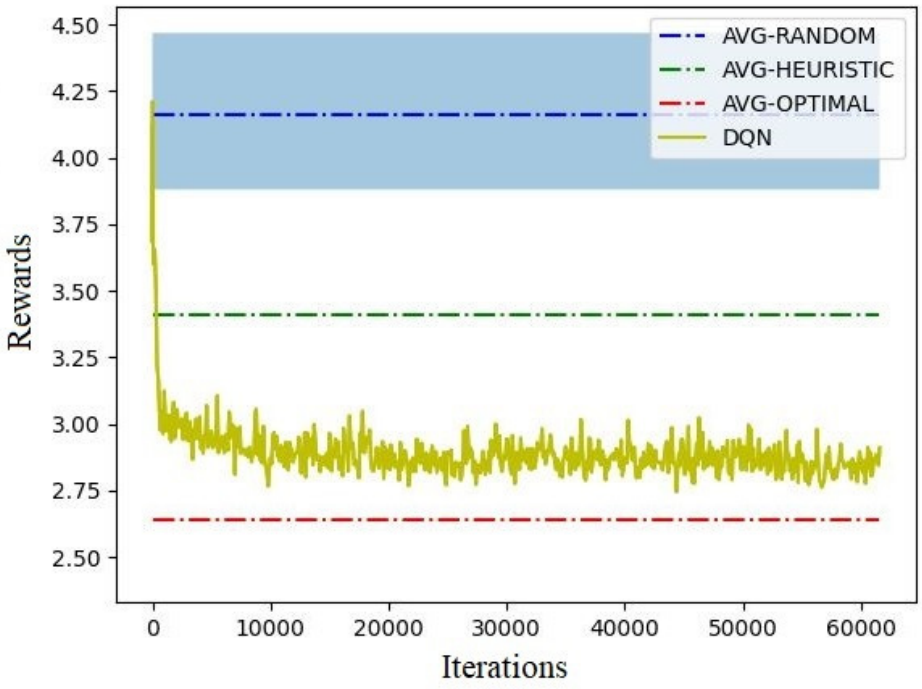

Figure 4. Total reward of the algorithm against the fixed validation set during training. The shaded area depicts the spectrum between the worst and the best random trajectories.

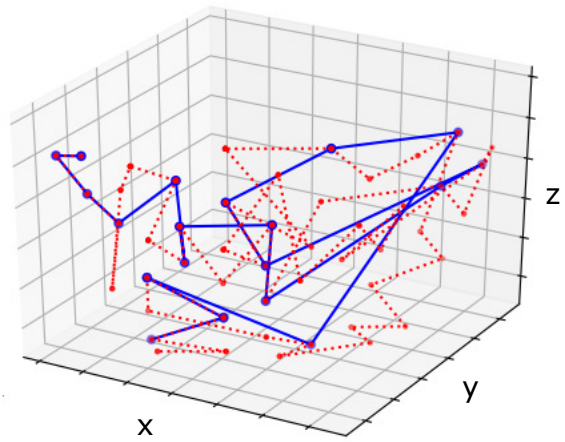

Figure 5. Comparison of the trajectories of the DQN-method (blue) and the Clinical-method (red).

takes a short amount of time. However, the majority of the heuristics in the literature use only anatomical features of the patients.

As mentioned before, long treatment times are a fundamental drawback of the Cyberknife system treatment planning. We reduced the overall treatment time by considering travel times in the distance-driven part in reward computations of the proposed algorithm.

Finally, imposing spatial dispersion as the terminal reward of the episodes helps to choose the most spread subset of beams while minimizing the total rewards gained. As mentioned before, BOO can be modeled as a combinatorial optimization problem which is proven to be NP-hard. Adding these measures to this problem will increase the complexity of this problem. However, deep Q-learning method allows to consider multiple measures as different (immediate and terminal) rewards. Therefore, deep Q-learning is an efficient and effective method to tackle the BOO problem. This approach results in treatment plans with a substantially shorter treatment time compared to current clinical practice using all the beams. Although it takes a relatively long time for the model to be trained (as is the case for all learning tasks), once the model is trained, finding the solution to new instances in the inference part takes up to few seconds compared to the hours of computational time by solvers such as Gurobi. For training, we only generate random values for node coordinates and doses deposited to different organs through the beams corresponding at each node. Therefore, we do not restrain the learning for a particular case.

Another challenge with the current clinical methods is the use of a set of fixed trajectories devised by the manufacturer for all patients with the same tumors. Neglecting individual-specific variations is likely to diminish the quality of the plan, where even small changes can greatly influence the health of the patients.

The treatment plan resulting from the beams generated by the DQN-plan has acceptable dose quality. For the first patient, the maximum dose, average dose, and median dose deposited in the target volume is 4.5 Gy lower, 0.2 Gy larger and 0.8 Gy larger respectively compared to the Full-Plan. The maximum dose is reduced by 4.5 Gy in DQN-plan compared to the full-plan. While the mean dose and the median are increased with negligible values of 0.2 Gy and 0.8 Gy respectively. As for the right lung where the tumor is located, the maximum dose and the median dose are reduced by 3.3 Gy and 0.6 Gy while the average dose is 0.3 Gy higher in DQN-plan.

For the second patient, maximum dose delivered to tumor is 5.34 Gy less in the DQN-Plan. The right lung, heart and esophagus are also much better protected, considering the maximum and mean dose in DQN-Plan. The maximum dose deposited in the right lung is 12.56 Gy lower than the Full plan, while the mean dose is if reduced by around 80%. maximum dose attained by the heart and esophagus are down by 4.33 Gy and 4.59 Gy respectively. As for the left lung, where the tumor is located at, a small reduction of 5.32 Gy is obtained. On the other hand, although the maximum dose delivered to the ribs is almost 1.5 times greater using the DQN-Plan, the average dose is on par with the Full-Plan.

For the last patient, the maximum dose is about 9 Gy less in the tumor in the DQN-Plan. The maximum dose deposited in the left lung is also 20 Gy lower while the max and mean dose in the other organs are similar in both plans. Therefore, the DQN-Plan at maintains the dose-volume parameters of the clinical full-Plan. The dose-wash images for different patients comparing DQN method and the clinical full-plan is illustrated in Figure A3.

Table 2. Dose-volume parameters for the target volume and critical structures for treatment plans generated by different method for patient 1.

Plan	Structure	Statistics		
DQN-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.0	58.1	81.3
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	81.1	5.1	8.9
	Left Lung	26.4	2.2	0.5
	Ribs	73.9	**2.9**	2.8
	Hearts	24.4	1.1	0.2
Gurobi-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.0	58.2	**81.2**
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	**80.5**	5.2	7.8
	Left Lung	26.6	2.3	0.4
	Ribs	75.8	3.2	**2.2**
	Hearts	20.5	1.2	0.0
Heuristic-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.0	57.7	85.7
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	85.5	5.0	7.4
	Left Lung	33.0	2.3	1.7
	Ribs	79.3	3.1	2.8
	Hearts	**19.3**	**0.5**	0.0
Random-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.0	56.3	97.5
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	94.7	5.2	9.9
	Left Lung	58.0	2.2	4.3
	Ribs	79.1	3.4	6.5
	Hearts	62.7	0.8	1.5
Full-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.0	**56.3**	85.8
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	84.5	**4.8**	**6.5**
	Left Lung	**17.3**	**2.0**	**0.0**
	Ribs	**72.4**	3.6	2.8
	Hearts	20.5	0.9	**0.0**

The values corroborate the fact that intelligently selecting a subset of nodes results in relatively the same treatment quality. It should be remarked that these similar

treatments are obtained by reducing the robotic arm movement time. This reduction of treatment time yields higher comfort for the patients, eliminates the burden on the clinics and allows more patients to be treated in any given time duration.

5. Conclusion

In this work, we have proposed a Deep Q-learning algorithm for the Beam Orientation Optimization problem for the Cyberknife system treatment planning. This algorithm generates a set of favorable beams which is tailored to consider patient-specific dosimetric features and beam-related geometric features. This approach also tries to maximally distribute the selected beams around the patient. The proposed Deep Q-learning algorithm has the following advantages over the other methods that can be found in the literature:

- The majority of the previous methods consider only a single feature (geometric or dosimetric) for the individual beam scores. Using this method, we can integrate any number of features into the neural network structure to generate a favorable trajectory.
- By intelligently selecting the beams, we attained treatment plans with much shorter times while maintaining the treatment quality of using all the possible beams.

Finally, we have evaluated our proposed algorithm on three challenging lung cancer cases to demonstrate the effectiveness and efficiency of the algorithm. While the training may take a long time, it should be remarked that the solution time is really short.

Although we have considered identical importance for different features in generating rewards at each step of the Deep Q-learning algorithm, one might tune the weights to achieve an even better treatment quality. For instance, having higher weight on the beam-spread score would lead to an even more scattered beam formation. In addition, as the training is not restricted to a particular patient or cancer type, it would be beneficial to evaluate the possibility of applying the same model to patients suffering from the different types of cancers.

Acknowledgments

The Authors would like to thank Dr. Stephane Bedwani and the Department of radiation oncology, Centre hospitalier de l'Universite de Montreal (CHUM), who helped us by preparing and providing the case data and for their illuminating insights. This work has been approved by CHUM research ethics committee with the approval number CER 20.244.

Appendix

Mathematical Programming Model

The Mathematical Programming model which is implemented in Gurobi is represented as follows in (A.1)-(A.7). Let $G = (V, E)$ be a graph where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of n vertices and E is a set of edges. V corresponds to the nodes at which the robotic arm delivered beam towards the patient. An edge $e \in E$ represents the path the robotic arm traverses from one node to the other. Let a score f_i be associated to each node $v_i \in V$ and a distance d_e associated to each edge. A beam spread measure is denoted by $s_{ij} = K\,(1 - \cos\alpha_{ij})^{-1}$ for every pair of the nodes (beams) i and j. α_{ij} is the angle separation between the pair of beams [12]. Distance d_e, score f_i and spread measure s_{ij} corresponds to r_{dist}, r_{dose}, and r_{spread} described in Section 3.5.

We associate a binary variable x_e to every edge $e \in E$, equal to 1 if the edge is traversed and 0 otherwise. Another binary variable y_i is associated with every $v_i \in V$, equal to 1 if and only if the corresponding node is used in the solution.

$$\min \quad \sum_{e \in E} d_e\, x_e \; + \sum_{v_i \in V} f_i\, y_i + \sum_{v_i \in V} \sum_{v_j \in V} s_{ij}\, y_i\, y_j \tag{A.1}$$

$$\text{s.t.} \sum_{e\,\in\,\delta(\{v_i\})} x_i = 2\, y_i \qquad v_i \in V \tag{A.2}$$

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad S \subseteq V, 2 \leq |S| \leq |V| - 2 \tag{A.3}$$

$$\sum_{v_i \in V} y_i = \mathcal{N} \tag{A.4}$$

$$y_1 = 1 \tag{A.5}$$

$$x_e \in \{0, 1\} \qquad e \in E \tag{A.6}$$

$$y_i \in \{0, 1\} \qquad v_i \in V. \tag{A.7}$$

Constraints (A.2) are the degree constraints. Constraints (A.3) eliminates subtours form the solutions. As discussed before, the number of selected nodes is illustrated by $\mathcal{N} = 25$ and enforced in constraint (A.4). The resting point of the robotic arm is denoted by node 1 as is imposed by constraint (A.5).

Tables and Figures

In this section, we present the DVH diagrams and the tables discussing the dose-volume parameters of different methods for patients 2 and 3.

Table A1. Dose-volume parameters for the target volume and critical structures for treatment plans generated by different method for patient 2.

Plan	Structure	Statistics		
DQN-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.00	58.60	**80.98**
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	**1.59**	**0.01**	0.00
	Left Lung	**80.98**	5.87	9.26
	Ribs	61.83	5.94	11.45
	Hearts	**0.06**	**0.00**	0.00
	Esophagus	**1.88**	0.07	0.00
Gurobi-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.00	58.50	84.00
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	6.94	0.76	0.00
	Left Lung	84.00	5.96	8.69
	Ribs	56.32	5.48	**0.00**
	Hearts	6.35	0.22	0.00
	Esophagus	5.26	0.88	0.00
Heuristic-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.00	58.09	84.35
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	7.72	0.65	0.00
	Left Lung	84.35	5.85	8.70
	Ribs	57.71	**5.53**	10.75
	Hearts	3.39	6.43	0.00
	Esophagus	9.43	1.27	0.00
Random-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.00	58.80	87.60
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	3.65	0.07	0.00
	Left Lung	87.60	6.45	9.47
	Ribs	58.27	5.62	10.46
	Hearts	4.04	0.02	0.00
	Esophagus	4.55	**0.62**	0.00
Full-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	60.00	**58.1**	86.32
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	14.15	0.54	**0.0**
	Left Lung	86.32	**5.46**	**7.52**
	Ribs	**38.92**	5.90	3.86
	Hearts	4.39	0.09	**0.00**
	Esophagus	6.47	1.05	**0.00**

Table A2. Dose-volume parameters for the target volume and critical structures for treatment plans generated by different method for patient 3.

Plan	Structure	Statistics		
DQN-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	50.0	**43.0**	90.2
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	90.2	13.05	**20.1**
	Left Lung	**11.1**	**0.04**	**0.0**
	Ribs	86.9	6.2	14.7
	Hearts	57.4	3.1	4.9
Gurobi-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	50.0	45.5	**82.8**
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	**82.1**	12.1	23.7
	Left Lung	22.4	2.7	0.6
	Ribs	76.6	6.9	14.9
	Hearts	**44.4**	3.1	**4.7**
Heuristic-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	50.0	44.3	84.6
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	84.7	12.3	22.4
	Left Lung	34.6	2.6	5.9
	Ribs	**76.5**	**5.3**	9.9
	Hearts	54.4	4.8	10.8
Random-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	50.0	43.6	86.0
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	86.1	13.7	24.1
	Left Lung	43.2	3.3	7.5
	Ribs	82.2	6.2	11.2
	Hearts	69.4	7.7	15.7
Full-Plan		**Dose 95% (Gy)**	**Dose 98% (Gy)**	**D max (Gy)**
	Target	50.0	43.7	99.1
		D max (Gy)	**D mean (Gy)**	**V 20Gy (%)**
	Right Lung	99.1	**9.5**	**16.9**
	Left Lung	31.6	0.8	0.3
	Ribs	87.9	6.9	**11.0**
	Hearts	55.2	3.3	4.88

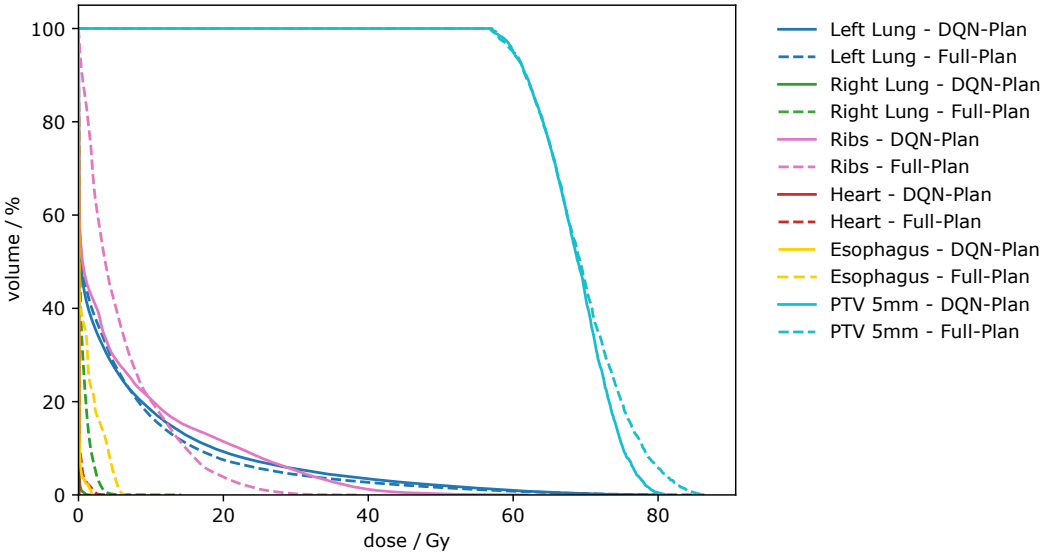

Figure A1. Comparison between DVH diagrams for patient 2.

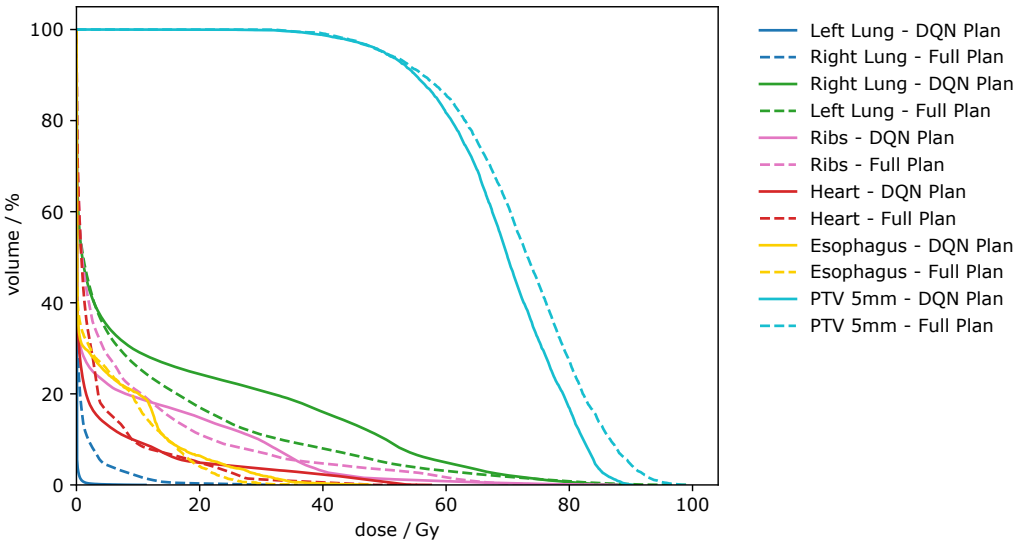

Figure A2. Comparison between DVH diagrams for patient 3.

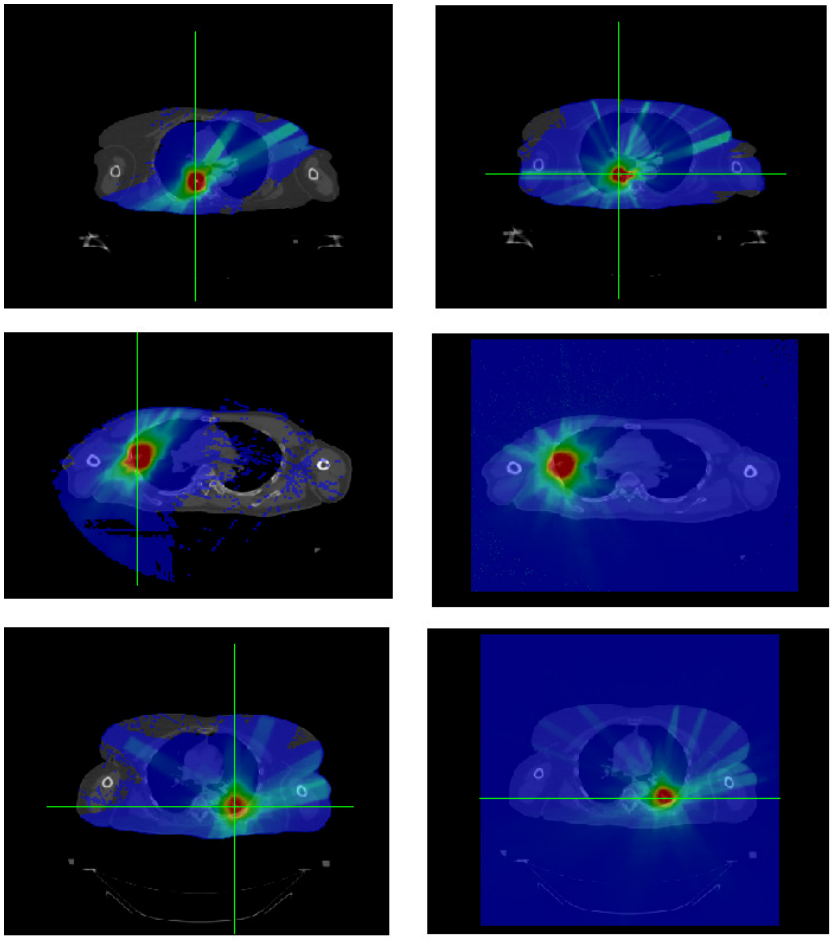

Figure A3. Dose-wash images. Left column represents the DQN-plan and right column is the Clinical full-plan. Rows 1, 2, and 3 correspond to patients 1, 2, and 3, respectively.

References

[1] Vasant Kearney, Joey P. Cheung, Christopher McGuinness, and Timothy D. Solberg. CyberArc: A non-coplanar-arc optimization algorithm for CyberKnife. *Physics in Medicine and Biology*, 2017. ISSN 13616560. doi: 10.1088/1361-6560/aa6f92.

[2] Laura Masi, Margherita Zani, Raffaela Doro, Silvia Calusi, Vanessa Di Cataldo, Ivano Bonucci, Samantha Cipressi, Giulio Francolini, Pierluigi Bonomo, and Lorenzo Livi. CyberKnife MLC-based treatment planning for abdominal and pelvic SBRT: Analysis of multiple dosimetric parameters, overall scoring index and clinical scoring. *Physica Medica*, 2018. ISSN 1724191X. doi: 10.1016/j.ejmp.2018.11.012.

[3] James L. Bedford, Henry S. Tsang, Simeon Nill, and Uwe Oelfke. Treatment planning optimization with beam motion modeling for dynamic arc delivery of SBRT using Cyberknife with multileaf collimation. *Medical Physics*, 2019. ISSN 0094-2405. doi: 10.1002/mp.13848.

[4] H. Edwin Romeijn, Ravindra K. Ahuja, James F. Dempsey, and Arvind Kumar. A column generation approach to radiation therapy treatment planning using aperture modulation. *SIAM Journal on Optimization*, 2005. ISSN 10526234. doi: 10.1137/040606612.

[5] Humberto Rocha, Joana Matos Dias, Tiago Ventura, Brígida da Costa Ferreira, and Maria do Carmo Lopes. Beam angle optimization in IMRT: are we really optimizing what matters? *International Transactions in Operational Research*, 2019. ISSN 14753995. doi: 10.1111/itor.12587.

[6] Mark Bangert and Jan Unkelbach. Accelerated iterative beam angle selection in IMRT. *Medical Physics*, 2016. ISSN 00942405. doi: 10.1118/1.4940350.

[7] A. B. Pugachev, A. L. Boyer, and L. Xing. Beam orientation optimization in intensity-modulated radiation treatment planning. *Medical Physics*, 2000. ISSN 00942405. doi: 10.1118/1.599001.

[8] Ruijie Yang, Jianrong Dai, Yong Yang, and Yimin Hu. Beam orientation optimization for intensity-modulated radiation therapy using mixed integer programming. In *IFMBE Proceedings*, 2007.

[9] Sifeng Lin, Gino J. Lim, and Jonathan F. Bard. Benders decomposition and an IP-based heuristic for selecting IMRT treatment beam angles. *European Journal of Operational Research*, 251(3):715–726, 6 2016. ISSN 03772217. doi: 10.1016/j.ejor.2015.12.050.

[10] Joel Mullins, Marc-André Renaud, Monica Serban, and Jan Seuntjens. Simultaneous trajectory generation and volumetric modulated arc therapy optimization. *Medical Physics*, 47(7):3078–3090, 7 2020. ISSN 0094-2405. doi: 10.1002/mp.14155. URL `https://doi.org/10.1002/mp.14155`.

[11] Andrei Pugachev and Lei Xing. Pseudo beam's-eye-view as applied to beam orientation selection in intensity-modulated radiation therapy. *International Journal of Radiation Oncology Biology Physics*, 2001. ISSN 03603016. doi: 10.1016/S0360-3016(01)01736-9.

[12] Lulin Yuan, Wei Zhu, Yaorong Ge, Yuliang Jiang, Yang Sheng, Fang Fang Yin, and Q. Jackie Wu. Lung IMRT planning with automatic determination of beam angle configurations. *Physics in Medicine and Biology*, 2018. ISSN 13616560. doi: 10.1088/1361-6560/aac8b4.

[13] Mark Bangert and Uwe Oelfke. Spherical cluster analysis for beam angle optimization in intensity-modulated radiation therapy treatment planning. *Physics in Medicine and Biology*, 2010. ISSN 00319155. doi: 10.1088/0031-9155/55/19/025.

[14] R. Lee MacDonald and Christopher G. Thomas. Dynamic trajectory-based couch motion for improvement of radiation therapy trajectories in cranial SRT. *Medical Physics*, 2015. ISSN 00942405. doi: 10.1118/1.4917165.

[15] Gregory Smyth, Philip M. Evans, Jeffrey C. Bamber, Henry C. Mandeville, Liam C. Welsh, Frank H. Saran, and James L. Bedford. Non-coplanar trajectories to improve organ at risk sparing in volumetric modulated arc therapy for primary brain tumors. *Radiotherapy and Oncology*, 2016. ISSN 18790887. doi: 10.1016/j.radonc.2016.07.014.

[16] Peng Dong, Percy Lee, Dan Ruan, Troy Long, Edwin Romeijn, Yingli Yang, Daniel Low, Patrick Kupelian, and Ke Sheng. 4π non-coplanar liver SBRT: A novel delivery technique. *International Journal of Radiation Oncology Biology Physics*, 2013. ISSN 03603016. doi: 10.1016/j.ijrobp.2012.09.028.

[17] Marco Langhans, Jan Unkelbach, Thomas Bortfeld, and David Craft. Optimizing highly noncoplanar VMAT trajectories: The NoVo method. *Physics in Medicine and Biology*, 2018. ISSN 13616560. doi: 10.1088/1361-6560/aaa36d.

[18] Qihui Lyu, Victoria Y. Yu, Dan Ruan, Ryan Neph, Daniel O'Connor, and Ke Sheng. A novel optimization framework for VMAT with dynamic gantry couch rotation. *Physics in Medicine and Biology*, 2018. ISSN 13616560. doi: 10.1088/1361-6560/aac704.

[19] Marc André Renaud, Monica Serban, and Jan Seuntjens. On mixed electron-photon radiation therapy optimization using the column generation approach. *Medical Physics*, 2017. ISSN 00942405. doi: 10.1002/mp.12338.

[20] James L. Bedford, Peter Ziegenhein, Simeon Nill, and Uwe Oelfke. Beam selection for stereotactic ablative radiotherapy using Cyberknife with multileaf collimation. *Medical Engineering and Physics*, 2019. ISSN 18734030. doi: 10.1016/j.medengphy.2018.12.011.

[21] James L. Bedford, Simeon Nill, and Uwe Oelfke. Dosimetric accuracy of delivering SBRT using dynamic arcs on Cyberknife. *Medical Physics*, 47(4):1533–1544, 2020. ISSN 00942405. doi: 10.1002/mp.14090. URL `https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.14090`.

[22] R.S. Sutton and A.G. Barto. Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks*, 1998. ISSN 1045-9227. doi: 10.1109/tnn.1998.712192.

[23] Richard Bellman. Dynamic programming and Lagrange multipliers. *Proceedings of the National Academy of Sciences of the United States of America*, 42(10):767, 1956.

[24] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 1992. ISSN 0885-6125. doi: 10.1007/bf00992698.

[25] Martin Riedmiller. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In João Gama, Rui Camacho, Pavel B Brazdil, Alípio Mário Jorge, and Luís Torgo, editors, *Machine Learning: ECML 2005*, pages 317–328, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31692-3.

[26] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[27] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized Experience Replay. 11 2015. URL `https://arxiv.org/abs/1511.05952`.

[28] Petar Veličković, Arantxa Casanova, Pietro Liò, Guillem Cucurull, Adriana Romero, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, 2018.

[29] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. 2 2021. URL `http://arxiv.org/abs/2102.09544`.

[30] Hanjun Dai, Elias B. Khalil, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, 2017.

[31] Quentin Cappart, Emmanuel Goutierre, David Bergman, and Louis-Martin Rousseau. Improving Optimization Bounds Using Machine Learning: Decision Diagrams Meet Deep Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019. ISSN 2159-5399. doi: 10.1609/aaai.v33i01.33011443.

[32] Chieh Hsiu Jason Lee, Dionne M. Aleman, and Michael B. Sharpe. A fast beam orientation optimization method that enforces geometric constraints in IMRT for total marrow irradiation. *International Transactions in Operational Research*, 2015. ISSN 14753995. doi: 10.1111/itor.12093.

[33] Ignacio Martín, Sebastian Troia, José Alberto Hernández, Alberto Rodríguez, Francesco Musumeci, Guido Maier, Rodolfo Alvizu, and Óscar de Dios. Machine Learning-Based Routing and Wavelength Assignment in Software-Defined Optical Networks. *IEEE Transactions on Network and Service Management*, 16(3):871–883, 2019. doi: 10.1109/TNSM.2019.2927867.

[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. 10 2017. URL `https://arxiv.org/abs/1710.10903`.

[35] Gurobi Optimization LLC. Gurobi Optimizer reference manual. *Www.Gurobi.Com*, 2019.

[36] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.