UCLouvain

École polytechnique de Louvain

Robust Reinforcement Learning Optimization of Radiotherapy Dose Fractioning

Author: **Florian Martin**
Supervisor: **Benoît Macq**
Readers: **Manon Dausort, Damien Dasnoy-Sumell, Christophe De Vleeschouwer**
Academic year 2022–2023
Master [120] in Electrical Engineering

Acknowledgements

First, I wish to extend my deepest gratitude to my supervisor, Professor Benoit Macq. His guidance, patience, and astute insights have been invaluable throughout the process of crafting this thesis.

I would also like to express my sincere appreciation to Damien Dasnoy-Sumell, Manon Dausort, and Mélanie Ghislain. Their involvement in my project, the time they spent discussing and debating the nuances of the work, and the constructive criticism they offered have been instrumental in refining my thoughts and polishing this work to its current form. They have taught me the value of collaboration and the strength that lies in diverse perspectives. Their commitment to my work has made a profound impact on my research and personal growth.

I would also like to express my deep gratitude to Christophe De Vleeschouwer for graciously taking the time to read and review this work.

In addition, I wish to acknowledge the academic and non-academic staff of the university, whose tireless efforts often go unnoticed but are crucial for facilitating my studies and research. Your commitment to creating and maintaining a conducive learning environment has not gone unnoticed.

Finally, I wish to acknowledge the unwavering support and encouragement of my family. Their belief in my potential and their continuous moral support have been a pillar of strength throughout my academic journey. I am fortunate and grateful for their love, sacrifice, and unwavering belief in my abilities.

This thesis would not have been possible without the collaboration, support, and encouragement of each and every one of you. My deepest appreciation and heartfelt thanks to all of you.

ChatGPT has been involved to assist in the writing process.

Abstract

Radiation therapy is an integral treatment option for cancer, its success hinging on the precise dosage delivery to the target and the preservation of surrounding healthy tissues. This master thesis examines the robustness of Reinforcement Learning (RL) optimization, a potential tool for automating radiation therapy dose fractionation, in response to varying parameters within cellular models.

Specifically, we use the combined cellular model proposed by A. Jalalimanesh [1] and O'Neil [2], and manipulate key parameters such as nutrients consumption rate of cancer cells, radiation sensitivity, and cell cycle duration to observe their impact on optimization results. Employing RL algorithms like Q-learning, Sarsa, and Expected Sarsa, we optimize the radiotherapy dose fractionation schedule and compare their effectiveness under different parameter conditions.

The goal of this study is to generate superior treatment plans concerning Tumor Control Probability (TCP), number of radiation fractions, total radiation dose, treatment duration, and healthy cell survival rate. We extensively test our RL agents' ability to navigate a wide range of scenarios. In benchmark tests, our agents outperformed the baseline treatment by an average of 51.1% across these metrics.

Moreover, we incorporate observations of the patient during treatment for a more realistic approach. We've developed an ML-based adaptative decision-making framework that autonomously modifies treatment plans based on individual tumor images. Leveraging machine learning algorithms, we automated the decision-making process for treatment adaptation. This framework, in the benchmark tests, showed a 54.9% average improvement compared to the baseline treatment plan. Remarkably, this automated system demonstrated its prowess in crafting adapted treatment plans even in extraordinarily aggressive environments where standalone reinforcement learning algorithms failed to find any solutions. These advancements present opportunities for more personalized and effective cancer treatments, reducing patient morbidity and enhancing overall treatment efficacy.

Contents

1 Introduction **6**

2 Literature Review: Reinforcement Learning **7**
2.1 Reinforcement Learning Framework . . . 7
2.1.1 Multi-armed bandits . . . 7
2.1.2 Incremental Implementation . . . 8
2.1.3 Markov Decision Process . . . 9
2.2 Reinforcement learning algorithms . . . 13
2.2.1 Dynamic Programming methods . . . 13
2.2.2 Monte Carlo methods . . . 14
2.2.3 Temporal-Difference Learning . . . 14
2.2.4 Comparison DP, MC, and TD methods . . . 17
2.2.5 Deep Reinforcement Learning . . . 17

3 Literature Review: Radiotherapy **20**
3.1 Overview & Basic Principles . . . 20
3.1.1 Cancer & Healthy Cells . . . 20
3.1.2 Radiotherapy treatment . . . 20
3.1.3 RT treatment example . . . 21
3.1.4 Working principle of RT . . . 21
3.2 Cell-based Computational Model . . . 21
3.2.1 Principle . . . 21
3.2.2 Radiation Response . . . 22
3.3 Existing cellular models . . . 22

4 Materials **23**
4.1 Cellular Model for simulations . . . 23
4.1.1 Cell Cycle . . . 24
4.1.2 Cell survival curve . . . 24
4.1.3 Nutrients thresholds . . . 25
4.1.4 TCP and NTCP . . . 27
4.1.5 Radiation . . . 28
4.2 Simulation Behavior . . . 28
4.2.1 Observations from environment . . . 29
4.2.2 Termination conditions . . . 29
4.2.3 Baseline Simulation . . . 32
4.3 Reinforcement learning algorithms . . . 32
4.3.1 Reward function . . . 33
4.3.2 Observation space quantization . . . 34

5 Methods **37**
5.1 Agent Evaluation . . . 37
5.1.1 Agent's learning behavior . . . 37
5.1.2 Conclusion . . . 40
5.2 Nutrients Consumption Changes . . . 41

5.3 Cell Cycle Duration Changes . . . 42
5.4 Radiosensitivities Changes . . . 43

6 Results **44**
6.1 RL Agent Selection . . . 44
6.2 Nutrients Consumption Changes . . . 45
6.3 Cell Cycle Duration Changes . . . 49
6.3.1 Cell cycle robustness through re-training . . . 52
6.3.2 Comparison with agents trained on environments with various cell cycle duration . . . 52
6.3.3 Conclusion . . . 53
6.4 Radiosensitivities Changes . . . 54
6.4.1 Radiosensitivity robustness through re-traininng . . . 56
6.4.2 Comparison with agents trained on environments with various radiosensitivity . . . 57
6.4.3 Conclusion . . . 57
6.5 Mean value of first and second radiation dose . . . 58

7 ML-based Adaptive Decision-Making: Utilizing Tumor Observations for Robust Radiotherapy Treatments **59**
7.1 Simulation tendencies . . . 59
7.2 Cancer Cells' Evolution Observations . . . 60
7.2.1 TCP classifier and TCP regressor . . . 61
7.3 Convolutional Neural Network Classifier . . . 66
7.3.1 Classifier comparison . . . 68
7.3.2 Conclusion . . . 69
7.4 Agent robustness classes . . . 70
7.4.1 Conclusion . . . 72
7.5 Automatic Robustness Decision Making . . . 74
7.5.1 Methodology . . . 74
7.5.2 Results . . . 75
7.5.3 Summary and discussion . . . 83
7.5.4 Ways of improvements . . . 84
7.5.5 Conclusion . . . 86
7.6 Further Works . . . 87

8 Conclusion **90**

9 Annex **91**
9.1 Reinforcement learning algorithms . . . 91
9.1.1 Dynamic Programming . . . 91
9.1.2 Monte Carlo Methods . . . 93
9.1.3 Policy Improvement Theorem . . . 96
9.2 Early Stopping Possibilities . . . 97
9.3 Results of the automatic robustness decision-making . . . 99
9.3.1 Treatment shapes . . . 100
9.3.2 Comparison with the baseline treatment . . . 101
9.4 Graphical User Interface . . . 103
9.4.1 Treatment page . . . 104
9.4.2 Nutrients page . . . 105
9.4.3 Cell cycle page . . . 106
9.4.4 Radiosensitivity page . . . 107
9.4.5 Classifier page . . . 108
9.4.6 Simulation page . . . 108

Acronyms

ARA Auto-Robust Agent. 76

ATP Adenosine Triphosphate. 23

AUC Area Under the Curve. 61

CNN Convolutional Neural Network. 32

CT Computed tomography. 20

DNA Deoxyribonucleic acid. 20

DP Dynamic Programming. 13

DRL Deep Reinforcement Learning. 17

GUI Graphical User Interface. 89

LQ Linear-Quadratic. 22

MC Monte Carlo. 93

MDP Markov Decision Process. 9

MRI Magnetic Resonance Imaging. 20

NTCP Normal Tissue Complication Probability. 22

RF Random Forest. 61

RFR Random Forest Regressor. 62

RL Reinforcement Learning. 2

ROC Receiver Operating Characteristic. 61

RT Radiotherapy. 20

TD Temporal-Difference. 13

Chapter 1

Introduction

Since Wilhelm Conrad Roentgen's discovery of radioactive X-rays in 1895, radiotherapy has been a primary tool in cancer treatment. The field has evolved significantly over the years, improving patient outcomes and cure rates. A critical challenge in radiotherapy is the optimization of treatment plans—determining the best radiation dose and schedule to control tumor growth while preserving surrounding healthy tissues.

Reinforcement Learning (RL), a type of machine learning where an agent learns actions to maximize cumulative rewards, has emerged as a potential solution to this problem. In domains like gaming, robotics, and finance, RL has demonstrated its capacity for sophisticated decision-making processes. The application of RL in healthcare, specifically in optimizing radiotherapy treatment plans, is the focus of this study.

In a recent study, Moreau et al. utilized RL for automating the fractioning of radiation doses in radiotherapy. They created an RL agent capable of optimizing the radiotherapy process through a simulated environment mimicking tumor growth and cellular activity. Their work inspired our research, prompting us to investigate the impact of variable cellular model parameters on the RL agent's optimization process.

In this master thesis, we extend the work of Moreau et al. by examining the robustness of RL optimization under variations in cellular model parameters. We will study how these changes affect the agent's performance in terms of controlling tumor cells and preserving healthy cells. Additionally, we will investigate how the agent adapts its learning process to these changing parameters.

Moreover, we will examine 'during treatment' patient observations, a real-world factor that could impact the effectiveness of radiotherapy treatment plans. This is a step towards more personalized treatment plans that accommodate the specificities of individual tumors and patients.

This research has the potential to contribute to more effective, personalized cancer treatments, ultimately improving patient outcomes and advancing the field of radiotherapy.

Chapter 2

Literature Review: Reinforcement Learning

Reinforcement Learning is one of the three main paradigms of machine learning where a learner (called "intelligent agent") learns how to map situations to actions in order to maximize a numerical reward. Intelligent agents are aware of their environments and take actions autonomously to achieve their goals based on a context-specific reward function. In this way, an action is taken by the intelligent agent which affects the environment and updates its state. A reward is received, after performing this action on the environment, which is proportional to the "goodness" of the action with regard to a specific reward function. Figure 2.1 shows the relations between the agent, environment, actions, states, and rewards.

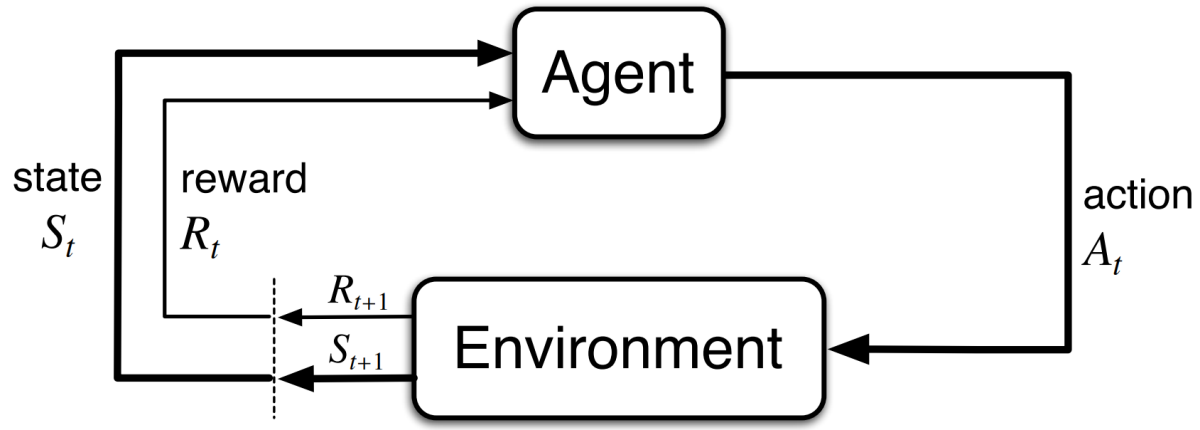


Figure 2.1: Basic Principle of Reinforcement Learning.

Where S_t, R_t, and A_t are, respectively, the state, the reward, and the action taken by the agent, at time step t. The action of time t is applied to the environment which outputs the reward and the state at time $t+1$. In the beginning, the environment is in a given state S_0, the agent then takes an action A_0, and a reward R_1 is received regarding the action. This affects the environment since it is updated in state S_1, where the process continues as such. The following sections are mainly focused on describing the context of reinforcement learning as Richard S. Sutton describes it in his book: "Reinforcement Learning: An Introduction" [3].

2.1 Reinforcement Learning Framework

2.1.1 Multi-armed bandits

Multi-armed bandits are a classic problem in reinforcement learning, where an agent is faced with the task of choosing actions from a set of available options (arms), each of which provides a reward that is uncertain. The goal of the agent is to learn to maximize the total reward over time, by balancing the exploration of

different arms (non-optimal decisions) with the exploitation of arms believed to provide high rewards (optimal decisions). In a multi-armed bandit problem, the agent starts with an initial belief about the expected reward of each arm (that can be set totally random or with a prior about the random distribution) and gradually updates these beliefs based on observed rewards. This allows the agent to make informed decisions about which arms to choose, and to explore less promising arms while exploiting those believed to provide high rewards.

Each of the k actions has an expected reward, denoted $q_*(a) := \mathbb{E}[R_t|A_t = a]$ where R_t is the corresponding reward and A_t is the action selected at time step t. If the value of each action is known, solving the problem is simply selecting always the action A_t with the highest value. However, it is assumed that $q_*(a)$ is unknown but an estimate can be computed and is denoted $Q_t(a)$. When one selects the action with the highest value (greedy actions), we say that he is **exploiting** its current knowledge of the action values. While if one selects the non-greedy actions, we say that he is **exploring**, which can be useful to improve the estimate of the non-greedy action's value. In this algorithm, called **epsilon-greedy** algorithm, the agent takes the action with the highest estimated value with probability $1 - \epsilon$, and takes a random action with probability ϵ. This allows the agent to balance exploration and exploitation, and to gradually learn which action is the best.

One easy way to compute $Q_t(a)$ is the following one :

$$Q_t(a) := \frac{sum\ of\ rewards\ when\ a\ taken\ prior\ to\ t}{number\ of\ times\ a\ taken\ prior\ to\ t} \tag{2.1}$$

This method is called **sample-average**. Note that if the denominator is 0, $Q_t(a) = 0$. As the denominator goes to infinity, the estimation tends to the real value $q_*(a)$. In this way, as mentioned before, taking the best action is simply :

$$A_t := \arg\max_a Q_t(a)$$

2.1.2 Incremental Implementation

In traditional dynamic programming algorithms (see section 9.1.1), the values of all states (see section 2.1.3 about Markov Decision Process) or state-action pairs are computed in a batch mode, where all the values are updated simultaneously. Batch mode refers to a computational mode where a set of operations are performed all at once in a batch, rather than incrementally. In this mode, all the necessary data is collected and processed, and the results are obtained in a single operation. This is computationally expensive, especially for large state spaces, and may also require a large amount of memory to store all the intermediate values.

For example, in reinforcement learning, a batch mode approach to dynamic programming (see section 9.1.1) would involve computing the values of all states or state-action pairs in a single batch. This requires the entire state space to be known in advance, and all the values to be updated simultaneously, based on the data available.

In contrast, the incremental implementation approach updates the values of states or state-action pairs incrementally, one at a time, as they are encountered in the course of the agent's interactions with the environment. This allows the algorithm to be more computationally- and memory-efficient, as it does not require the entire state space to be known in advance and does not need to store all the intermediate values. For a single action, selected $n - 1$ times :

$$Q_n = \frac{R_1 + R_2 + ... + R_{n-1}}{n - 1}$$

Given Q_n and the n-th reward R_n, the new average of all n rewards is :

$$
\begin{aligned}
Q_{n+1} &= \frac{1}{n}\sum_{i=1}^{n} R_i \\
&= \frac{1}{n}(R_n + \sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)\frac{1}{(n-1)}\sum_{i=1}^{n-1} R_i) \\
&= \frac{1}{n}(R_n + (n-1)Q_n) \\
&= \frac{1}{n}(R_n + nQ_n - Q_n) \\
&= Q_n + \frac{1}{n}(R_n - Q_n)
\end{aligned} \tag{2.2}
$$

This implementation does not require storing in memory all the previous rewards, only Q_n and n. One can notice that this equation follows the general form :

$$NewEstimate \leftarrow OldEstimate + StepSize\ [Target - OldEstimate] \tag{2.3}$$

As mentioned before, one of the key benefits of incremental implementation is its ability to efficiently update the estimated values without requiring a complete re-evaluation of all previous data. Instead, each new data point can be used to update the estimated value of the corresponding arm, while leaving the estimates for the other arms unchanged. This is particularly useful when dealing with large data sets or in situations where the environment is changing rapidly, as it allows the algorithm to adapt quickly to new information. Later on, we will see that Q-learning, Sarsa, and other reinforcement learning algorithms are based on this concept of incremental implementation.

2.1.3 Markov Decision Process

Markov Decision Process (MDP) [4] is a mathematical framework that is commonly used in reinforcement learning and even more to model decision-making problems in artificial intelligence, operations research, and control theory. It consists of a set of states, a set of actions, and a set of probabilities that govern the transition between states.

In an MDP, an agent interacts with an environment and makes decisions based on the state of the environment. At each time step, the agent observes the current state of the environment and selects an action to perform. The environment then transitions to a new state based on the action and probability distribution. The agent receives a reward for his action, which is used to evaluate the quality of the decision.

On the contrary to supervised learning problem, reinforcement learning has an approach to understanding and automating the learning and decision-making parts. It focuses on learning with an intelligent agent through interactions with its environment without relying on any supervision. In supervised learning, the learning process does not take into account the actions taken by the model, instead, it provides what a good action is. In reinforcement learning, the learning process takes into account the action performed by the agent and judges this action through a reward function. MDPs are the formalization of sequential decision-making where actions influence the instantaneous reward but also what is called delayed rewards. Thus MDPs have to deal with both, immediate and delayed rewards.

Example of MDP

A robot in a warehouse has the task to navigate to a specific location to pick up an item. The warehouse is represented by a grid of cells, and the robot can only move in four directions (up, down, left, or right). Some cells in the warehouse are blocked, and the robot cannot enter them. The robot starts at a random cell, and the goal is to reach the cell where the item is located.

In this example, the states of the MDP are the cells in the warehouse, the actions are the four possible movements (up, down, left, or right), and the rewards are determined based on the robot's actions. For

example, if the robot moves in the correct direction, it will receive a positive reward, and if it moves in the wrong direction, it will receive a negative reward. The robot's goal is to find the best policy, which is a set of actions that maximizes the expected cumulative reward over time.

Definition

A Markov Decision Process is composed of :

- States $s_t \in S$ where S is the State Space.
- Actions $a_t \in A$, where A is the Action Space.
- Rewards $r_t \in [0, R_{max}]$ where R_{max} is set to 1 by scaling the rewards.
- $T(s, a, s') : \ S \times A \longrightarrow \mathbb{P}(S)$, a transition function that gives the probability of the next state s' given state s and action a.
- Discount Factor $\gamma \in [0, 1)$

For finite MDPs (it means that $\mathbf{S}$, $\mathbf{A}$, and $\mathbf{R}$ have a finite number of elements), the random variables R_t and S_t have defined discrete probability distributions that depend only on the previous pair of action-state. The *dynamics* of the MDPs can be defined as :

$$p(s', r|s, a) := \mathbb{P}(\mathbf{S}_{t+1} = s', \mathbf{R}_{t+1} = r|\mathbf{S}_t = s, \mathbf{A}_t = a)$$

with s, s' $\in \mathbf{S}$, $a \in \mathbf{A}$ and $r \in \mathbf{R}$. The function p defines what is called *dynamics* of MDPs. It provides the probability given 4 arguments such that dynamics p : S × R × S × A $\longrightarrow$ [0,1]. In this way, each values that $\mathbf{S}_{t+1}$ and $\mathbf{R}_{t+1}$ can take, depend only on the previous state $\mathbf{S}_t$ and action $\mathbf{A}_t$. Then, the state follows the Markov Property, such that a stochastic process is memoryless. The dynamics of a finite Markov Decision Process can be completely characterized based on these probabilities given by the four-argument function p.

Given dynamics, the **state-transition probabilities** (p : S × S × A $\longrightarrow$ [0,1]) can be computed as :

$$p(s'|s, a) := \mathbb{P}(\mathbf{S}_{t+1} = s'|\mathbf{S}_t = s, \mathbf{A}_t = a) = \sum_{r \in \mathbf{R}} p(s', r|s, a)$$

The expected reward for state-action pairs can also be computed as $r : \mathbf{S} \times \mathbf{A} \longrightarrow \mathbb{R}$:

$$r(s, a) := \mathbb{E}[\mathbf{R}_t|\mathbf{S}_t = s, \mathbf{A}_t = a] = \sum_{r \in \mathbf{R}} r \sum_{s' \in \mathbf{S}} p(s', r|s, a)$$

The expected reward for state-action-next state triples can also be computed as $r : \mathbf{S} \times \mathbf{A} \times \mathbf{S} \longrightarrow \mathbb{R}$:

$$r(s, a, s') := \mathbb{E}[\mathbf{R}_t|\mathbf{S}_t = s, \mathbf{A}_t = a, \mathbf{S}_{t+1} = s'] = \sum_{r \in \mathbf{R}} r \frac{p(s', r|s, a)}{p(s'|s, a)}$$

It's important to distinguish between an agent and its environment. Determining what is part of the agent and what is part of the environment can sometimes be difficult. For the purpose of clarity, it is important to consider the environment as everything that cannot be changed by the agent. For instance, in the case of a robot, the sensors that the robot uses to perceive information from the world are considered to be a part of the environment. While the agent may have knowledge about its environment, including how the reward function is computed, this knowledge is always considered external to the agent. Therefore, it's crucial to understand that the knowledge of the agent and the boundaries between the agent and the environment are distinct. These boundaries are established when the states, actions, and rewards are chosen for the problem at hand.

Of course, MDP framework is abstract, meaning it is a simplified representation of a problem that captures its essential features. It is flexible, such that it can be applied in different ways, the time steps can refer to any successive stages of decision-making, the actions can be low-level controls or high-level decisions, and the

states can take many forms, they can be based on low-level sensations or high-level and abstract descriptions. It proposes that any problem of learning goal-directed behavior can be reduced to three signals passing back and forth between an agent and its environment: actions, states, and rewards.

Reward signal

The reward signal is the formalization of the agent's goal. For each time step, the reward takes a simple value denoted $\mathbf{R}_t$. The goal of the agent is to maximize the cumulative sum of all the rewards received for each time step (*reward hypothesis*). The reward signal has to reflect as much as possible the goal of the agent and only that. For example, an agent that learns to play chess will have a reward of +1 for winning, -1 for losing, and 0 for drawing. It reflects that no reward is accorded to the agent for taking its opponent's pieces or gaining control of the center of the board which are only subgoals, otherwise, the agent would find a way to maximize the subgoals and not the real goal. This is why there is a real trade-off between immediate and delayed rewards.

The goal of the agent in an MDP is to maximize the cumulative rewards over time. This objective is formalized as the **expected return**, denoted by G_t, which is typically defined as the sum of the rewards received by the agent at each time step :

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + R_{t+4} + ... + R_T$$

with T, the final time step. There are two situations, where T is finite and where T is infinite. In the case of finite T, the interactions between the agent and the environment can be broken down into distinct **episodes**. However, when T is infinite, the reward can become infinite as well. To address this issue, the concept of discounting is introduced :

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

where $\gamma \in [0, 1]$ is the discount factor that determines the relative importance of immediate versus delayed rewards. The value of γ is between 0 and 1, with a value of 0 (agent is said to be myopic) indicating that the agent is only concerned with maximizing immediate rewards and a value closer to 1 (agent is said farsighted) indicating that the agent is taking into account future rewards as well. Since $\gamma < 1$, the infinite sum is now bounded and its value is finite. This definition of G_t applies to both episodic and continuous tasks. In the case of episodic tasks, the infinite discounted sum of rewards will be, for example with T = 3, $R_1 = +1$, $R_2 = +1$, $R_3 = +1$, $R_4 = 0$, $R_5 = 0$, ... and so on. Note that an important property is that $G_t = R_{t+1} + \gamma G_{t+1}$.

Policy and value function

Now that the objective of the agent is a bit more detailed, the purpose is to put in context the concept of reward in the whole process with action and state space. The reward that an agent will receive is dependent on the action it will take. A **policy** is a mapping between the state space $\mathbf{S}$ and the action space $\mathbf{A}$. In this way, if an agent follows a policy π at time t, then $\pi(a|s)$ is nothing else than the probability of $\mathbf{A}_t = a$ if $\mathbf{S}_t = s$.

A **value function** is a function that assigns a value to a state or a state-action pair, indicating the long-term expected reward if the agent follows a certain policy. There are two types of value functions: state-value functions (denoted $v_\pi(s)$), which assign a value to a state, and action-value functions (denoted $q_\pi(s, a)$), which assign a value to a state-action pair.

The expected return, when the environment is in state $\mathbf{S}_t = s$ and following the policy π, is the value function :

$$v_\pi(s) := \mathbb{E}_\pi[\mathbf{G}_t|\mathbf{S}_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|\mathbf{S}_t = s], \quad \forall s \in \mathbf{S}$$

v_π is called the **state-value** function for policy π.

The expected return, when the environment is in state $\mathbf{S}_t = s$ with taking the action $\mathbf{A}_t = a$ and following the policy π, is the value function :

$$q_\pi(s,a) := \mathbb{E}_\pi[\mathbf{G}_t|\mathbf{S}_t = s, \mathbf{A}_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|\mathbf{S}_t = s, \mathbf{A}_t = a], \quad \forall s \in \mathbf{S}$$

q_π is called the **action-value** function for policy π.

State-value function and action-value functions can be estimated by Monte Carlo Methods, averaging over random samples will provide a good estimate. If an agent follows a specific policy π and keeps track of the average return for each state encountered, then the average will converge to the state's value, $v_\pi(s)$, as the number of times that state is encountered increases. Similarly, if the agent keeps track of separate averages for each action taken in each state, then these averages will converge to the action values, $q_\pi(s,a)$.

From the expression of v_π, the Bellman Equation can be derived :

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[\mathbf{G}_t|\mathbf{S}_t = s] \\ &= \mathbb{E}_\pi[\mathbf{R}_{t+1} + \gamma \mathbf{G}_{t+1}|\mathbf{S}_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma\, \mathbb{E}_\pi[\mathbf{G}_{t+1}|\mathbf{S}_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')], \quad \forall s \in \mathbf{S} \end{aligned} \tag{2.4}$$

This equation expresses the relationship a given state has with its successor states. Bellman equation [5] is important in order to provide an estimate/approximation of the state-value function v_π. It decomposes the value function between instant and delayed values such that it is easier to compute the value function $v_\pi(s)$.

In a similar way, the Bellman equation can be derived for $q_\pi(s,a)$:

$$q_\pi(s,a) = \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')], \quad \forall s \in \mathbf{S}$$

Optimal policy and value function

A policy π is said better than a policy $\pi^{'}$, if its expected return is greater than the one for $\pi^{'}$:

$$\pi \geq \pi^{'} \;\; if \;\; v_\pi \geq v_{\pi'}, \quad \forall s \in \mathbf{S}$$

The goal of a reinforcement learning agent is to find the best policy, π, to maximize the expected return. The optimal policy denoted π^*, is defined as the best policy that can be found using this criterion. Also, there can be more than only one policy and all of them share the same state-value function called the optimal state-value function v_* :

$$v_*(s) := \max_\pi v_\pi(s), \quad \forall s \in \mathbf{S}$$

Optimal policies share the same action-state function called the optimal action-value function q_* :

$$q_*(s,a) := \max_\pi q_\pi(s,a), \quad \forall s \in \mathbf{S}, \;\; \forall a \in \mathbf{A}$$

The **Bellman optimality** equation permits to avoid making reference to any specific policy. This equation expresses the fact that the expected return of the best action at a given state must be equal to the value of this state under an optimal policy :

$$\begin{aligned} v_*(s) &= \max_{a \in \mathbf{A}} q_{\pi_*}(s,a) \\ &= \max_{a \in \mathbf{A}} \mathbb{E}_{\pi_*}[\mathbf{G}_t|\mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \max_{a \in \mathbf{A}} \mathbb{E}_{\pi_*}[\mathbf{R}_{t+1} + \gamma \mathbf{G}_{t+1}|\mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \max_{a \in \mathbf{A}} \mathbb{E}[\mathbf{R}_{t+1} + \gamma v_*(\mathbf{S}_{t+1})|\mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \max_{a \in \mathbf{A}} \sum_{s', r} p(s', r|s, a)[r + \gamma v_*(s')] \end{aligned} \tag{2.5}$$

In a similar way, the Bellman optimality equation for q_* is :

$$\begin{aligned} q_*(s,a) &= \max_{a \in \mathbf{A}} \mathbb{E}[\mathbf{R}_{t+1} + \gamma \max_{a'} q_*(\mathbf{S}_{t+1}, a') | \mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q_*(s', a')] \end{aligned} \tag{2.6}$$

The Bellman optimality equation for v_* is a system of equations that can be used to find the optimal policy when dealing with finite Markov Decision processes. If there are n states, the system to solve is made of n equations and can be solved using methods for non-linear equations systems (if the dynamics p are known). However, in practice, the computational resources are not sufficient and the Markov property is not verified such that an agent will rarely find the optimal policy. Hence, it needs to approximate such policies and this is the purpose of Reinforcement Learning that aims to find what is the good action to perform in a set of frequently encountered states, rather than to make the computational effort for infrequently encountered states.

Agent category

An agent can be based on policies and/or value functions. In this way, it is possible to distinguish different types of agents. (1) The value-based agent. This agent knows an approximate of the value function and uses it to judge if an action is better than another. In this case, there might not be an explicit policy and then when one needs the policy, this one is constructed based on the value function. (2) A policy-based agent. This agent has a representation of the policy so, a mapping between states and actions, and no explicit value function. (3) Actor critic agent is an agent that has an explicit value function and an explicit policy. The policy is the actor and the value is the critic.

Another categorization can be made. (1) Model-free agent means that the agent knows the policy and/or the value function but has no model. (2) A model-based agent is an agent that knows a model that represents, at least partially, the environment. This agent can also know optionally the value and/or the policy. Actor-critic agents can be model free or not.

2.2 Reinforcement learning algorithms

There are many different ways to solve reinforcement learning problems. Framing the problem as a Markov Decision Process allows using different sorts of algorithms in order to find optimal policies. In this work, we will introduce briefly 4 types of algorithms: Dynamic Programming (DP) methods, Monte Carlo (MC) Methods, Temporal-Difference (TD) methods, and deep learning approaches. Each method has a complete description in the annex which has been built from the book of Richard S. Sutton, "An Introduction to reinforcement learning" [3].

All these algorithms (except the deep learning one) have the same basic principle:

1. Learning the state-value function v_π from a given policy π. This step is called **Policy Evaluation** or **Policy Prediction**.
2. Finding better policies from the state-value function v. This step is called **Policy Improvement**.
3. Iterating by alternating between policy evaluation and policy improvement. This step is called **Policy Iteration** or sometimes **Control**.

2.2.1 Dynamic Programming methods

Dynamic Programming (DP) is a general algorithmic technique used for solving complex problems by breaking them down into simpler subproblems. It is used to compute the optimal policy and optimal value function of a Markov Decision Process (MDP) by iteratively improving the value function estimate of each state in the MDP. The main bottleneck of DP is that it requires complete knowledge of the environment dynamics, which is often not available in many real-world scenarios. Another disadvantage of DP is that it is computationally expensive, and the time complexity grows exponentially with the size of the state space due to the well-known curse of dimensionality. A more detailed section about **Dynamic Programming** can be found in the section 9.1.1 and also a toy example in the section **??**, both in the annex.

2.2.2 Monte Carlo methods

Monte Carlo (MC) methods rely on averaging samples of returns to estimate the value function. MC methods are model-free, which means they do not require knowledge of the transition probabilities and reward function of the MDP. The main bottleneck of MC methods is that they require complete episodes to update the value function estimate. This means that MC methods are only suitable for episodic tasks, and cannot be used in continuous tasks. Additionally, MC methods are often biased and require a large number of episodes to achieve convergence. A more detailed section about **Monte Carlo** can be found in the annex 9.1.2.

2.2.3 Temporal-Difference Learning

Temporal-Difference (TD) Learning mixes up the ideas of Monte Carlo methods and Dynamic Programming. Like Monte Carlo methods, TD methods can learn from experiences without the knowledge of the environment's transitions. The main similarity between DP and TD methods is that they both use a bootstrapping approach, where the estimated value of a state or action is updated based on the estimated value of subsequent states or actions. In DP, this is done by breaking down the problem into smaller subproblems and recursively computing the value of each subproblem. In TD, the value estimate of a state is updated based on the difference between the observed reward and the estimated value of the next state. The main difference between DP, MC, and TD methods is the way they predict the policy (policy evaluation), while all of them use the same principle to find the optimal policy.

TD Policy Prediction

As mentioned earlier, TD methods and MC methods use experience to determine the value function v_π of a policy π. A simple every-visit Monte Carlo method is :

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \qquad MC\ methods$$

This equation is familiar, it reminds the equation 2.3 where the target is the return, for $V(S_t)$. It is called $constant - \alpha$ MC because α is a constant step-size parameter. In MC methods, the update is made on an episode-by-episode basis because the return is not available before the end of the episode. In TD methods, the update is made every time step using the observed reward R_{t+1} and the estimate of $V(S_{t+1})$:

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \qquad TD\ methods \tag{2.7}$$

where the target is now $R_{t+1} + \gamma V(S_{t+1})$. This method is called TD(0) or one-step TD because it is a special case of the TD(λ) and n-step TD methods. They are called "temporal-difference" methods because they estimate the values of states or state-action pairs by computing the difference between the expected return and the observed return at the current time step. Thus, Monte Carlo methods use a sample return to estimate the expected return, while DP methods use the current estimate of the value function as an approximation for the true expected return. TD methods also use an estimate, incorporating elements of both Monte Carlo and DP methods by sampling the expected values and using the current estimate of the value function instead of the true expected return. This combination allows combining the benefits of both, Monte Carlo and DP methods.

TD error can be defined : $\delta_t := R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$. It measures the difference between the estimated value of S_t and the better estimate $R_{t+1} + \gamma V(S_{t+1})$.

Advantages of TD

This section describes several advantages of TD methods compared to MC and DP methods. TD methods do not require a model of the environment, its reward, and next-state probability distributions (Advantage over DP). Only one-time step to wait to compute the return. It is, even more, a good property if the problem has very long episodes where the learning will be slow or if the application is a continuing task with no episode at all (Advantage over MC). It can be shown that convergence is ensured for TD(0) if the step-size parameter is sufficiently small and with a probability of 1 if the step-size decreases over iterations.

Sarsa: On-policy TD

Similarly, in MC methods, a trade-off is present also in TD methods, between exploration and exploitation. Thus, there exist on- and off-policy methods. Sarsa is an on-policy method, meaning that it needs to learn an action-value function $q_\pi(s, a)$ for each state $s \in \mathcal{S}$ and for each action $a \in \mathcal{A}$. It can be shown that the theorems ensuring the convergence of the value function approximation computed from the equation 2.7 to v_π, also ensure the convergence of the action value function (both are Markov chains). Then,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2.8}$$

As said before, the update is made at every time step. Thus, equation 2.8 is followed after each transition to a non-terminal state. If a terminal state is reached: $Q(S_{t+1}, A_{t+1}) = 0$. Therefore, this is why this algorithm is named Sarsa, because it uses every element of the quintuple of events $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$. The convergence of Sarsa is ensured if all state-action pairs are visited an infinite number of times. Algorithm 1 describes the Sarsa algorithm.

Algorithm 1 Sarsa Algorithm

$Q(s, a)$, *randomly* $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$
$Q(s, a) = 0$, *in terminal state*
while *S is not terminal* **do**
  Initialize S
  Choose A from S using policy derived from Q
  for each *step in episode* **do**
    Take action A, observe R, S′
    Choose A′ from S′ using policy derived from Q
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma Q(S', A') - Q(S, A)]$
    $S \leftarrow S'$
    $A \leftarrow A'$
  end for
end while
Ensure: $Q \approx q_*$

Q-Learning : Off-policy TD

Unlike Sarsa, Q-learning is an off-policy algorithm. The update of Q in Q-learning algorithm is as follows:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{2.9}$$

Equation 2.9 highlights that the update is no more made on the next action selected from the policy but rather on the action that leads to the highest action value function on the next state.

Algorithm 2 Q-Learning Algorithm

$Q(s,a)$, *randomly* $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$
$Q(s,a) = 0$, *in terminal state*
while *S is not terminal* **do**
  Initialize S
  for each *step in episode* **do**
    Choose A from S using policy derived from Q
    Take action A, observe R, S′
    $Q(S,A) \leftarrow Q(S,A) + \alpha[R + \gamma max_a Q(S',a) - Q(S,A)]$
    $S \leftarrow S'$
  end for
end while
Ensure: $Q \approx q_*$

Algorithm 2 describes the pseudo-code to implement the Q-learning algorithm. One can notice that choosing an action A from S is now made in the *for loop* because the action is always taken from an ϵ-greedy policy. Thus, there is no direct update of action as in Sarsa: $A \leftarrow A'$.

Difference between Sarsa and Q-learning

Sarsa is an on-policy algorithm meaning that it uses and updates the same policy all the time following a given action, while Q-learning uses the maximum of $Q(S_{t+1}, a)$ over all possible actions a for the next step. It is similar to following an ϵ-greedy policy where $\epsilon = 0$ (no exploration, it always selects the best action with regard to Q). So, while Sarsa uses π to choose an action and to update Q, the Q-learning algorithm uses π as a behavior policy to choose an action but updates Q with a target policy. A toy example comparing these two algorithms can be found in the annex, in the section **??**.

Expected Sarsa

Expected Sarsa is based on the same principle as the Q-learning algorithm, instead of using the maximum over the next state-action pairs, it is replaced by the expected value (hence, it takes into account the probability an action is taken at a given state, following the current policy). The update rule is the following one :

$$\begin{aligned} Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \mathbb{E}[Q(S_{t+1}, A_{t+1}|S_{t+1}] - Q(S_t, A_t)] \\ &\leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)] \end{aligned} \tag{2.10}$$

Equation 2.10 follows the same scheme of Q-learning. The advantage of Expected Sarsa is that it eliminates the variance due to the random selection of action A_{t+1}, such that it should perform better. Expected Sarsa can be used as an on- or off-policy algorithm (even if it is more used as an off-policy algorithm).

Double Learning

As Q-learning and Sarsa are based on greedy policies, it involves a maximization operation. This maximization operation can lead to significant bias due to the fact that the maximum is performed with overestimated values. Hence, it does not provide an estimate of the maximum value. Let's say that the true value of the action-value function are all 0 ($q(s,a) = 0 \quad \forall s \in \mathcal{S}, a \in \mathcal{A}$). The estimated action-values, $Q(s,a)$ are different from 0, with some negative and positive values. In this way, the maximum true action-value is 0, while the maximum estimated action-value is positive. This is what is called **maximization bias**. It occurs when the maximum of the estimates is used as an estimate of the maximum of the true values.

The idea about double learning is that the problem comes from the fact that the same samples are used to estimate the action-values and to determine the maximizing action. Double learning defines two estimates of $q(a)$, $Q_1(a)$ and $Q_2(a) \quad \forall a \in \mathcal{A}$. Q_1 will be used to determine the maximizing action $A^* = \arg\max_a Q_1(a)$, while Q_2 will provide the estimate of the given action A^*, $Q_2(A^*) = Q_2(\arg\max_a Q_1(a))$. Then, the estimate is now unbiased because $\mathbb{E}[Q_2(A^*)] = q(A^*)$. This process is also performed by inverting the roles of Q_1 and Q_2. Thus, the update rule for **Double Q-learning** (it exists also Double Expected Sarsa) is the following one :

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha[R_{t+1} + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t)] \tag{2.11}$$

where at each time step, with a probability of one half, Q_1 or Q_2 is updated.

Algorithm 3 Double Q-Learning Algorithm

$Q_1(s,a),\ Q_2(s,a),$ *randomly* $\forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s)$
$Q_1(s,a) = Q_2(s,a) = 0,$ *in terminal state*
while *S is not terminal* **do**
  Initialize S
  for each *step in episode* **do**
    Choose A from S using policy derived from Q_1 *and* Q_2
    Take action A, observe R, S'
    if $\theta < 0.5,$ *with* $\theta \sim \mathcal{U}(0,1)$ **then**
      $Q_1(S,A) \leftarrow Q_1(S,A) + \alpha[R + \gamma Q_2(S', \arg\max_a Q_1(S',a)) - Q_1(S,A)]$
    else
      $Q_2(S,A) \leftarrow Q_2(S,A) + \alpha[R + \gamma Q_1(S', \arg\max_a Q_2(S',a)) - Q_2(S,A)]$
    end if
    $S \leftarrow S'$
  end for
end while

2.2.4 Comparison DP, MC, and TD methods

To illustrate the differences between these techniques, we can consider the problem of learning to play a game of chess. In DP, we would need to consider the entire state space of the game, which is infeasible due to its large size. In MC, we would need to play several complete games of chess to estimate the value function, which is also infeasible due to the complexity of the game. In TD, we would update the value function estimate after each move, based on the difference between the predicted value and the observed value, which makes it more efficient and effective for learning to play chess.

In conclusion, each of these techniques has its own advantages and disadvantages, and the choice of which technique to use depends on the specific problem and the available resources. DP is more efficient in terms of sample complexity but requires complete knowledge of the environment. MC methods are more sample-efficient but can be biased and require complete episodes. TD methods combine the advantages of DP and MC methods, but are sensitive to the choice of the learning rate and require more samples than DP methods. In terms of performance, Dynamic Programming can converge quickly to the optimal solution if the model of the environment is known, while Monte Carlo methods and Temporal-Difference Learning can take longer to converge but can still find an optimal solution without knowing the full model of the environment. The choice of which method to use will depend on the specific requirements of the problem.

2.2.5 Deep Reinforcement Learning

Introduction

Deep Reinforcement Learning (DRL) is a subfield of Reinforcement Learning (RL) that has shown remarkable success in solving a wide range of applications, including gaming, robotics, finance, healthcare, and many more. DRL has achieved superhuman performance in games such as Go, chess, and Atari games, where it was once unimaginable that artificial intelligence could outperform humans. In robotics, DRL has been used to train agents to perform complex tasks such as grasping objects, navigating environments, and flying drones. In finance, DRL has been applied to algorithmic trading, portfolio optimization, and risk management. In healthcare, DRL has been used to develop personalized treatment plans and assist with medical diagnosis.

Comparison with Tabular Algorithms

DRL and tabular algorithms are both methods for learning optimal policies. However, they differ in their ability to handle large and complex state and action spaces, as well as their sample efficiency and generalization capabilities.

Tabular algorithms, such as Q-learning and SARSA, rely on discretizing the state and action spaces and maintaining a table of state-action values. This makes them highly sample efficient and easy to interpret, but they can struggle to handle continuous and high-dimensional state and action spaces. In addition, tabular algorithms can suffer from poor generalization, as they require a large number of samples to accurately estimate the state-action values for each state and action (remind the convergence condition where tabular algorithms quite often require an infinite number of visits for each state-action pair) [3].

On the other hand, DRL uses deep neural networks to approximate the value or policy function of the agent, allowing it to handle large and continuous state and action spaces. DRL can also leverage the representational power of deep neural networks to learn complex patterns and features in the data, leading to better generalization and scalability. However, DRL algorithms suffer from the amount of sample needed to train the neural network (which needs to be very large) and are computationally expensive (way more than tabular algorithms). They may also suffer from instability and non-convergence issues.

Training with Neural Networks

In DRL, neural networks are trained using backpropagation and stochastic gradient descent, and the Bellman equation (see equation 2.9) is used to update the weights of the network. The Bellman equation is a fundamental concept in RL that relates the value of a state or state-action pair to the expected future reward and the value of the next state or state-action pair. The use of deep neural networks allows DRL algorithms to learn complex mappings between states, actions, and rewards, enabling the agent to make more informed decisions.

Experience Replay

DRL algorithms typically use an experience replay buffer to store and randomly sample transitions from the agent's experience. This helps to reduce the variance of the updates and stabilize the learning process [6]. Experience replay allows the agent to learn from past experiences and to decorrelate the training data, preventing the network from overfitting to the current state of the environment.

Neural Fitted Q-Learning

In Q-learning, an action-value table $Q(s, a, \theta_0)$[1] is initialized randomly, quite often close to 0 in order to avoid slow learning. Neural Fitted Q-Learning approximates the action-value function with a neural network and no more a Q-table. Thus, states are provided as input to a neural network and a different output is produced for each possible action. This is the same goal as in Q-Learning, finding :

$$\pi^*(s) = \arg\max_a Q^*(s, a)$$

As neural network are universal function approximators, the goal is to build a neural network that looks like Q^*. The target value is the reference for updating the Q-values, $Q(s, a, \theta_k)$, at the k-th iteration :

$$y_k = r + \gamma \max_{a'} Q(s', a', \theta_k) \tag{2.12}$$

with θ_k, the weights of the neural network at the k-th iteration [7].

The stochastic gradient descent has to minimize the square loss function (this function should ensure that it converges to the expected value y_k) :

$$L_{DQN} = (Q(s, a, \theta_k) - y_k)^2 \tag{2.13}$$

where the weights update's rule can be derived :

[1] θ_0 are the initial parameters (initial neural network weights).

$$\theta_{k+1} = \theta_k + \alpha(y_k - Q(s, a, \theta_k))\nabla_{\theta_k} Q(s, a, \theta_k) \tag{2.14}$$

with α, the learning rate. Neural Fitted Q-Learning [7] suffers from two things : (1) the target y_k also changes when the neural network weights θ_k are updated, leading to errors that propagate and since neural network have great generalization abilities, the convergence may be slow, (2) the same problem occurring in Q-Learning, the maximization operation leads to a positive bias because the maximum of the estimates is used as an estimate of the maximum of the true action-values.

Deep Q-Network

Deep Q-Network (DQN) has become one of the most widely used RL algorithms. The main idea of DQN is also to use a neural network to approximate the Q-values in order to handle high-dimensional state spaces. Deep Q-network uses many concepts already introduced in this work :

- **Experience Replay Buffer** : Agent stores experiences in a memory buffer (state, action, reward, next state).
- **Q-Network**: Agent uses a deep neural network to approximate the action-value function Q(s,a), which estimates the expected discounted reward of taking action a in state s and following the optimal policy thereafter.
- **Target Q-Network**: The agent uses a separate target network to compute the target values for updating the Q-network. The target network is a copy of the Q-network that is updated every C iterations.
- **Epsilon-Greedy Exploration**: The agent uses an ϵ-greedy policy to balance exploration and exploitation.
- **Update Rule**: The agent updates the Q-network parameters using the Bellman equation and the square loss function.

Target Q-Network

In DQN, the Q-network is updated using the Bellman equation (see equation 2.9). However, instabilities and overestimation of Q-values can arise due to the maximization bias (as seen in section 2.2.3). In DQN, the problem of maximization bias is exacerbated by the use of a deep neural network to approximate the action-value function, which can introduce high variance in the estimates of the Q-values. Thus, the main purpose of the target Q-network is to provide a stable target for updating the Q-network parameters.

To address the problem of maximization bias, the target Q-network is introduced in DQN as a separate network that is updated less frequently than the Q-network [7]. By using a separate network to estimate the target Q-values, the DQN algorithm can reduce the correlation between the target and the Q-values used in the update rule, which can reduce the impact of maximization bias.

Chapter 3

Literature Review: Radiotherapy

3.1 Overview & Basic Principles

3.1.1 Cancer & Healthy Cells

Cancer is a disease occurring when cells grow in an uncontrollable manner and spread to other parts of the body through the blood and lymphatic system. Contrarily to benign tumors, cancerous tumors are malignant and can spread into nearby tissues or other parts of the body [8]. The difference between cancer cells and normal cells is the absence of a signal telling them to stop dividing. Indeed, normal cells have the property to stop dividing if the surrounding conditions are not optimal, in other words, if the nutrient levels are too low or if they are too many neighboring cells.

Genetic changes are involved in cancer, leading to modifying the cells' functions, especially how they grow and divide. Normal or "healthy" cells go through a process called "cell cycle", which can be different in terms of duration (denoted as T_c and called cell cycle time) depending on the type of cell. From Malaise et al. [9], the mean T_c is about 43.5 hours for Squamous cell carcinoma (cancer developed from an epithelial tissue, for example, the skin), about 34.9 hours for adenocarcinoma, and about 102 hours for melanoma. This cycle has the purpose to describe the process, for a cell, of growing and dividing. Cells grow and divide in order to ensure the human body functions and structures over time. Sometimes, this process goes wrong and Deoxyribonucleic acid (DNA) is damaged affecting cell growth and division, leading to cells that do not die when they should and new cells are formed when the body does not need them. These extra cells will form a non-functional mass of tissue called a tumor [10] [8].

3.1.2 Radiotherapy treatment

Radiotherapy (RT) has the purpose of using ionizing radiation to treat diseases, such as cancers. The ultimate cancer cure would be to kill every cancer cell without damaging any healthy cells. Nowadays, this treatment does not exist but a lot of research is made in the field of radiotherapy, mainly trying to target the cancerous cells by delivering a high dose to a considered volume while keeping the dose to surrounding healthy cells as low as possible.

Nowadays, radiotherapy is widely used. It is highly cost-effective even if the radiotherapy equipment is a significant initial investment. The life cycle of this equipment is quite long such that a lot of patients can be treated. In the last 30 years, 3D visualization of gross tumor volume as well as organs at risk has been possible thanks to Magnetic Resonance Imaging (MRI) and Computed tomography (CT). It has allowed the development of a treatment planning discipline where it studies the impact of radiotherapy treatment on the cancer cure.

3.1.3 RT treatment example

The dose delivery in radiotherapy treatment is cancer dependent. The British Columbia agency [11] has conducted a 20-year follow-up study where it studies breast cancer on 318 patients that received 37.5 Gy in 16 fractions during the middle of a 6-month chemotherapy course. Another type of treatment, "CHART" (Continuous Hyperfractionated Accelerated Radiotherapy) is constituted of 36 fractions of 1.5 Gy, three times a day for 12 consecutive days. Conventional radiotherapy is a more general process where 30 fractions of 2 Gy are given, for a total of 60 Gy in 6 weeks. CHART regimen has shown superior outcomes compared to the conventional treatment plans, however, it is logistically difficult to set up. This is why treatment planning in radiotherapy is still studied a lot to try to find more optimal ways of delivering the same or less total radiation dose, with the same or less number of fractions, in the same or less duration, while keeping the main purpose: destroying the cancerous cells.

3.1.4 Working principle of RT

The principle behind radiotherapy is, as mentioned before, to ionize with radiation in a given area. Ionizing radiation can be divided into directly ionizing and indirectly ionizing. In the directly ionizing process, particles are sent with adequate kinetic energy to the given area but they can produce chemical and biological damages in the absorbing medium. On the other hand, X- and γ- rays are both electromagnetic radiations and then indirectly ionizing because it does not produce chemical or biological changes.

Irradiation of a cell can lead to its death. A cell is considered dead when it has lost its reproductive capacity. Cells can die in many different ways after radiation: apoptosis, autophagy, necrosis, mitotic catastrophe, and senescence-like growth arrest [8].

- Apoptosis is a programmed death that occurs in multi-cellular organisms. This is a self-destruction of the cell and it can be due to DNA damage or problems in the cell's development.
- Autophagy is a cellular recycling process occurring as a response to cellular stress such as nutrient deprivation or pathogen invasion. Thus, it can lead to cell death in some cases.
- Necrosis is a type of cell death occurring as a response to different cellular stress such as physical trauma, infections, toxins, or inflammation. This is not a programmed (accidental) process.
- Mitotic catastrophe is a type of cell death occurring as a result of errors during cell division (mitosis).
- Senescence-like growth arrest refers to a state where a cell is not able to divide anymore. It does not imply necessarily cell death.

3.2 Cell-based Computational Model

3.2.1 Principle

A cell-based computation model has the goal to simulate the behavior of human cells and their development. Each cell that belongs to the system has and follows a determined behavior that can be modeled by equations. Cells can also interact with each other and their environment such that complex patterns of the system are modeled.

In the context of radiotherapy, cell-based computational models are used to simulate the behavior of cancer cells in response to radiation treatment. The cells are modeled as individual units that go through the cell cycle and migrate in response to radiation exposure.

These models are used to predict the effectiveness of different radiation treatment plans, to identify the most effective dosing and scheduling of treatment, to study the interactions between cancer cells and the surrounding tissues, or also to evaluate the potential side effects of radiation treatment on healthy tissues. Modeling cancer cells and their behavior can help to improve our understanding of the underlying biology of cancer and to develop more effective and personalized radiation treatments.

Cell-based computational models can be represented as a 2D grid environment (lattice-based model) where healthy and cancer cells can proliferate. Such models require significant computational resources to run. Experimental data is quite often used to refine the model parameters in order to perform accurate simulations. Thus, the accuracy of the model predictions depends on the quality and quantity of the data used to build the model, as well as the accuracy of the underlying biological and physical assumptions.

3.2.2 Radiation Response

If radiation therapy is included in the cell-based model then, the response of cells to radiation doses must be integrated. It exists different ways of modeling the radiation response. It can be made at a molecular level, where radiation kills cells by producing secondary charged particles which in turn produce damage in DNA (single-strand break or double-strand break) [12]. Target theory brings the idea that there may be specific regions of the DNA that are important to maintain the reproductive capacity of cells. These specific regions are considered as specific targets for radiation damage. Then, the radiation damage amount is proportional to the number of targets inactivated. Another type of model is the **Linear-Quadratic (LQ)** model. This model is often used to describe the survival fraction of cells with regards to a dose d. Thus, the cell survival curve is represented using a second-order polynomial with a constant term to bring the survival fraction to 1 if no dose is applied. Then, the probability of survival is: $p(survival) = exp(-\alpha D - \beta D^2)$. It will be shown later that this model is used in our simulation. The linear-Quadratic model describes in a quite accurate way the survival fractions from low radiation doses until 3 Gy. However, it has been difficult to make accurate measurements of cell dying by radiation for doses below 1 Gy meaning that this model can suffer in accuracy in this range [13]. In our model, agents can choose radiation doses above 3 Gy, meaning that the model can suffer in accuracy in this range also.

3.3 Existing cellular models

Many cellular models have been developed for years now. Advanced simulation methods are expected to improve the understanding of how a cellular system behaves. For example, it could lead to significant improvements in terms of radiotherapy treatment planning [1]. There exist two groups : (1) lattice-based methods where cells have a specific place on a grid, and (2) off-lattice methods, where cells have a free positioning.

The optimization of radiotherapy is a popular research topic due to its importance in cancer treatment. There are two main approaches to treatment planning: forward planning (trial-and-error) and inverse planning (computer-aided). Forward planning improves the treatment plan by conducting experiments with different radiation doses. On the other hand, for inverse planning methods, this is the computer system that tries to calculate the optimal settings for the radiation therapy. Both methods have the goal to maximize the tumor control probability (TCP) and to minimize the Normal Tissue Complication Probability (NTCP).

For the optimization part, there are also two main approaches : (1) dose-based models and (2) radiobiological models. For radiobiological models, the simulation will try to give us the biological effects of radiotherapy on the patient, which are hard to obtain.

In this way, such simulations can use different ways to model the tumor response. For example, there exist models based on ordinary differential equations [14] to model the response of tumors to radiotherapy that can be solved through finite elements or boundary-integral methods. Some models try to characterize the multiscale process behind cancers, from the subcellular level to the tissue level, behind cancer [15]. Cells can also be represented as sticky and elastic particles, capable of growth and migration [16]. 3D models based on Monte Carlo [17] simulations also have been explored. Diffusion equations [18] have been used to model the spread of tumors in space. Cellular Potts Model defines cells as discrete entities on a lattice and an Hamiltonian energy function describes the interactions between cells and the environment. Off-Lattice Cell models, unlike the Cellular Potts Model, describe cells as continuous entities (it has the advantage to get more flexibility for representing the shape of the cells), whereas interactions are described with physical principles.

Chapter 4

Materials

This section describes the complete simulation process, the cellular model used, the tumor growth simulation, the reinforcement learning agents that will be used. As mentioned before, this master thesis has the paper of G. Moreau et al. as a starting point [19].

4.1 Cellular Model for simulations

As seen in section 3.3, many models have been developed to characterize the growth of tumor cells. G. Moreau and al. use a model based on the model of O'Neil [2] combined with the model of Jalalimanesh and al. [1]. O'Neil presented a model, a lattice-gas cellular automaton which can be represented as a tuple (G, E, U, f), with G a grid of cells, E, a set of states (cell cycle), U, a set of neighborhood cells and f, a local rule. The representation of the model is a 2D 50-by-50 grid where each pixel on the grid contains : (1) a list of cells on the pixel, (2) an amount of nutrients, and (3) a nutrient source (if present). This model is constituted of the cell cycle presented in figure 4.1. The proliferation of the human cell requires about 24 hours on average to be completed and is the baseline of this part of the model. However, the cell cycle can vary a lot for different types of cells [20]. Even for our considered purpose, quite often the Gap 1 step can vary in terms of duration [21]. However, in order to be consistent, Figure 4.1 describes the cell cycle and how to go from one state to another for each cell on the grid. Further details are provided later on, for the glucose and oxygen thresholds (same for "high cell density" in terms of the number of cells).

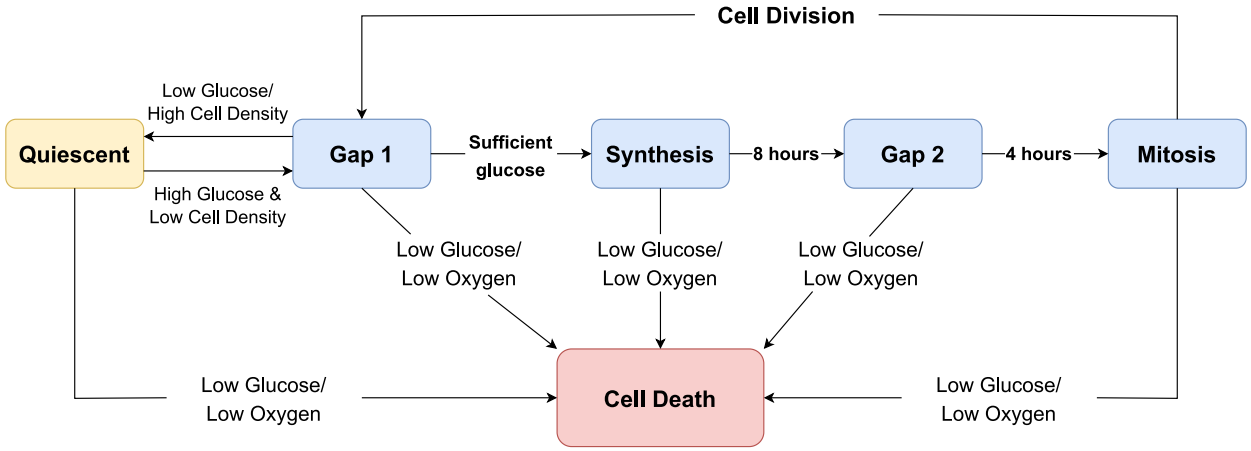


Figure 4.1: Cell cycle flow chart.

Moreau et al. built their model based on O'Neil's model, which only considers glucose as a nutrient. This simplifies the model because it ignores the role of oxygen in Adenosine Triphosphate (ATP) production, which is discussed in section 4.1.3. Oxygen is also a factor in the cell's radiosensitivity and this depends on the stage of the cell cycle.

4.1.1 Cell Cycle

As explained earlier, cells go through the cell cycle. The cell cycle is the sequence of events that a cell goes through as it grows and divides into two daughter cells. There are two main stages: the interphase (G1, S, G2) and the mitotic phase (M). In interphase, the cell grows and replicates its DNA in preparation for cell division. In the mitotic phase, the cell physically divides into two genetically identical daughter cells. The cell cycle is a regulated process that ensures the proper duplication and distribution of genetic material to the daughter cells. In this model, the duration [20] [22] of each part of the cell cycle is :

1. Gap 1 (G1): Cell Growth and preparation for DNA replication. This is the most varying part in terms of duration. It can go from 5 to 12 hours.
2. Synthesis (S): DNA replication. This part lasts between 8-10 hours.
3. Gap 2 (G2): Cell growth and preparation for Mitosis. It lasts 2 to 4 hours.
4. Mitosis (M): Formation of two daughter cells. This part does not vary a lot and is the shorter one. It generally lasts between half an hour and an hour.

In Gap 1, cells take up nutrients and grow while in Gap 2, cells continue to grow and prepare for Mitosis by synthesizing the necessary proteins. Each hour corresponds to a single time step in the simulation. Thus, 24 time steps are required for a 24-hours cell cycle. Notice that a cell can also go out of the cell cycle and goes in **Quiescence** (G0).

A cell is said to be in a **quiescent** state when it is not actively dividing. Cells have to know when to stop dividing themselves in order to sustain healthy tissue. If the surrounding cell density is too high, continued replication could cause deficient amounts of nutrients to sustain the growing population or result in a lack of physical space for individual cells to adequately grow. This is where cells go in the Quiescent state. Those checks are usually done in Gap 1 step during the simulation. Then, if the surrounding conditions become suitable for replication, the cell can re-enter the cell cycle.

4.1.2 Cell survival curve

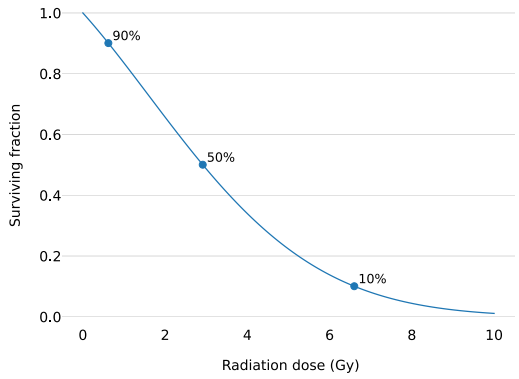


Figure 4.2: Cell survival curve for cells irradiated with effect dose (ED).

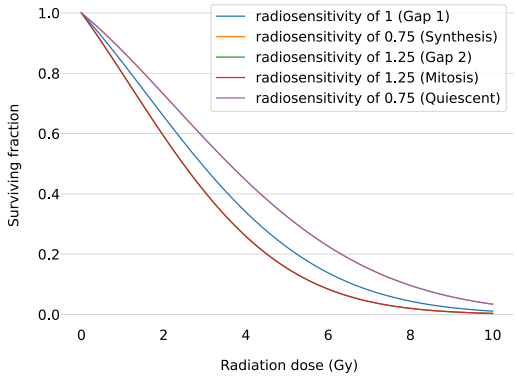


Figure 4.3: Cell survival curve for different radiosensitivities.

Figure 4.2 shows the cell survival curve for cells irradiated with a dose d. As described by Michael Joiner and Albert van der Kogel [8], a cell survival curve is a plot of surviving fraction against radiation dose. With a linear scale, the survival curve should follow a sigmoid theoretical curve. Figure 4.2 highlights also the effect dose (ED) for 10, 50, and 90 percent ($ED_{10}, ED_{50}, ED_{90}$)). A Linear Quadratic (LQ) [23] [1] [19] function is used to model the response of cancer cells to radiotherapy :

$$SF(D) = exp(-\alpha D - \beta D^2) \tag{4.1}$$

where SF represents the fraction of cancer cells surviving irradiation with a dose D (Gray) [24]. α $[Gy^{-1}]$ and β $[Gy^{-2}]$ are parameters. In Jalalimanesh and al. paper [1], a modified LQ function is used to reflect the survival probability of a cell to a given radiation dose :

$$S(d) = exp[\gamma_r(-\alpha OMF\ d - \beta(OMF\ d)^2)] \tag{4.2}$$

where d is the radiation dose applied to a cell, OMF is the oxygen-modified factor increasing the sensitivity of a cell when it is more oxygenated [25] and γ_r, a factor reflecting the radiosensitivity (varying across the cell cycle) of a given cell. Depending on the value of S obtained by the irradiation of a given cell by dose d, whether the cell survives or not. Of course, for a dose of zero, the survival fraction must be equal to 1.

4.1.3 Nutrients thresholds

O'Neil and al. [2] highlight the significant differences in the behavior of different cell lines, including their nutrient requirements, growth and division rates, and density. To establish a starting point for studying these differences, it is needed to build a baseline, in order to have accurate parameters that reflect the behavior of naturally occurring cell populations. In the paper of O'Neil and al. [2], the number of critical neighbors around a cell is taken to be eight, as it is used in [19].

In this cellular model, only two nutrients have been chosen to perform aerobic respiration [26] : **Oxygen** and **Glucose**. Aerobic respiration is the aerobic catabolism of nutrients to carbon dioxide (CO_2), water (H_2O), and energy (as ATP), and involves an electron transport system in which molecular oxygen is the final electron acceptor. The overall reaction is:

$$C_6H_{12}O_6 + 6O_3 \longrightarrow 6CO_2 + 6H_2O + 38ATP \tag{4.3}$$

Glucose ($C_6H_{12}O_6$) is oxidized to produce carbon dioxide (CO_2) and oxygen (O_2) is reduced to produce water (H_2O). The theoretical ATP yield from aerobic respiration is 36 to 38 ATP molecules per glucose molecule. Aerobic respiration takes place in mitochondria and the energy produced (ATP) is used for the chemical reaction present in the cell's function. [26]

Tumor hypoxia is common in malignant tumors and contributes to dysfunctional vascularization resulting in cell mobility and metastasis. Hypoxia also alters cancer cell metabolism and induces cell quiescence, leading to therapy resistance. [27]

At the beginning of the simulation, each healthy cell is randomly placed on a grid with a randomly assigned stage (with uniform probability) in the cell cycle and two efficiency factors : (1) oxygen consumption efficiency reflecting the oxygen consumption of the cell for DNA replication, (2) glucose absorption efficiency that reflects the glucose absorption of the cell useful in DNA replication also.

Glucose and Oxygen in the simulation model

Two average factors: **average oxygen consumption** and **average glucose absorption** are, respectively, the oxygen consumed per hour and the amount of glucose absorbed per hour for an individual cell. The average glucose absorption was taken to be $3.6 \times 10^{-9} mg/cell/hour$ for healthy cells and $5.4 \times 10^{-9} mg/cell/hour$ [2]. The oxygen average consumption is taken to be $2.16 \times 10^{-9} ml/cell/hour$ [1]. These two average values can vary a lot for different types of cells. Indeed, these values are general numbers that are plausible.

Each generated cell has a glucose absorption efficiency derived from the multiplication of the average glucose absorption by a random factor $\alpha \sim \mathcal{N}(1, 1/3)$ (similarly for the oxygen nutrient). Figure 4.4 shows the distribution of the glucose absorption coefficient factor. Each cell has its own glucose absorption coefficient that is derived from this distribution. In this way, the glucose absorption distribution follows $\mathcal{N}(3.6 \times 10^{-9}, 1.2 \times 10^{-9})$ [mg/-cell/hour] for healthy cells and $\mathcal{N}(5.4 \times 10^{-9}, 1.2 \times 10^{-9})$ [mg/cell/hour] for cancer cells. This distribution has the purpose to reflect real life where each cell has different needs based on its genes.

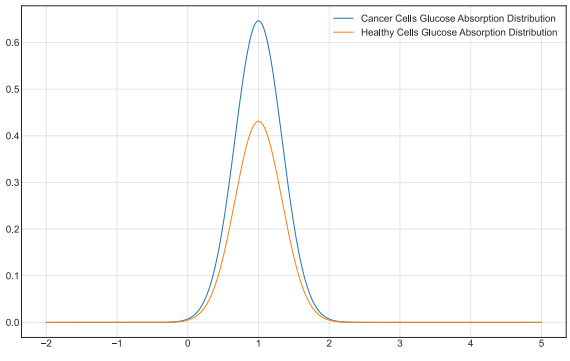


Figure 4.4: Glucose absorption distribution.

The other nutrient parameters are derived from the average parameters. The **quiescence glucose level** is the average amount of glucose required for two cells to complete an entire cell cycle. Thus : $2 \times 3.6 \times 10^{-9} \times 24 = 1.728 \times 10^{-7}\ mg/cell$. The **critical glucose level** (the glucose level where cells will die if the surrounding amount of glucose falls below) is taken as three-quarters the amount of the glucose required for an entire cell cycle. Indeed, cells in quiescence use less glucose than when they are actively dividing but they still require enough glucose to maintain regular metabolic functions. Thus : $3/4 \times 3.6 \times 10^{-9} \times 24 = 6.48 \times 10^{-8}\ mg/cell$. The starting glucose value is taken as the amount for which a healthy cell population would maintain quiescence due to cell density (glucose should remain above the quiescent glucose level everywhere on the grid) and if some cells were killed, the surrounding cells would re-enter the cell cycle to replace them. Thus, the initial value is set to $10^{-6}\ mg$ with a replinishing rate of $1.3 \times 10^{-6}\ mg/cell$ [2]. In the same way, the oxygen quiescence level is $2 \times 2.16^{-9} \times 24 = 10.37 \times 10^{-8}\ ml/cell$, the critical oxygen level is $3/4 \times 2.16 \times 10^{-9} \times 24 = 3.88 \times 10^{-8}\ ml/cell$.

In Gap 1 step, two independent conditions can lead to a Quiescent state (G_0) : (1) if the nutrient levels around the cell are below the critical amount defined previously, (2) if the number of neighboring cells is bigger than 9 (including the cell itself), as the figure 4.1 depicted earlier.

Parameters	Theoretical Values	Units	Initial Model Values	Modified Model Values
Starting healthy cells	1000	-	1000	1000
Starting cancer cells	1	-	1	1
Starting nutrient sources	100	-	100	100
Starting glucose level	1×10^{-6}	mg	100	100.0
Starting oxygen level	1×10^{-6}	ml	1000	1000.0
Average glucose absorption (healthy)	3.6×10^{-8}	mg/cell/hour	0.36	0.36
Average glucose absorption (cancer)	3.6×10^{-8}	mg/cell/hour	0.54	0.54
Average oxygen consumption (healthy)	2.16×10^{-8}	ml/cell/hour	20	20
Average oxygen consumption (cancer)	2.16×10^{-8}	ml/cell/hour	20	20
Critical oxygen level	3.88×10^{-8}	ml/cell	360	360
Critical glucose level	6.48×10^{-8}	mg/cell	6.48	6.48
Quiescent oxygen level	10.37×10^{-8}	ml/cell	960	960
Quiescent glucose level	1.728×10^{-8}	mg/cell	17.28	17.28

Table 4.1: Cellular model parameters.

Table 4.1 provides the information about the cellular model used for the simulation. There will always be, the **initial model values** that characterizes the model used by Grégoire Moreau and al. [19] and the modified

models that will be encountered later on. Table 4.1 is composed of the parameters calculated before and found in the literature [2] [1].

Nutrients diffusion

At the beginning of the simulation, some nutrient sources are placed randomly across the grid. At each time step, these sources give a defined amount of nutrients oon the pixel they are located. Nutrient sources are filled following table 4.1 parameters. Nutrients can also diffuse on the 2D grid and follow a differential equation :

$$c_j(t+1) = (1 - d_c)\ c_j(t) + \frac{1}{8} \sum_{k \in N(j)} d_c\ c_k(t) \tag{4.4}$$

where $c_j(t)$ is the concentration of nutrient on the patch j of the grid at time instant t. d_c is a diffusion rate and N(j) represents the neighborhood of the patch j. This is how every pixel on the grid can have its proper amount of nutrients.

4.1.4 TCP and NTCP

The concepts of TCP (Tumor Control Probability) and NTCP (Normal Tissue Complication Probability) [28] are two criteria used in radiation therapy. They can be used to optimize the treatment schedule of the radiation dose. These terms are commonly used in the field of medical physics to describe the likelihood of successful tumor control and the risk of radiation-induced harm to normal tissues, respectively.

- Tumor Control Probability (TCP): Probability that the tumor is destroyed after treatment. This criterion needs to be maximized.
- Normal Tissue Complication Probability (NTCP): Probability that tissues and organs close to the tumor suffer from side effects. This criterion needs to be minimized.

In this research, TCP is a crucial criterion used to assess the performance of treatment. At the same time, NTCP is not taken into account because of the absence of organs at risk in the model, making the estimation of side effects impossible.

To accurately interpret the value of TCP, one must understand that it is calculated as the ratio of successful simulations to the total number of conducted simulations. In this process, multiple simulations are carried out in the same environment, with the only randomness being the initial placement of glucose, oxygen, and cells on the grid.

A successful simulation is one in which the tumor is destroyed by the end of the simulation. For instance, a TCP value of 90% indicates that out of 100 simulations, 90 were successful, and 10 were unsuccessful. In this case, if a simulation with a previously observed TCP of 90% is conducted, it has a 9 out of 10 chance of succeeding.

It is crucial to note that the TCP value does not represent the percentage of cancer cells killed or any other metric, it solely reflects the likelihood of a simulation's success in eradicating the tumor.

4.1.5 Radiation

Radiation is included in the model by irradiating a given area with a certain dose. First of all, the tumor's current center of mass is calculated by averaging the positions of all cancer cells within the grid. Secondly, the tumor's radius will be determined by measuring the distance between the center of mass and the farthest cancer cell from this center.

Utilizing the identified center and radius, the radiotherapy dose will be distributed across each pixel on the grid according to a specific function. This function is derived from the convolution product of a rectangular window and a Gaussian distribution. The resulting dose deposition function, as illustrated in Figure 4.5, will be employed to find out the dose received by each cell based on its distance from the tumor's center of mass, which also serves as the irradiation center.

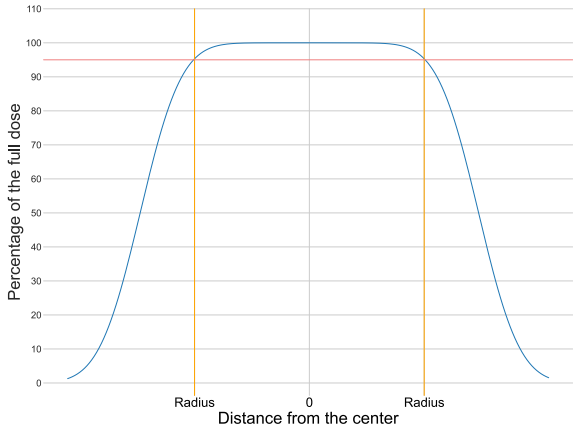


Figure 4.5: Dose deposition.

Due to radiotherapy treatment, healthy cells can suffer from collateral damage, such that they need to proceed to DNA reparation. Thus, each hour of simulation (each time step), cells can repair if they have suffered from any radiation. The function has been designed in order to reach 95% of the full dose if the distance is smaller than the determined radius.

4.2 Simulation Behavior

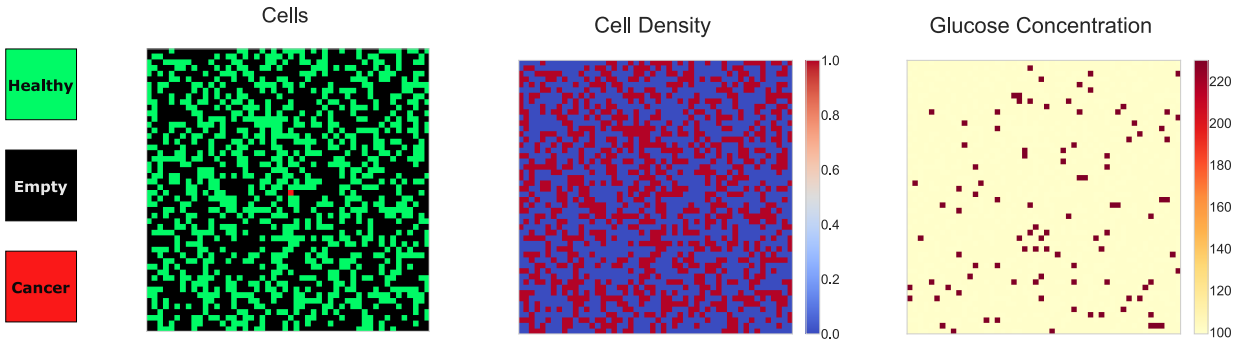


Figure 4.6: Initialization of the environment.

Figure 4.6 shows the initialization of the environment. One cancer cell is placed at the center of the grid and 1000 healthy cells are spread uniformly across the grid. On the left plot, healthy cells are represented in green, cancer cells are represented in red and empty spaces are represented in black. The cell density plot does not really provide more information at this stage of the simulation except that one can see where there are 2 cells on a single pixel. The glucose concentration plot gives an overview of the randomness in the placement of glucose on the grid.

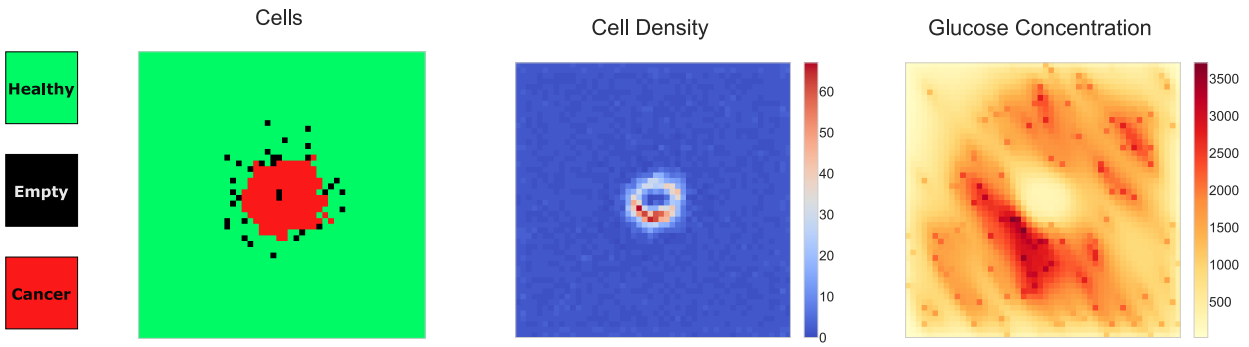


Figure 4.7: Environment before radiotherapy starts.

Figure 4.7 shows the environment before the radiotherapy begins. Indeed, the environment is initialized, as mentioned before, and evolves until the time step 350. From this point, the radiotherapy will begin where a dose is given each 24 hours of simulation. The cancer cells have started to spread across the grid. At this stage, the tumor is made of thousands of cancer cells (about 9000 cancer cells). As can be seen from the glucose plot, the location of the tumor creates a hole where there is less glucose [1].

4.2.1 Observations from environment

As it is mentioned later on, in section 4.3, Moreau and al. [19] use the complete 2D representation of the grid in a multi-layer form including the type of cells, the cell density, and the oxygen/glucose concentration, as observations of the environment. This is mainly because of the use of deep learning models that can take a large amount of data and perform pattern extraction with these raw unshaped data. In this work, reinforcement algorithms do not use deep learning models, such that we used to call them tabular algorithms (because these algorithms store the value function approximation in an array). Thus, the observations are limited to the number of healthy cells and cancer cells alive on the grid. Section 4.3.2 describes how these observations are quantized in order to fit the requirements of the tabular RL algorithms.

4.2.2 Termination conditions

Three conditions lead to the termination of the simulation :

1. All cancer cells are dead due to a lack of nutrients or radiation. If only one cancer cell remains, it can lead to the formation of a new tumor.

2. Radiation therapy is unsuccessful such that it leads to a growing tumor. It corresponds, in the simulation, to cancerous cells near the edges of the 2D grid. The implemented condition is that if the number of healthy cells is below 10, then the simulation is ended and the treatment has failed.

3. If the treatment has a too long duration. This happens if the beginning of the treatment is not aggressive enough, such that neither healthy cells nor cancerous cells are spreading or getting reduced. This condition is implemented, in the simulation, as a limit of time of 1200 hours (50 days of treatment, without including the first 350 hours of simulation)

[1] For the rest of this master thesis, only glucose plots will be shown but remind that the oxygen plot is exactly the same to a proportional factor.

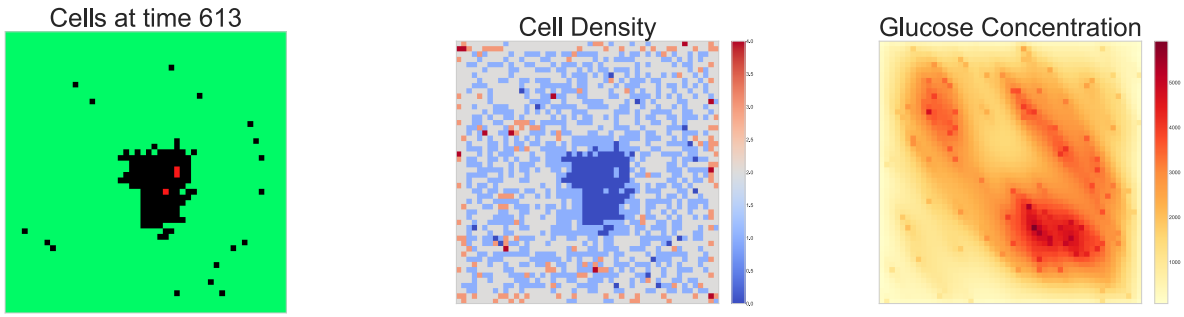


Figure 4.8: Termination example: Successful treatment.

Figure 4.8 shows the first termination condition. The radiotherapy treatment ended up in the death of every single cancer cell after almost the entire allowed time (On Figure 4.8, 3 cancer are left but die the next radiation dose). Obviously, this is the desired termination where the cancer cells cannot proliferate anymore.

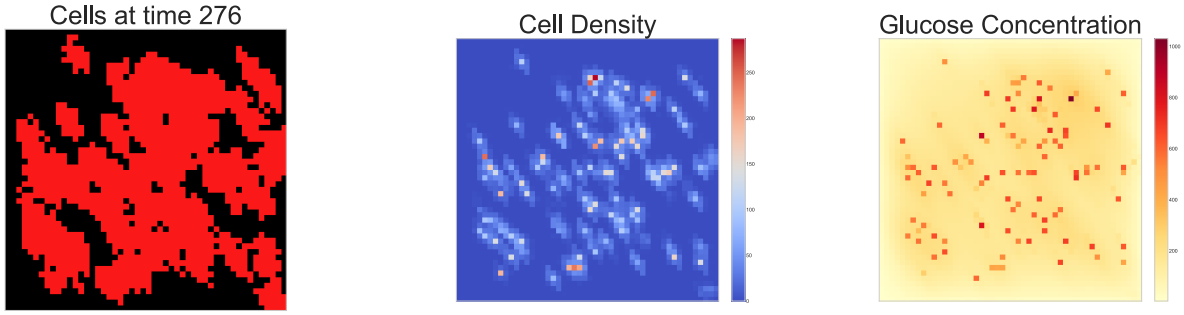


Figure 4.9: Termination example: Cancer Spread.

Figure 4.9 shows the second termination condition where the cancer cells have spread across the entire grid. One can see that the center of the tumor is necrotic such that the lack of nutrients starves cancer cells to death. The glucose concentration is very low across the entire grid. Of course, this is the undesired simulation termination.

Figure 4.10 illustrates the dynamic behavior of healthy and cancerous cells in an example simulation when the second termination condition is met. This termination condition occurs when cancer cells grow at a rate that outpaces their elimination by radiotherapy, leading to the tumor spreading across the entire grid. As a result, the figure displays an oscillating pattern that emerges due to the complex interplay of growth, death, and treatment factors.

The oscillating phenomenon can be explained as follows: As the tumor expands, cancer cells at the periphery continue to proliferate, while those in the central region begin to suffer from nutrient deprivation. The lack of nutrients is caused by the increased competition for limited resources among the rapidly multiplying cancer cells. Consequently, the cells in the center of the tumor mass start to die off.

This localized cell death, in turn, temporarily reduces the overall cancer cell population, causing a dip in the oscillating curve. However, as the central region becomes less congested, the remaining cancer cells once again have access to more nutrients, which allows them to resume their rapid proliferation. This growth pushes the curve upward, and the cycle repeats itself.

.

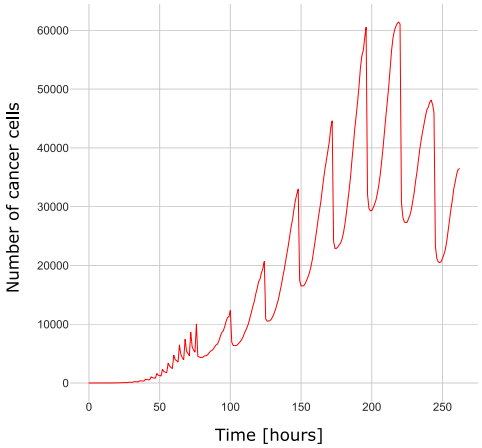


Figure 4.10: Termination example: Cancer Spread. Cancer cell number evolution over a period of 262 hours.

4.2.3 Baseline Simulation

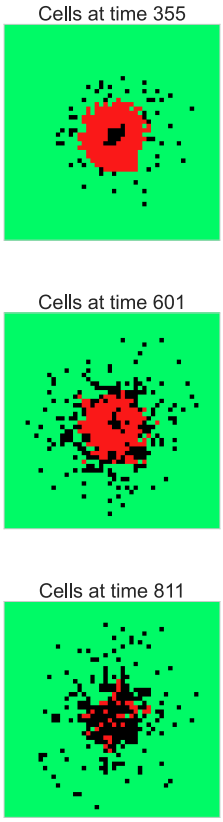


Figure 4.11: Baseline simulation example.

The baseline simulation in radiotherapy is crucial in determining the effectiveness of the treatment in controlling the growth of cancer cells. As mentioned before, the first 350 hours are dedicated to cell proliferation, after which radiotherapy doses are applied every 24 hours. The baseline treatment involves a dose of 2 Grays every 24 hours for 35 consecutive days. The default cellular parameters used in the simulation can be found in the table 4.1.

Figure 4.11 provides a visual representation of the baseline simulation over time. It can be seen that after 350 hours of cell proliferation, a tumor of 8 thousand cancer cells had formed. However, once the radiotherapy treatment started, the tumor had reduced in size, with the number of cancer cells dropping to 500. The last 200 hours of the simulation show that the tumor continued to reduce in size and was almost completely dead by the end of the treatment. Although it is not explicitly shown in the figure, the baseline treatment was successful in killing all the cancer cells.

Table 4.2 summarizes the performance of the baseline treatment through four key indicators: Tumor Control Probability (TCP), the total dose given to the patient, the number of fractions the treatment is composed of, and the total duration of the treatment in hours. To provide a more comprehensive understanding of the treatment's performance, several simulations were conducted, and the mean and standard deviation of the results were computed. The results indicate that, on average, the entire dose of 70 Grays is given to the patient, with 35 fractions of 2 Grays each. However, it is important to note that the baseline treatment is not always 100% successful, as reflected by the TCP not being 100%.

	TCP (%)	**Dose (Gy)**	**Fractions (-)**	**Duration (h)**	Survival (%)
Baseline	**99**	70.1 ± 10.89	35.1 ± 5.45	841.4 ± 130.70	99.9 ± 1.90

Table 4.2: Baseline performances

4.3 Reinforcement learning algorithms

Moreau et al. [19] employed deep reinforcement learning to explore the possibility of enhancing the performance of radiotherapy treatments in terms of duration, total dose, or total fractions smaller than those used in conventional radiotherapy. The primary advantage of utilizing deep reinforcement learning lies in the incorporation of deep neural networks. Indeed, the last decade has witnessed significant breakthroughs in the field of neural networks, driven by the development of powerful algorithms.

In the context of a 2D cellular model, employing a Convolutional Neural Network (CNN) is an astute approach to identify patterns within the 2D grid simulation. Moreau et al. implemented this strategy in their research, focusing on the three essential layers of 2D grid simulation (see figure 4.12): the cells' location, glucose distribution, and oxygen concentration.

Convolutional neural networks excel at identifying local patterns and spatial hierarchies within grid-based data, making them particularly suited for the task. By leveraging the power of CNNs, Moreau et al. were able to effectively analyze and understand the complex relationships between the various layers of the 2D

grid simulation. Their findings contribute to the ongoing quest for more precise, efficient, and personalized radiotherapy treatment plans that minimize side effects while maximizing therapeutic outcomes.

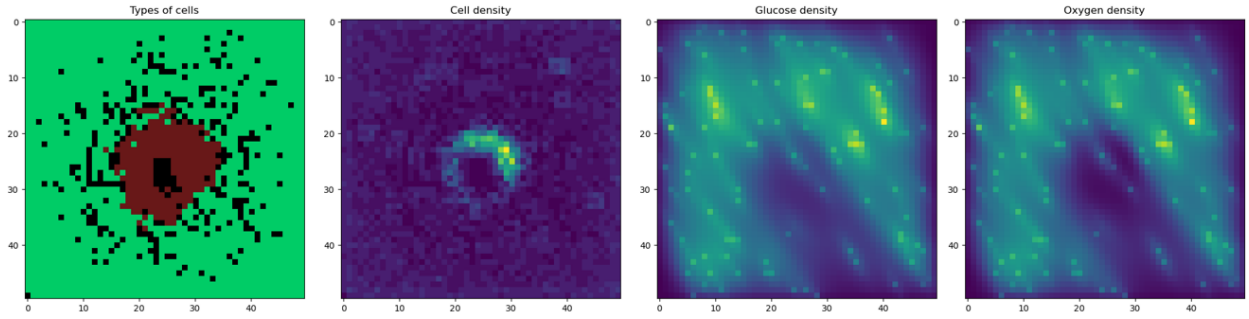


Figure 4.12: Input features of the convolutional neural network from Moreau et al. [19]

4.3.1 Reward function

Moreau and al. [19] describe how to calculate, for a given state, the reward to provide to an agent in order to achieve two goals: destroying the tumor and sparing the surrounding tissues. Two types of rewards are computed, one for non-terminal states and one for terminal states. The reward function should be representative of the objectives of an agent. In this way, for non-terminal states, the reward **R** is computed as :

$$R_t = -\frac{dose}{400} + \frac{(\#\ killed\ cancer\ cells - 5 \times \#\ killed\ healthy\ cells)}{100000} \tag{4.5}$$

A term including the amount of dose is in the expression to represent the objective of minimizing the total amount of dose provided to the patient. The factor 400 is also determined empirically.

Factor 5 gives more importance to the number of healthy cells killed such that the agent will avoid using too aggressive treatment. Factor 100 000 leads to rewards laying into the interval -1 and 1. These factors have been determined experimentally. In terminal state, the reward **R** is computed as :

$$R_T = -\frac{dose}{400} + 0.5 - \frac{(Init_{hcells} - End_{hcells})}{3000} \tag{4.6}$$

where $Init_{hcells}$ represents the number of healthy cells after initializing the environment (placing 1000 healthy cells, 1 cancer cell, and **running the simulation for 350 hours**) and End_{hcells} represents the number of healthy cells at the end of the simulation. The two factors 0.5 and 3000 have been determined empirically such that it guarantees to obtain a reward between 0 and 1.

4.3.2 Observation space quantization

Tabular RL agents require the quantization of the observation space for effective learning. In the case of radiotherapy simulations, the observations for a given state consist of the number of healthy cells and cancer cells present around the tumor on the grid. Discretizing the observation space accurately is essential for the RL agent to make informed decisions during the treatment process.

To establish lower and upper bounds for each cell type, **multiple simulations are performed using different constant levels of radiotherapy, meaning that for an entire simulation, every 24 hours, the same dose is given to the patient. It is done for multiple simulations, each with a different constant radiation dose**. The lower bound for cancer cells is set at 0, as the ultimate goal of radiotherapy is to achieve a TCP of 100%.

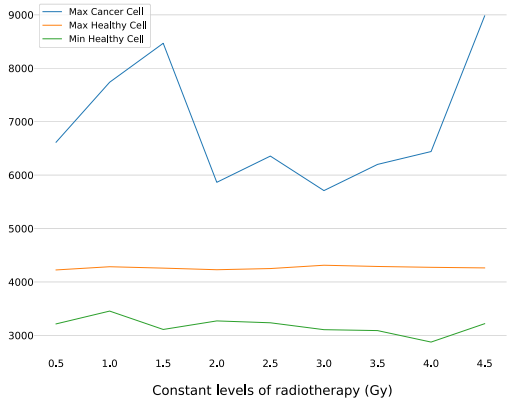


Figure 4.13: Number of min/max healthy and cancer cells to determine thresholds.

Notice that the thresholds will be determined on the part of the simulation where the treatment is administered, meaning that we ignore data before the radiotherapy treatment commences since agents can act only at this moment.

The cellular model parameters for these simulations are outlined in Table 4.1. The constant dose varies for each new simulation, ranging from 0.5 Gy to 4.5 Gy, with increments of 0.5 Gy (it is chosen arbitrarily). This range is chosen to align with the agent's limitations, which are restricted to selecting integer values between 1 Gy and 4 Gy for the dose.

By conducting these simulations, the upper bound for cancer cells and the corresponding thresholds for healthy cells can be accurately determined. This enables effective quantization of the observation space, allowing the Tabular RL agent to operate within these bounds and make informed decisions during radiotherapy treatment simulations.

Figure 4.13 showcases the minimum and the maximum number of healthy and cancer cells for different constant radiotherapy doses. Interestingly, the maximum number of healthy cells remains fairly constant, leading to the selection of an upper bound of 4300. The lower bound is chosen as the lowest value observed: 2875. To quantize the healthy cells, the interval [2875, 4300] is divided into five subintervals. We take an arbitrary subdivision in 5 intervals because it was the choice of Moreau and al. in their work [19].

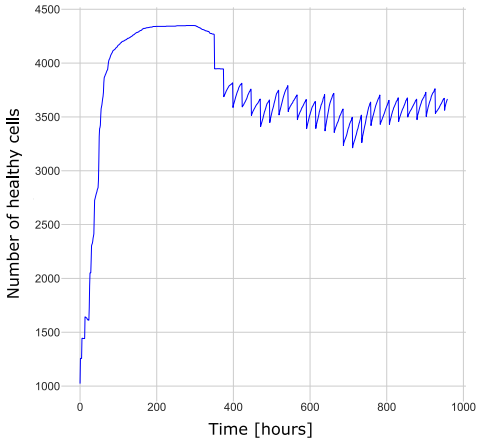


Figure 4.14: Example of healthy cells number evolution over a period of 958 hours.

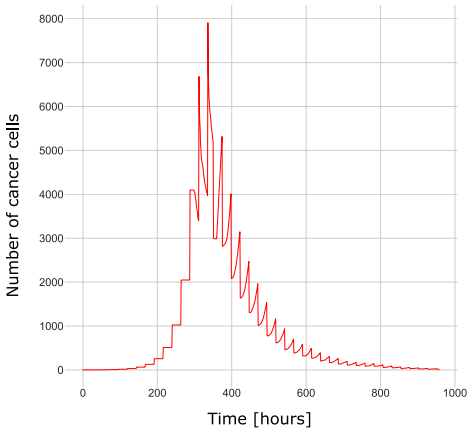
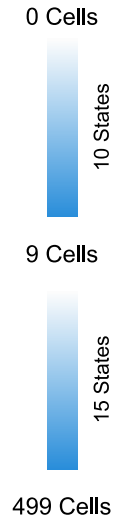


Figure 4.15: Example of cancer cells number evolution over a period of 958 hours.

Figure 4.14 and figure 4.15 present the resulting curve depicting the number of healthy and cancer cells at each time step for a representative simulation. In this example, the simulation monitors the growth and decline of both cell populations, with the data being stored for further analysis. The curve illustrates that the high threshold for healthy cells is approximately 4300, while the low threshold hovers around 2875. This indicates that the population of healthy cells fluctuates within this range throughout the simulation.

For cancer cells, the upper bound of the population is higher than the maximum value observed in this particular simulation. This implies that the cancer cell population has the potential to grow even larger under different conditions or circumstances. It is crucial to consider that the growth dynamics of cancer cells may vary depending on factors such as the aggressiveness of the cancer, the presence of treatment interventions, or the individual patient's biological characteristics.



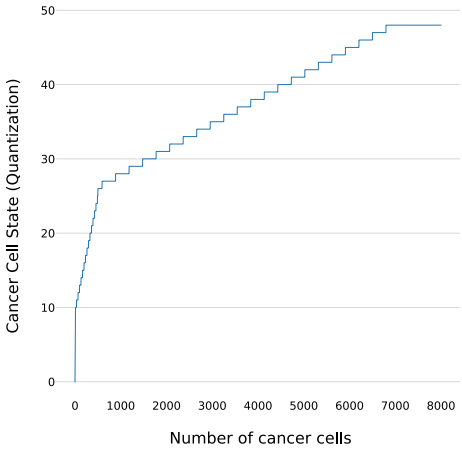


Figure 4.16: Quantization of the cancer cells observation.

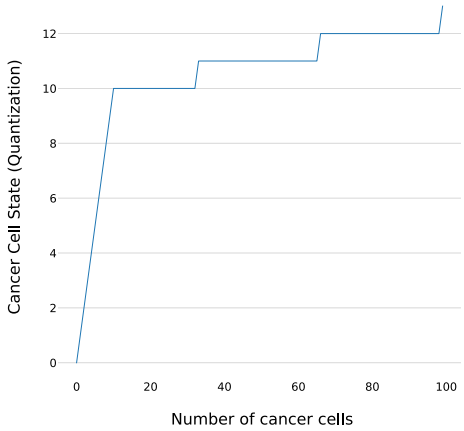


Figure 4.17: Quantization of the cancer cells observation (Zoom).

For cancer cells, the upper bound is selected to be 9000. Figure 4.16 and figure 4.17 illustrate the quantization of cancer cells. It is important to note that the overall shape of the quantization tends to resemble a square root function. This shape is chosen empirically based on the observations from the simulations.

Intuitively, a linear quantization would not be ideal for optimizing performance, as not all states hold equal significance. In fact, a high number of cancer cells is a more common occurrence, which means that the RL agent is more likely to encounter these states. On the other hand, a low number of cancer cells typically only happens towards the end of the simulation if the agent has been successful in its treatment strategy. Therefore, a greater emphasis should be placed on states with a lower number of cancer cells in order to optimally train the agent.

The cancer cells are quantized into 50 distinct states, with the details of the quantization displayed on the left of the page, indicating the number of states between two given cell counts. For instance, there are 10 states between 0 and 9 cells.

Chapter 5

Methods

This section is devoted to detailing the methodology employed in analyzing the performance of our RL agent within various cellular environments. Our primary objective is to scrutinize the robustness of our RL agent across a broad spectrum of environments.

To achieve this, our initial step entails selecting a suitable RL agent for testing within these environments. This selection process requires us to establish and apply a well-defined set of criteria.

Subsequently, we delineate the approach we employ to assess the robustness of the selected RL agent and outline the method of comparison with other agents within the same parameters. Our ultimate aim is to provide a comprehensive analysis that substantiates the reliability and adaptability of our RL agent across diverse cellular environments.

5.1 Agent Evaluation

5.1.1 Agent's learning behavior

The primary goal of this section is to provide an in-depth analysis of the learning behavior of RL agents, focusing on three types of tabular algorithms: Q-learning, Sarsa, and Expected Sarsa. By examining various performance indicators, we can determine what constitutes a "good" agent, and explore methods to reduce training time, such as early stopping. This analysis will help understand the learning process of each algorithm and offer insights into their relative strengths and weaknesses.

The training process consists of several steps:

1. Training the agent using its specific update algorithm: Q-learning, Sarsa, or Expected Sarsa. This is conducted across 2000 training epochs, meaning that each algorithm is executed 2000 times.
2. Testing the agent across multiple environments: During each testing epoch, a new environment is created, and the agent is evaluated using its q-table. There are 50 tests for each testing phase every 2000 training epochs, during which the agent's performance is assessed.
3. The two previous points are repeated 20 times for each type of algorithm.

Throughout the analysis, plots will be presented that feature numerous agents, categorized as Q-learning, Sarsa, or Expected Sarsa. It is crucial to note that although these plots display multiple agents, there are only three distinct types of agents, each representing a different learning algorithm. Each type of agent is trained for 20 training epochs, which means that we will observe results for 60 agents. The 20 agents that belong to a class of algorithm (i.e. Q-learning, Sarsa or Expected Sarsa) are the same agent that is more and more trained. It will be impossible to know which agent is at which training step because this is not the purpose of this section. This section is dedicated to highlighting relationships between different performance indicators.

In order to evaluate the performance of these algorithms, several simulations have been conducted, with each agent's performance assessed through four key indicators: Tumor Control Probability (TCP), the total dose delivered to the patient, the number of fractions comprising the treatment, and the total duration of the treatment in hours. The mean and standard deviation of the results are calculated for each performance indicator, providing a comprehensive understanding of each agent's performance across the simulations. All simulations utilize the default parameters listed in Table 4.1.

By analyzing the results of these simulations, we can draw meaningful conclusions about the learning behavior of each RL agent, as well as compare and contrast their performance in various aspects.

Figure 5.1 illustrates the relationship between the number of doses and the treatment duration for various tabular agents [1]. It is important to note that only agents with a 100% TCP have been included in this plot. Remarkably, out of the 60 agents examined, only 2 failed to achieve this benchmark. These two agents are the two first ones of the Q-learning training process, such that they did not have learned enough yet.

Upon examining the graph, a clear linear relationship emerges between the number of doses and the treatment duration for each type of agent. The data also reveals that the vast majority of agents require no more than 700 hours to eradicate all cancer cells, highlighting the potential for effective treatments within a relatively short time frame.

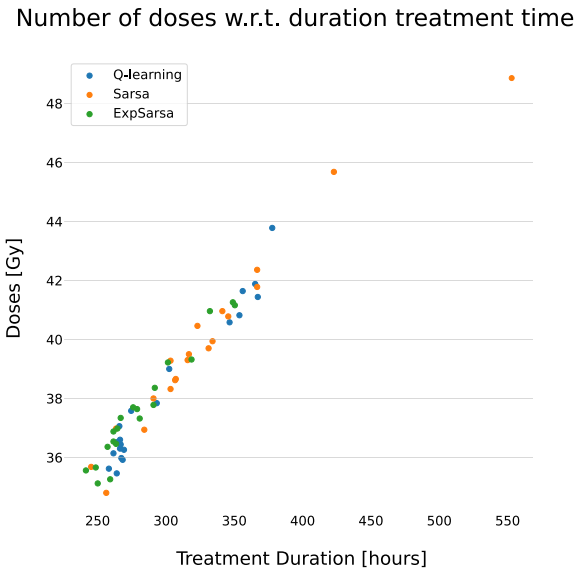


Figure 5.1: Number of doses with regards to treatment duration for different Tabular Agents.

It is essential to consider, however, that an agent with a low dose or short treatment duration is not necessarily indicative of a superior agent. As noted by Jalalimanesh et al. [1], additional factors beyond these parameters should be taken into account when evaluating the effectiveness of different agents. For instance, potential side effects, drug interactions, and patient-specific factors such as age, overall health, and the presence of comorbidities may all influence the suitability of a given agent [29] [30]. Furthermore, the impact on the patient's quality of life during and after treatment should also be considered, as a shorter treatment duration or lower dosage may not always translate into improved patient outcomes. However, in this study, these factors are not taken into account and then we need simpler criteria to choose an optimal agent.

Establishing a clear definition of what constitutes a good agent is crucial for selecting candidates to evaluate their robustness in subsequent tests. A good agent must consistently achieve a 100% TCP while also maintaining a survival percentage comparable to that of the baseline treatment. It is important to recognize that some agents may yield treatments with survival percentages lower than those provided by the baseline treatment. This outcome can primarily be attributed to the agents' ability to administer radiation doses exceeding 2 Gy to both cancerous and healthy cells, which can result in increased mortality within the healthy cell population.

Therefore, while aiming for a 100% TCP and comparable survival percentages to the baseline treatment are essential criteria for defining a good agent, it is important to keep in mind that these objectives should be weighed against potential side effects and the overall impact on patient outcomes.

[1] Remind that there are only three distinct agents (see 5.1.1).

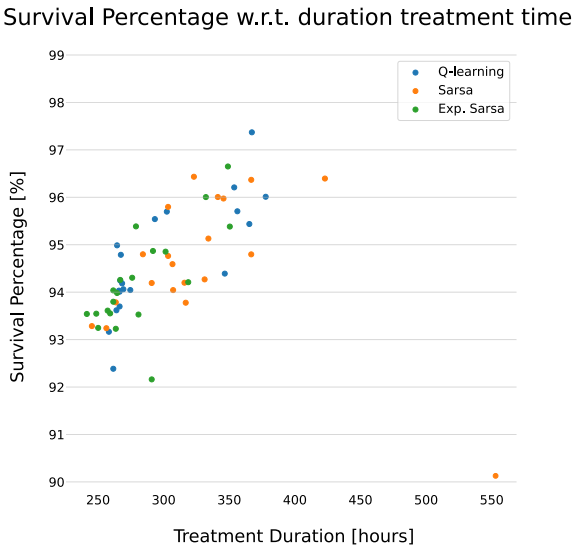


Figure 5.2: Survival Percentage of Healthy Cells with regards to treatment duration for different Tabular Agents.

Figure 5.2 presents the survival percentage of healthy cells from the beginning (corresponding to the initiation of radiotherapy) to the conclusion of the simulation, plotted against the treatment duration. The time taken for agents [2]to reach the end of the simulation varies significantly, depending on the decisions they make concerning dosage, as previously observed in Figure 5.1. As mentioned earlier, a shorter treatment duration does not necessarily imply a superior treatment outcome.

All treatments administered by the agents in this study achieve a survival percentage of over 92% for healthy cells, which is acceptable compared to the baseline survival percentage of 98%. This finding highlights that the agents can deliver treatments with relatively high survival rates for healthy cells while still addressing the primary goal of achieving a 100% TCP.

Figure 5.3 illustrates the relationship between the number of unexplored states in the state space and the training epochs for more and more trained agents. As the training progresses, the number of unexplored states demonstrates a monotonic decrease until it reaches a plateau. This plateau signifies that the agents are no longer able to discover new states, as the environment never encounters those particular states. This is also due to the fact that some states have been designed for further explorations, for example, during the re-training process to render agents more robust.

Selecting an agent for robustness testing should take into account the number of states it has already explored. An agent with more experience in different situations is likely to exhibit greater robustness in the face of new challenges or unexpected inputs. By having encountered a diverse range of states, the agent has had the opportunity to refine its decision-making processes and strategies, leading to a better understanding of the environment and its dynamics.

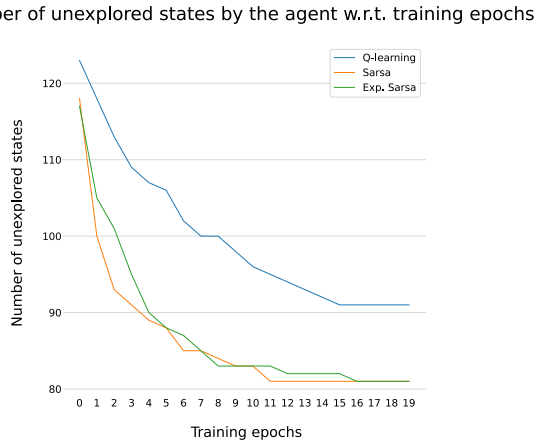


Figure 5.3: Number of unexplored states by the agents w.r.t. the training epochs.

In summary, the RL agents, such as Q-learning, Sarsa, and Expected Sarsa, demonstrate impressive efficiency in meeting the two primary criteria for selection for robustness tests: achieving a TCP of 100% and maintaining a survival percentage of healthy cells close to the baseline. When assessing an agent's robustness, it is advantageous to prioritize those that have already explored a larger number of states in the state space, as they are likely to provide increased adaptability in diverse environments. This performance showcases the potential of these agents in optimizing radiotherapy treatment plans while preserving the integrity of surrounding healthy tissues.

The accomplishment of these criteria allows researchers and practitioners to select RL agents with greater confidence and flexibility for further testing and evaluation. By meeting these benchmarks, the selected

[2]Remind that there are only three distinct agents (see 5.1.1).

agents have proven their ability to effectively balance tumor eradication with the preservation of healthy cells. This makes them ideal candidates for more extensive testing of their robustness and generalizability in a variety of clinical settings and treatment scenarios.

Learning rate and epsilon Decay

The learning rate is a crucial hyperparameter in RL algorithms determining the magnitude of updates applied to the Q-values during the learning process. In the context of Q-Learning, a higher learning rate (denoted as α) signifies larger updates to the Q-values at each step. However, as the learning progresses, it is necessary to reduce the learning rate gradually in order to stabilize the model output and facilitate the convergence to an optimal policy.

Epsilon (ϵ) is another significant parameter employed in action selection based on the current Q-values. For instance, adopting a purely greedy method ($\epsilon = 0$) would entail consistently selecting the action with the highest Q-value for a given state. This approach, however, can lead to inadequate exploration and result in the agent becoming trapped in local optima.

In summary, the learning rate is associated with the magnitude of updates to Q-values, while epsilon is related to the degree of randomness in action selection. As the learning process advances, both parameters should be decayed to stabilize the model and facilitate the exploitation of the learned policy, ultimately converging to an optimal solution.

For our purpose, the value of α and ϵ have been determined by Moreau and al. [19] experimentally. Both, α and ϵ are decreased linearly during the training process. α starts at a value of 0.8 and ends at 0.5, while ϵ goes from 0.8 to 0.05.

5.1.2 Conclusion

In conclusion, an agent must be selected to perform robustness tests against various environments, as it will be discussed in Chapter 6. To choose an appropriate agent, several key performance indicators must be met to a certain extent:

1. The chosen agent must achieve a 100% TCP.
2. The agent must maintain a survival percentage of healthy cells comparable to the baseline level (over 90%). This is the closest criterion to an NTCP estimation.
3. The agent should exhibit superior performance in terms of the number of fractions, amount of doses administered to the patient, and treatment duration, compared to its counterparts.
4. The chosen agent should explore a greater number of states than its counterparts, where states refer to those described in Section 4.3.2.

It is important to note that this work does not consider the side effects of treatment; therefore, it is not used as a criterion for agent selection.

As mentioned before, the selected agent will be assessed in numerous and distinct environments. Specifically, we assess how the agent copes with changes in nutrient consumption, cell cycle duration, and radiosensitivities for each stage of the cell cycle. These factors are crucial in determining the agents' ability to effectively control tumor growth and adapt to different tumor environments. By analyzing the agents' performance under these diverse conditions, we can identify their limitations and areas for improvement.

To maintain consistency and better reflect real-world scenarios, the initiation of radiotherapy is no longer based on a fixed time after 350 hours of simulation. Instead, we employ a volumetric criterion for the start of radiotherapy. This approach is particularly useful when dealing with an aggressive environment, in which cancer cells proliferate and spread across the grid much more rapidly than in normal conditions. In such cases, if we were to wait for 350 hours before starting radiotherapy treatment, the tumor would be significantly larger than it should be at the time of detection. The tumor should ideally be detected and treated when

it reaches a certain size, and thus, the volumetric criterion states that radiotherapy will commence once the number of cancer cells exceeds 9000 (corresponding to the upper bound of the quantization of the observations space). From this starting point, the treatment proceeds as usual, with one dose administered every 24 hours. However, if this criterion is not reached before 350 hours, we start as usual the radiotherapy treatment [3].

In the next section, the term "selected agent" is used to refer to the agent that will be rendered robust against some types of environments. The selected agent should be the agent corresponding to the criteria described in this section.

5.2 Nutrients Consumption Changes

In this section, we aim to outline the methodological approach we intend to utilize to explore the impact of nutrient consumption on radiotherapy treatments. This investigation will focus specifically on modulating the average glucose and oxygen consumption rates of both healthy and cancerous cells. Consequently, we will assess how these adjustments influence cellular evolution with respect to critical and quiescent levels during the simulations.

As a reminder, in section 4.1.3, the average glucose absorption was taken to be $3.6 \times 10^{-9} mg/cell/hour$ for healthy cells and $5.4 \times 10^{-9} mg/cell/hour$ [2]. The oxygen average consumption is taken to be $2.16 \times 10^{-9} ml/cell/hour$ [1]. As one can notice in the benchmark environment (see Table 4.1), there is only one critical glucose (oxygen) level for both healthy and cancer cells. Consequently, healthy and cancer cells die under the same critical level. This level is calculated based on the average glucose (oxygen) consumption of **healthy cells**. This means that even if the average nutrient consumption of **cancer cells** increases, the critical level at which the cancer cells die, remains the same.

Our methodology primarily involves manipulating the average nutrient consumption of cancer cells. In Section 6.2, we will examine two scenarios: one where we increase the average nutrient consumption of cancer cells, and another where we decrease it. This process, as detailed in this section, should provide insights into the influence exerted by the critical nutrient levels on cancer cell behaviors.

Figure 5.4 provides a visual depiction of the three primary processes that are updated during each timestep of the simulation. It is composed of refiling the sources on the grid, performing the cell cycle update for every cell present on the grid, and the diffusion of nutrients across the grid.

The initial phase of this process involves the replenishment of nutrient sources. Each nutrient source is restocked with a predetermined amount of nutrients.

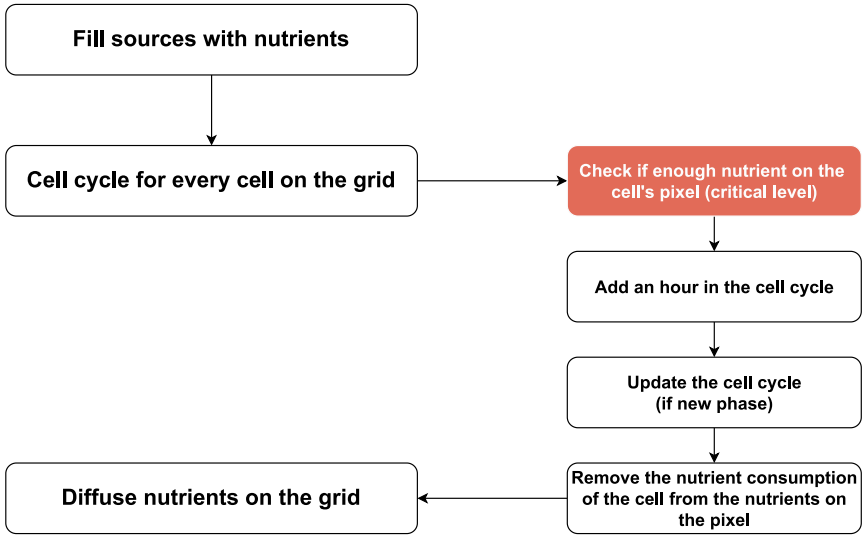


Figure 5.4: Nutrients modifications at each time step of simulation.

Next, the cell cycle is executed for every cell occupying the grid. During this phase, the simulation assesses whether the nutrient level of a particular pixel surpasses the established critical level. If the nutrient level in a pixel drops below this critical level, all cells, irrespective of being cancerous or healthy, inhabiting that

[3] In the next section, we will see a simulation with a high nutrient consumption rate that reaches only about 500 cancer cells after 350 hours but radiotherapy starts at this time.

pixel are deemed unviable and thus, they perish. This scenario is denoted by the red portion on Figure 5.4. In the final phase of the cell cycle update, the simulation subtracts the average nutrient consumption of a cell from the total nutrient content present in the cell's pixel.

Our methodology primarily involves manipulating the average nutrient consumption of cancer cells. In Section 6.2, we will examine two scenarios: one where we increase the average nutrient consumption of cancer cells, and another where we decrease it. This process, as detailed in the said section, should provide insights into the influence exerted by the critical nutrient levels on cancer cell behaviors.

Limitations

It's crucial to note that we cannot arbitrarily increase the nutrient consumption rate in our tests. There are inherent risks if the nutrient consumption rate rises too much, as such scenarios are not accommodated in the cellular model. Specifically, the critical nutrient level is calculated as the average nutrient consumption of healthy cells multiplied by 18 (equivalent to $24 \times \frac{3}{4}$, as detailed in section 4.1.3). It's imperative that the average nutrient consumption of cancer cells does not surpass this threshold, being 18 times the average nutrient consumption of healthy cells.

If such a situation arises, we could confront a circumstance where the nutrient quantity on a given pixel might drop below zero. This could happen if the nutrient amount on a pixel is slightly above the critical nutrient level, and the cell's nutrient consumption is marginally above the remaining nutrients present on the pixel. The simulation would proceed as follows: the cell progresses in the cell cycle by consuming the nutrients present on the pixel, then deducts the nutrient consumption from the pixel's nutrient quantity. Given our assumptions, the updated nutrient value on this pixel would be less than zero, a scenario that lacks coherence.

Considering this, we will restrict ourselves to a tenfold increase in the nutrient consumption rate. The initial average glucose absorption for healthy cells is set at $3.6 \times 10^{-9} mg/cell/hour$, while for cancer cells it's $5.4 \times 10^{-9} mg/cell/hour$. Thus, the initial average glucose absorption of cancer cells is 1.5 times higher than that of healthy cells. If we enhance the glucose absorption of cancer cells tenfold, it becomes 15 times higher than that of the healthy cells. This value remains below the previously mentioned 18-fold limit, and is therefore deemed safe for use in our context.

5.3 Cell Cycle Duration Changes

In this section, we will provide our approach to evaluating a RL agent faced to cell cycle duration changes. As we want to test the robustness of our RL agents, we describe how to evaluate an agent in environments with various cell cycle duration. We face two cases where we have train agents on environments with specific cell cycle duration.

First, we want to evaluate the performances of a selected agent (to test its robustness) that is re-trained on an environment with a specific cell cycle duration. We re-train the selected agent by following the training process as described in Section 5.1.1. Our approach involved training the selected agent on environments with cell cycle durations (T_c) of 16 hours, 18 hours, and 20 hours.

We started by taking the selected agent and training it on an environment with a 16-hour cell cycle duration. Next, we once again took the selected agent and trained it on an environment with an 18-hour cell cycle duration. It is important to note that the selected agent was not trained on multiple environments sequentially; instead, each training process was conducted independently. For instance, we did not use the agent trained in a 20-hour cell cycle environment to train it on the 18-hour cell cycle environment.

Secondly, we also want to compare these re-trained agents with agents that are directly trained on the corresponding environments. In this case, we simply follow the training process described in Section 5.1.1 with an environment that has a specific cell cycle duration.

5.4 Radiosensitivities Changes

This section has the purpose to describe the methodology to evaluate agents' robustness on environments with various sum of radiosensitivities. The radiosensitivity has been described in the Section 4.1.2. Equation 4.2 describes the survival probability of a cell to a given radiation dose. The factor γ_r reflects the radiosensitivity, varying across the cell cycle (even quiescence has a specific radiosensitivity).

In Section 4.1.2, the default values for radiosensitivities are depicted in Figure 5.5. It is crucial to remember that the agents can only administer a radiation dose of up to 4 Grays, ensuring that the survival probability drops below 30% for the Mitosis stage, for instance. The radiosensitivity for each stage can be modified to explore different scenarios.

The stage duration plays a significant role in determining the likelihood that a radiation dose will be administered at a specific stage. For example, the Mitosis phase is consistently the shortest phase, lasting approximately an hour. Consequently, it is less probable that the majority of cells will be in this stage when the radiation dose is administered. To account for this, a weighted average is calculated to represent the radiosensitivity across the entire cell cycle (excluding the quiescence state).

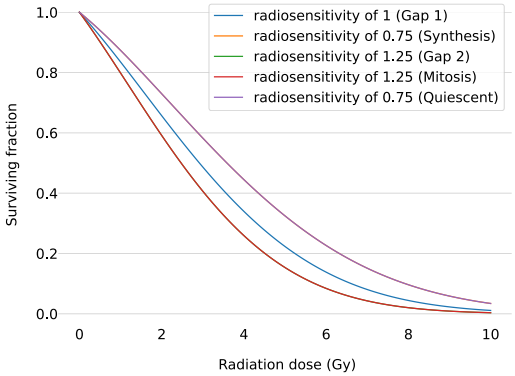


Figure 5.5: Cell survival curve for different radiosensitivities.

In the following sections and chapters, we will use the term radiosensitivity to refer to this weighted sum of radiosensitivities.

As in the previous section, we will assess the performance of a selected agent re-trained on environments with modified radiosensitivities. We use the selected agent and re-train it on a variety of environments with different radiosensitivities from the range 0.5 to 0.8.

We also want to compare these re-trained agents with agents that are directly trained on the corresponding environments. In this case, we simply follow the training process described in Section 5.1.1 with an environment that has a specific radiosensitivity.

To evaluate fairly, we keep the cell cycle length, denoted as T_c, constant at its default value of 24 hours. The duration of each stage in the cell cycle also remains fixed, following the cell cycle description provided in Section 4.1.1. As a result, the weighted average can be computed as the radiosensitivity at each stage, multiplied by the duration of that stage and divided by 24 hours.

Chapter 6

Results

This section presents the results of various experiments designed to evaluate the performance of the trained RL agents and their robustness under different conditions.

This comprehensive evaluation of the agent's performance and robustness provides valuable insights into the effectiveness of the trained RL agent in controlling tumor growth and offers a solid foundation for future improvements to the agent's adaptability and performance.

The results are organized into subsections, each focusing on a particular aspect of the agents' performance. The first subsection investigates the agent's performance in a benchmark environment, comparing it to a baseline treatment. The following subsections discuss how the agent's robustness is affected by variations in nutrient consumption, cell cycle duration, and radiosensitivities. Each subsection concludes with an analysis of the agent's performance, highlighting the implications of the observed trends and identifying possible avenues for enhancing the agent's robustness.

In the following sections, the reader will be confronted with the words "Key performance indicators". These represent the performances of the agent in terms of Tumor Control Probability (TCP), the treatment duration, the number of fractions administered, the amount of dose, or the survival rate of healthy cells (see Section 5.1.2).

6.1 RL Agent Selection

The primary objective is to select an agent from the trained candidates to conduct robustness tests and, if necessary, enhance its performance by re-training it on specific environments. As outlined in Section 5.1.2, several criteria must be considered to make an informed decision regarding the selection of the most suitable agent.

	TCP (%)	Dose (Gy)	Fractions	Duration (h)	Survival (%)
Baseline	**99**	70.1 ± 10.89	35.1 ± 5.45	841.4 ± 130.70	99.9 ± 1.90
Sarsa 17	**100**	41.8 ± 6.26	15.28 ± 2.76	366.7 ± 66.17	96.4 ± 2.50

Table 6.1: Comparison of performances between the baseline and the selected agent over 100 distinct simulations.

Table 6.1 compares the performances of the baseline treatment and the treatment provided by a trained RL agent. These performances have been evaluated on a benchmark environment (see Table 4.1). The results indicate that the selected RL agent outperforms the baseline treatment in every performance metric but is also among the ones that have the lowest number of unexplored states in the state space. Having identified an optimal agent, the next step is to test the robustness of the optimization provided by this agent under

different conditions, such as variations in nutrient consumption, cell cycle duration (T_c), and changes in the radiosensitivities for each stage of the cell cycle. This section will focus on assessing the robustness of the agent, while a subsequent section will be dedicated to improving the agent's robustness, if necessary.

For the following sections, the selected agent is now the Sarsa 17 agent. We will use this name to refer to the selected agent from this point.

6.2 Nutrients Consumption Changes

In this section, we will concentrate our attention on this particular aspect of the model: how the average nutrient consumption of cancer cells affects the survival of both healthy and cancer cells. Specifically, we will be manipulating the average consumption rate of nutrients by cancer cells and observing the resultant changes in cell survivability.

We will investigate if a change in nutrient consumption could lead to the development of tumors that are more difficult to eradicate. Specifically, alterations in glucose absorption rates, similar to changes in oxygen consumption, can greatly impact tumor behavior and growth (increasing or decreasing the cancer cell proliferation). Unlike healthy cells, cancer cells do not possess a quiescent mode, they either replicate or die.

In this section, we will explore two types of environments: one with high nutrient consumption for cancer cells (10 times more) and another with low nutrient consumption for cancer cells (10 times less) compared to the benchmark environment detailed in Table 4.1.

	TCP (%)	**Dose (Gy)**	**Fractions (-)**	**Duration (h)**	**Survival (%)**
Benchmark Environment	100.0	41.8 ± 6.26	15.3 ± 2.76	366.7 ± 66.2	96.4
Nutrient consumption x10	100.0	43.5 ± 6.43	16.3 ± 2.89	378.8 ± 68.4	96.2
Nutrient consumption ÷10	100.0	39.9 ± 6.39	15.1 ± 2.81	355.6 ± 64.8	96.1

Table 6.2: Results for simulations with a nutrient consumption rate multiplied by 10 and with a nutrient consumption rate divided by 10. These results are based on the selected Sarsa agent presented in Section 6.1.

Table 6.2 displays the outcomes of simulations wherein the nutrient consumption rate was increased tenfold and decreased to one-tenth of the original value. These simulations were juxtaposed with the standard environment, as outlined in Section 4.1.

A cursory review of the data reveals surprisingly similar results across these differing scenarios, with less than a 5% variation in all key performance indicators. The consistency extends to the standard deviation of these indicators, suggesting that the agent's performance remains unaffected by changes in the environmental conditions.

This uniformity implies that adjustments to the nutrient consumption rate may not significantly alter the environment's behavior. We hypothesize that the agent exhibits the same performance regardless of the nutrient consumption rate because the nature of the environment remains essentially unchanged. To confirm this hypothesis, we will provide an in-depth understanding of the simulation behavior by analyzing individually the two types of presented simulations.

High Nutrient Consumptions Increasing the nutrient consumption of cancer cells can lead to the development of a tumor with more extensive necrosis in its center than usual. This occurs because a significant number of cancer cells die off naturally due to the lack of nutrients. Figure 6.1 illustrates a tumor with necrosis at its core, where cancer cells are effectively starving to death because of the limited availability of nutrients. As mentioned before, the critical nutrient levels are determined by the nutrient consumption of healthy cells, and these levels remain unchanged even as the nutrient consumption of cancer cells increases. This results in the death of a large number of cancer cells. However, it is crucial to note that if the critical levels were based on the nutrient consumption of cancer cells, increasing nutrient consumption would lead to much higher critical levels, causing all cancer cells to die without the need for radiotherapy.

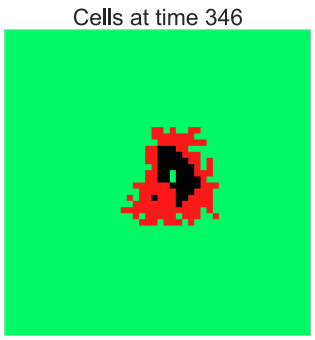


Figure 6.1: Tumor Necrosis on an increased nutrient consumption rate simulation.

Low Nutrient Consumptions In this scenario, when nutrient consumption is decreased, the tumor experiences a much higher concentration of cancer cells in its center. The nutrient availability is significantly greater than what the cells consume, implying that the primary factor limiting cellular replication in a specific area becomes the maximum capacity of neighboring cells that can coexist.

When radiation therapy is applied to such a tumor, the administered dose has the potential to impact a large number of cancer cells due to their high concentration in the targeted area. As a result, it becomes more feasible for the therapeutic agents to eliminate all cancer cells, as they are primarily concentrated within the area subjected to radiation treatment.

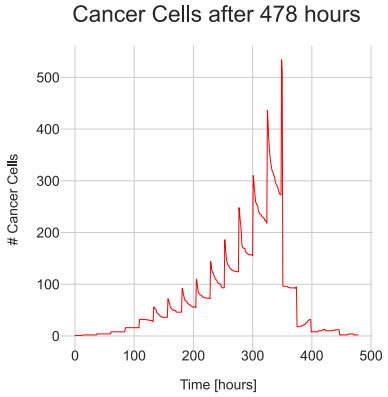


Figure 6.2: Number of cancer cells for a simulation with a nutrient consumption rate multiplied by 10.

One particularly noteworthy observation is the absence of oscillations in the evolution of cancer cells under low nutrient consumption conditions (shown in Figure 6.3). This suggests that there is no necrosis occurring in the center of the tumor, unlike the scenario with high nutrient consumption where such oscillations are present. Necrosis, which typically arises due to a lack of oxygen and nutrients, can lead to the death of cancer cells and contribute to tumor heterogeneity.

In conclusion, the comparison between Figure 6.2 and Figure 6.3 clearly demonstrates the impact of nutrient consumption on the growth and development of cancer cells. The absence of oscillations and necrosis in the low nutrient consumption scenario presents an intriguing avenue for further research.

To gain a deeper understanding of the behavior observed in these simulations, we will conduct an analysis of the differences in the trends exhibited by the cancer cell curves, as well as the variations in cell densities at the center of the grid.

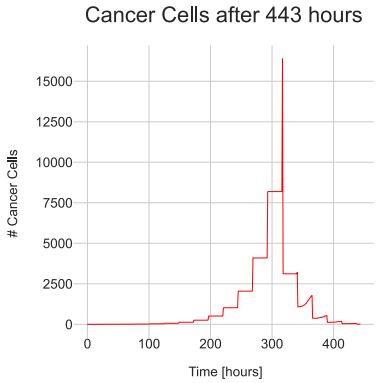


Figure 6.3: Number of cancer cells for a simulation with a nutrient consumption rate divided by 10.

Figure 6.4 illustrates the evolution of cancer cells in percentage throughout the simulation. These curves are normalized by the maximum number of cancer cells (which typically occurs just before radiotherapy begins).

Firstly, it is evident that radiotherapy treatment with low nutrient consumption often begins earlier than 350 hours. For the low nutrient consumption simulation, this is due to the absence of necrosis during the exponential growth phase, which results in a rapid increase in cancer cell number. Consequently, the threshold of 9,000 cancer cells is reached more quickly, leading to an earlier initiation of radiotherapy.

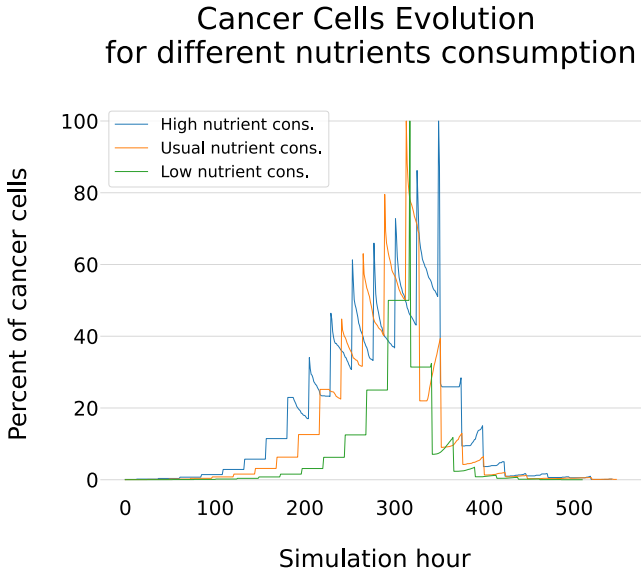


Figure 6.4: Cancer cells evolution in percentage through the simulation for different nutrient consumption rates.

Furthermore, the high and low nutrient consumption simulations exhibit relatively similar plateaus in terms of percentage following the first and second radiotherapy doses. In the high nutrient consumption simulation, this can be primarily attributed to the low number of cancer cells present within the tumor. Consequently, the initial doses of radiotherapy effectively eradicate a significant portion of these cells. Conversely, the low nutrient consumption simulation results in a highly concentrated tumor, as illustrated later in Figure 6.6. This implies that the targeted area of radiotherapy treatment will induce the death of a large number of cells, despite the increased density of the tumor.

However, regardless of these differences, all three simulations display a comparable behavior pattern. This observation suggests that the nutrient consumption rates may not have a substantial impact on the performance of our trained RL agents, as mentioned before.

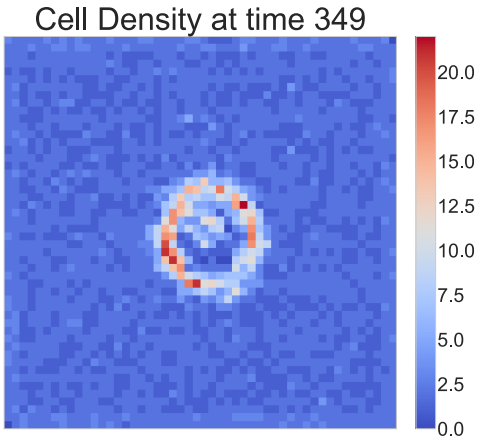


Figure 6.5: Cell density (Cancer Healthy) on the grid for simulation with high nutrient consumption rate.

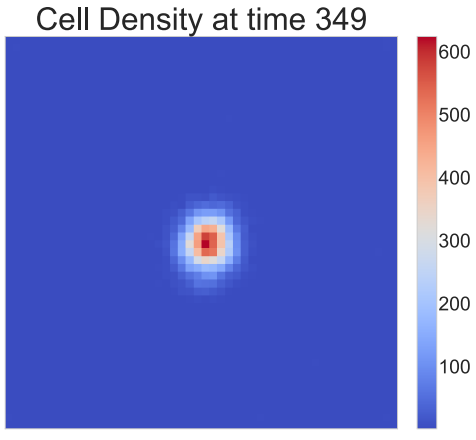


Figure 6.6: Cell density (Cancer Healthy) on the grid for simulation with low nutrient consumption rate.

A closer examination of Figure 6.5 and Figure 6.6 reveals notable differences in cancer cell concentration. Indeed, the concentration of cancer cells is significantly higher for the low nutrient consumption rate simulation. The necrosis observed in the high nutrient consumption simulation is clearly evident, as the center of the tumor contains fewer cells than the edges. This pattern reflects the effects of nutrient availability on cell proliferation and survival.

On the other hand, the high density of cancer cells in the center of the grid for low nutrient consumption simulations is quite striking. As explained earlier, this increased density is precisely why the radiotherapy doses are so efficient at killing cancer cells in these simulations. The densely packed cancer cells, coupled with the targeted radiotherapy approach, result in a substantial reduction of viable cells, contributing to tumor shrinkage and improved treatment outcomes.

Summary

In conclusion, the analysis of these simulations sheds light on the relationship between nutrient consumption rates and tumor dynamics. Despite variations in nutrient availability and cancer cell density, the RL agents effectively are still efficient in different scenarios (they always achieve 100% TCP even with extreme nutrient consumption rates). Table 6.3 highlights a summary of the findings of this section.

Table 6.3: Summary of the behavior of the simulation for low and high nutrients consumption rates.

Criterion	**Low Nutrient Consumption Rate**	**High Nutrient Consumption Rate**
Number of cancer cells	High number of cancer cells within the tumor. It can reach over 15 000 cancer cells at the same time on the grid.	Low number of cancer cells within the tumor. It never exceeds 1500 cancer cells at the same time on the grid.
Concentration of the cancer cells	Cells are concentrated in the center of the tumor. The cancer cell density is up to 600 cells in close range of the center.	Cells are concentrated on the edges of the tumor. The cancer cell density reaches a maximum of 20 cells on the edge of the tumor.
Necrosis	No, there is a constant growth of the tumor if no radiation dose is applied.	Yes, cancer cells die even without radiotherapy due to the lack of nutrients in the center of the tumor.
Radiotherapy Beginning	Radiotherapy treatment starts before 350 hours. The volumetric criterion of 9000 cells within the tumor is reached almost every time.	Radiotherapy treatment starts at 350 hours. The volumetric criterion of 9000 cells is never reached.

6.3 Cell Cycle Duration Changes

This section focuses on investigating the robustness of the agent when faced with **variations in cell cycle duration for cancer cells**. As discussed in Section 4.1.1, the duration of cell cycle stages can vary to a certain extent. Although the cell cycle of human cells typically lasts about 24 hours, we have opted to consider the cell cycle duration for animal cells in order to examine a broader range of durations. This approach allows us to analyze the effect of T_c, the cell cycle length, on tumor growth and treatment robustness.

To thoroughly explore this range, we have selected the following durations for each cell cycle stage: Gap 1 (G1) stage varies between 5 to 12 hours, Synthesis (S) stage varies between 8 to 10 hours, Gap 2 (G2) stage varies between 2 to 4 hours, and Mitosis (M) is fixed at 1 hour. By adjusting the cell cycle stages individually, we can gain valuable insights into how different stages affect the tumor's response to treatment and the overall efficiency of the therapy.

Cell Cycle Duration [h]	Gap 1 [h]	Synthesis [h]	Gap 2 [h]	Mitosis [h]	TCP [%]	Fractions [-]	Duration [h]	Doses [Gy]	Survival [-]	Start RT [h]	Init HCells [-]	Init CCells [-]
16	5	8	2	1	72.0	34.74 ± 11.33	833.76 ± 272.15	81.82 ± 26.95	0.8196	259	4134	9480
17	5	9	2	1	78.0	36.16 ± 11.56	867.84 ± 277.62	85.68 ± 26.7	0.8558	284	4079	10132
17	6	8	2	1	88.0	31.42 ± 9.879	754.08 ± 237.1	76.58 ± 21.92	0.8777	272	4124	10334
18	5	10	2	1	90.0	31.26 ± 11.53	750.24 ± 276.89	78.18 ± 23.51	0.9283	288	4113	9956
17	5	8	3	1	94.0	29.46 ± 9.761	707.04 ± 234.27	73.4 ± 0.349	0.9086	272	4157	9424
18	5	9	3	1	94.0	31.58 ± 9.514	757.92 ± 228.34	79.76 ± 18.96	0.9835	265	4256	10420
18	6	9	2	1	96.0	27.42 ± 9.366	658.08 ± 224.78	70.7 ± 8.375	0.9614	300	4142	9090
18	7	8	2	1	96.0	26.8 ± 0.7793	643.2 ± 62.705	68.44 ± 14.35	0.9299	309	4100	9746
19	6	10	2	1	96.0	26.46 ± 7.195	635.04 ± 172.68	68.2 ± 3.741	0.955	305	4027	9408
20	8	9	2	1	96.0	23.38 ± 7.238	561.12 ± 173.72	59.6 ± 2.167	0.9566	301	4185	9996
19	5	9	4	1	98.0	24.12 ± 7.021	578.88 ± 168.52	62.44 ± 11.41	0.9594	280	4195	10188
20	6	10	3	1	98.0	24.04 ± 7.799	576.96 ± 187.19	61.8 ± 5.182	0.9608	304	4142	10446
18	5	8	4	1	100.0	25.82 ± 8.289	619.68 ± 198.93	67.5 ± 6.889	0.9756	307	4117	9850
18	6	8	3	1	100.0	27.06 ± 10.12	649.44 ± 243.02	69.76 ± 20.35	0.963	264	4243	9666
19	5	10	3	1	100.0	24.8 ± 0.1057	595.2 ± 46.537	65.38 ± 12.18	0.9709	308	4150	9788
19	6	8	4	1	100.0	22.6 ± 0.4093	542.4 ± 53.824	60.04 ± 13.93	0.9627	309	4172	10428
19	6	9	3	1	100.0	23.38 ± 6.829	561.12 ± 163.89	61.82 ± 14.33	0.9704	323	4134	9726
19	7	8	3	1	100.0	22.62 ± 6.177	542.88 ± 148.24	59.8 ± 2.745	0.9768	327	4075	10696
19	8	8	2	1	100.0	24.58 ± 7.866	589.92 ± 188.79	64.12 ± 16.85	0.9775	307	4130	10214
20	5	10	4	1	100.0	22.96 ± 6.532	551.04 ± 156.78	59.84 ± 12.89	0.9691	321	4112	10484
20	6	9	4	1	100.0	19.7 ± 0.8466	472.8 ± 16.319	52.66 ± 9.22	0.9576	314	4191	9006
20	7	8	4	1	100.0	20.18 ± 4.831	484.32 ± 115.96	54.16 ± 10.38	0.9596	340	4153	10182
20	8	8	3	1	100.0	20.32 ± 5.334	487.68 ± 128.02	55.2 ± 0.768	0.9733	341	4131	9224
21	6	10	4	1	100.0	19.96 ± 5.102	479.04 ± 122.46	53.2 ± 0.8285	0.9672	350	4110	3540
21	8	10	2	1	100.0	21.84 ± 5.675	524.16 ± 136.21	58.0 ± 1.324	0.9655	339	4147	9240
21	8	8	4	1	100.0	18.56 ± 4.341	445.44 ± 104.18	50.54 ± 9.421	0.9685	315	4237	10054
21	8	9	3	1	100.0	20.72 ± 4.103	497.28 ± 98.492	54.78 ± 8.732	0.9666	350	4166	7152
22	8	10	3	1	100.0	19.78 ± 4.423	474.72 ± 106.17	53.04 ± 9.987	0.971	350	4151	3820
22	8	9	4	1	100.0	18.28 ± 3.893	438.72 ± 93.45	49.72 ± 8.806	0.9716	309	4192	9312
23	8	10	4	1	100.0	17.16 ± 3.413	411.84 ± 81.932	47.06 ± 8.001	0.968	350	4267	5858

Table 6.4: Performances of the selected agent (see Section 6.1) on environments with different cell cycle duration. Each experiment conducted has been repeated 100 times to compute the mean and standard deviation of the key performance indicators.

Table 6.4 presents the evaluation of the selected agent (Sarsa, see Section 6.1) when introduced to new environments with altered cell cycle durations for cancer cells. In these environments, the cell cycle duration of healthy cells remains constant at 24 hours, while the cancer cells' cell cycle durations vary.

Our findings suggest that when the cell cycle duration T_c of cancer cells falls below 21 hours, the agent's robustness is compromised. At this threshold, the rate of cell division for cancerous cells accelerates to such an extent that the agent is no longer capable of effectively eradicating the tumor.

Table 6.4 demonstrates an interesting trend: as the cell cycle duration T_c decreases, the standard deviation of all key performance indicators increases. For example, for a T_c of 18 hours, the duration of the entire treatment process can vary significantly, ranging from 406 hours to 892 hours after the start of radiotherapy. Consequently, it becomes increasingly challenging to accurately estimate the treatment requirements in terms of duration or the number of treatment fractions.

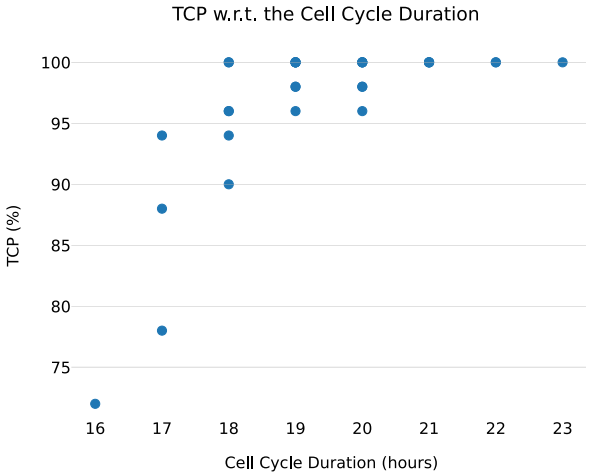


Figure 6.7: TCP w.r.t. the cell cycle duration variations for a pre-trained agent (Sarsa).

Figure 6.7 illustrates the TCP achieved by the agent in the modified environments with varying cell cycle durations. This graphical representation emphasizes that the agent maintains robustness in response to changes in cell cycle duration only when the duration is equal to or greater than 21 hours. For durations below this threshold, the TCP begins to decline. This observation allows us to conclude that the agent's robustness is limited to cell cycle durations exceeding 21 hours. By identifying the specific threshold at which the agent's performance begins to deteriorate, we can focus on enhancing the agent's adaptability in combating tumors by increasing the robustness of the agent with regard to variations of the cell cycle durations.

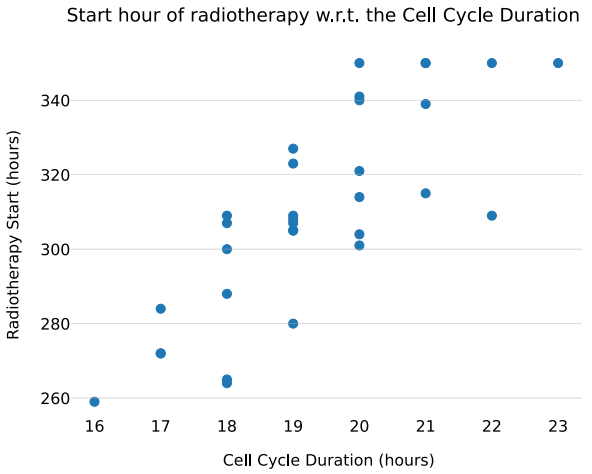


Figure 6.8: Starting time of radiotherapy treatment based on the volumetric criterion.

Figure 6.8 illustrates the relationship between the starting time of radiation therapy and the cell cycle duration T_c. The default starting value for the therapy is set at 350 hours. The graph demonstrates that as the cell cycle duration T_c decreases, the initiation of radiation therapy occurs earlier. Although there is no precise equation defining the correlation, a general linear trend can be observed between the starting hour of radiation therapy and the cell cycle duration T_c. This implies that shorter cell cycles necessitate the earlier implementation of therapy to effectively combat rapidly dividing cancer cells.

Now that the basic observations have been made, we can get interested in the cancer cells behavior in modified cell cycle duration simulations to understand more deeply the accelerated proliferation of such simulations.

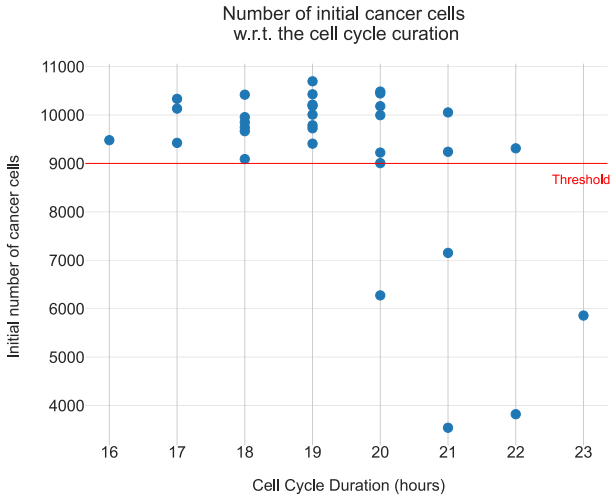


Figure 6.9: Number of cancer cells at the start of radiotherapy.

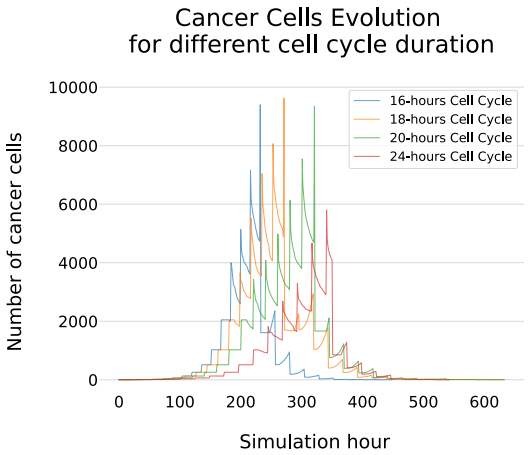


Figure 6.10: Number of cancer cells w.r.t. simulation time for different cell cycle duration (with Sarsa 17 agent).

Figure 6.9 demonstrates the number of cancer cells present at the start of radiation therapy. The graph highlights that for cell cycle durations T_c less than 20 hours, the threshold for cancer cell proliferation is consistently reached. These conditions can be considered aggressive due to the swift multiplication of cancer cells, leading to a more difficult treatment situation. In such cases, it becomes essential to begin radiation therapy as early as possible to combat the aggressive nature of the cancer and enhance the likelihood of a successful treatment outcome.

Figure 6.10 illustrates the growth of cancer cell populations for various cell cycle durations, T_c. As previously mentioned, we observe that the threshold of 9,000 cancer cells is attained in the simulations with the three shortest cell cycle durations. Evidently, the shorter the cell cycle, the more quickly the peak is reached, as cancer cells proliferate at a faster rate. The 16-hour simulation concludes more swiftly compared to the others. This is because the chosen agent (Sarsa 17) had to make more aggressive decisions as a response to the high degree of aggressiveness in the environment itself. Consequently, higher doses were selected for this simulation, ultimately resulting in the rapid eradication of the tumor.

Summary

In this section, we have made several key observations regarding the robustness of our agent in relation to cell cycle duration. To summarize, these findings include:

- When the cell cycle duration (T_c) is less than 21 hours, the robustness of our agent is compromised.
- When T_c becomes smaller, the standard deviation of the key performance indicators increases, leading to more uncertainty in the treatment requirements.
- A linear relationship is observed between the starting hour of radiation therapy and the cell cycle duration (T_c).
- For T_c values less than 20 hours, the threshold of 9000 cancer cells is consistently reached. As a reminder, this threshold serves as a guideline for administering the first dose earlier if the tumor size is substantial.
- As T_c decreases, higher peaks are observed in the cancer cell population due to the threshold being surpassed more frequently.

These observations provide valuable insights into the performance of our agent with respect to cell cycle duration variations. The next section will help us to render the agent more robust with regard to the cell cycle duration. The objective is to build an agent able to provide the best radiotherapy treatment, whatever the cell cycle of the environment.

6.3.1 Cell cycle robustness through re-training

In the previous section, we observed the need for improving the robustness of our agent. In this section, we aim to enhance the agent's performance by training it on modified environments with various cell cycle durations. We will follow the methodology described in Section 5.3.

Cell Cycle Duration: 16 h

Fractions (-)	Doses (Gy)	Duration (h)	TCP (%)	Survival (%)
47.58 ± 10.05	105.32 ± 10.05	1141.92 ± 241.20	34.0	73.82
22.62 ± 11.50	64.06 ± 11.50	542.88 ± 276.07	92.0	87.23
17.06 ± 4.05	54.74 ± 4.05	409.44 ± 97.24	100.0	93.38
14.7 ± 3.20	50.94 ± 3.20	352.8 ± 76.84	100.0	89.77
14.26 ± 3.21	50.44 ± 3.21	342.24 ± 76.92	100.0	88.46

Cell Cycle Duration: 18 h

Fractions (-)	Doses (Gy)	Duration (h)	TCP (%)	Survival (%)
35.92 ± 13.85	79.94 ± 13.85	862.08 ± 332.37	68.0	82.58
46.12 ± 8.08	100.16 ± 8.08	1106.88 ± 194.01	52.0	89.74
14.74 ± 2.83	48.42 ± 2.83	353.76 ± 67.85	100.0	94.06
15.14 ± 3.28	48.7 ± 3.28	363.36 ± 78.73	100.0	93.37
18.44 ± 3.72	54.84 ± 3.72	442.56 ± 89.31	100.0	95.93

Cell Cycle Duration: 20 h

Fractions (-)	Doses (Gy)	Duration (h)	TCP (%)	Survival (%)
23.38 ± 5.94	59.1 ± 5.94	561.12 ± 142.62	98.0	92.78
21.44 ± 6.77	56.6 ± 6.77	514.56 ± 162.43	98.0	94.07
12.82 ± 3.64	42.44 ± 3.64	307.68 ± 87.42	100.0	93.07
13.28 ± 2.98	44.12 ± 2.98	318.72 ± 71.52	100.0	92.43
14.56 ± 2.90	45.38 ± 2.90	349.44 ± 69.59	100.0	93.83

Table 6.5: First 5 training epochs of the robustness process. For each training epoch, 2000 simulations have been conducted to compute the mean and standard deviation properly.

Table 6.5 presents the results of the first five training epochs aimed at enhancing the agent's robustness to varying cell cycle durations, as described in Section 6.1. The rapid convergence achieved is noteworthy; after just three epochs, a TCP of 100% is reached, and this performance is maintained throughout the subsequent training epochs. This allows us to confidently conclude that the convergence has been successfully achieved.

It is worth noting that the differences observed between the 16-hours and 18-hours cell cycle durations are not statistically significant for simulation achieving a TCP of 100%.

However, as anticipated, we do find a general trend in the data: as cell cycle duration decreases, the required fractions, doses, and treatment duration for effective tumor control tend to increase. Table 6.5 provides the standard deviation values of the key performance indicators. Previously, we observed a tendency for standard deviations to increase as cell cycle duration decreases. This relationship is no longer as apparent. When selecting a robust agent, it is important to consider this criterion: the lower the standard deviation of each key performance indicator, the better the agent is.

6.3.2 Comparison with agents trained on environments with various cell cycle duration

This section aims to compare the selected agent that has been rendered robust and agents that are directly trained in the considered environments with various cell cycle duration (see previous section). We will only present the results for the 16-hour cell cycle environment because similar observations can be made on the 18-hour and 20-hour environments.

Agent	**TCP (%)**	**Dose (Gy)**	**Fractions (-)**	**Duration (h)**	**Survival (%)**
Selected agent retrained	100.0	49.52 ± 3.12	14.02 ± 3.12	336.5 ± 74.9	91.57
Sarsa directly trained	100.0	46.75 ± 3.11	13.88 ± 3.11	333.1 ± 74.7	91.48

Table 6.6: Best performances achieved by the selected agent Sarsa 17, retrained on an environment with a cell cycle duration of 16 hours and an expected sarsa agent directly trained on a 16-hour cell cycle environment. For each agent, 100 simulations have been conducted to compute the mean and standard deviation properly.

Table 6.6 presents the comparison between the Sarsa 17 agent and a new Sarsa agent trained only on the environment with a cell cycle of 16 hours. The performances depicted in table 6.6 are the best performances achieved by the new expected sarsa agent. It highlights that no noticeable improvements are made.

6.3.3 Conclusion

To conclude, our agent has achieved to be robust whatever the cell cycle duration T_c. The environments with a small value of cell cycle duration are not difficult to solve since the convergence of the agent has been achieved in only three training epochs. The choice of an agent among the robust ones has to be made with regard to the standard deviation of the key performance indicators and the criteria described in section 5.1.2.

There is no difference observed in terms of performance between an agent trained directly on an environment with various cell cycle duration and an agent retrained on such environments.

6.4 Radiosensitivities Changes

In this section, the radiosensitivities are under study.

Sum of radiosensitivities	Gap 1	Synthesis	Gap 2	Mitosis	Quiescence	TCP [%]	Fractions [-]	Duration [h]	Doses [Gy]	Survival [-]	Start RT [h]	HCells [-]	CCells [-]
0.583333	0.75	0.25	0.75	0.75	0.75	28.0	47.92 ± 6.327	1150.0 ± 151.8	123.4 ± 14.6	0.8952	350	4180	2334
0.593750	0.75	0.25	0.75	1.00	0.25	24.0	48.82 ± 4.978	1171.6 ± 119.4	123.3 ± 16.4	0.888	350	4251	7193
0.593750	0.75	0.25	0.75	1.00	0.50	38.0	47.36 ± 6.192	1136.6 ± 148.6	122.9 ± 15.15	0.8888	350	4203	5854
0.593750	0.75	0.25	0.75	1.00	0.75	20.0	49.14 ± 4.98	1179.3 ± 119.5	125.0 ± 13.7	0.8682	350	4249	7610
0.645833	1.00	0.25	0.50	0.50	0.25	52.0	44.58 ± 7.98	1069.9 ± 191.5	112.4 ± 17.84	0.8993	349	4236	9134
0.645833	1.00	0.25	0.50	0.50	0.50	46.0	45.92 ± 6.97	1102.0 ± 167.3	114.9 ± 16.7	0.9084	336	4256	9006
0.645833	1.00	0.25	0.50	0.50	0.75	32.0	47.52 ± 6.096	1140.4 ± 146.3	118.3 ± 14.2	0.9018	350	4247	3902
0.656250	1.00	0.25	0.50	0.75	0.25	50.0	44.9 ± 0.1098	1077.6 ± 194.63	113.5 ± 16.7	0.9258	350	4186	6242
0.656250	1.00	0.25	0.50	0.75	0.50	70.0	40.36 ± 9.595	968.64 ± 230.28	106.7 ± 19.1	0.9428	350	4155	4227
0.656250	1.00	0.25	0.50	0.75	0.75	44.0	44.9 ± 0.5188	1077.6 ± 204.45	111.1 ± 14.6	0.9122	350	4138	4837
0.666667	1.00	0.25	0.50	1.00	0.25	66.0	42.3 ± 0.2346	1015.2 ± 197.63	106.0 ± 16.2	0.9226	350	4192	5983
0.666667	1.00	0.25	0.50	1.00	0.50	70.0	42.44 ± 8.087	1018.5 ± 194.0	111.3 ± 16.3	0.9485	341	4177	9128
0.666667	1.00	0.25	0.50	1.00	0.75	58.0	42.16 ± 9.226	1011.8 ± 221.4	107.0 ± 17.9	0.9299	350	4259	4022
0.687500	1.00	0.25	0.75	0.50	0.25	84.0	38.64 ± 8.788	927.36 ± 210.91	100.5 ± 17.3	0.9556	350	4167	6510
0.687500	1.00	0.25	0.75	0.50	0.50	76.0	40.32 ± 8.509	967.68 ± 204.23	105.3 ± 16.5	0.9698	350	4191	3968
0.687500	1.00	0.25	0.75	0.50	0.75	82.0	38.2 ± 0.9955	916.8 ± 15.893	100.2 ± 18.2	0.966	350	4223	4795
0.697917	1.00	0.25	0.75	0.75	0.25	80.0	36.76 ± 10.46	882.24 ± 251.19	95.84 ± 20.63	0.9643	350	4232	4073
0.697917	1.00	0.25	0.75	0.75	0.50	90.0	36.06 ± 8.396	865.44 ± 201.5	96.52 ± 17.93	0.9687	350	4180	4242
0.697917	1.00	0.25	0.75	0.75	0.75	86.0	37.06 ± 9.634	889.44 ± 231.21	95.9 ± 0.327	0.9548	350	4165	5707
0.708333	1.00	0.25	0.75	1.00	0.25	90.0	34.86 ± 9.068	836.64 ± 217.64	92.6 ± 7.816	0.9654	350	4130	2932
0.708333	1.00	0.25	0.75	1.00	0.50	92.0	36.96 ± 8.163	887.04 ± 195.91	97.0 ± 7.079	0.9747	350	4204	4795
0.708333	1.00	0.25	0.75	1.00	0.75	92.0	33.62 ± 8.51	806.88 ± 204.26	90.16 ± 17.8	0.9644	350	4210	3654
0.729167	1.00	0.25	1.00	0.50	0.25	90.0	33.76 ± 9.551	810.24 ± 229.22	88.28 ± 18.73	0.9641	350	4227	5480
0.729167	1.00	0.25	1.00	0.50	0.50	90.0	34.44 ± 8.131	826.56 ± 195.16	91.46 ± 16.55	0.9727	350	4183	6141
0.729167	1.00	0.25	1.00	0.50	0.75	96.0	33.66 ± 8.739	807.84 ± 209.75	89.12 ± 18.38	0.9739	350	4184	6397
0.739583	1.00	0.25	1.00	0.75	0.25	98.0	32.18 ± 7.819	772.32 ± 187.67	85.7 ± 6.243	0.9642	337	4200	9162
0.739583	1.00	0.25	1.00	0.75	0.50	100.0	31.5 ± 0.7327	756.0 ± 61.586	84.96 ± 14.82	0.9765	350	4216	4305
0.739583	1.00	0.25	1.00	0.75	0.75	100.0	32.72 ± 8.699	785.28 ± 208.78	87.48 ± 18.26	0.9819	350	4112	5120
0.750000	1.00	0.25	1.00	1.00	0.25	100.0	30.72 ± 7.216	737.28 ± 173.2	83.46 ± 15.77	0.9766	350	4156	3169
0.750000	1.00	0.25	1.00	1.00	0.50	100.0	29.1 ± 0.3192	698.4 ± 99.662	79.48 ± 18.02	0.9683	350	4161	4068

Table 6.7: Performances of Sarsa 17 agent on environments with different radiosensitivity. Each experiment conducted has been repeated 100 times to compute the mean and standard deviation of the key performance indicators.

Over 150 simulations have been conducted, experimenting with various levels of radiosensitivity. Table 6.7 displays a selection of experiments where the radiosensitivity is notably low, leading to a decrease in TCP. This table highlights the sum of radiosensitivities [1] which corresponds to the weighted average of the individual radiosensitivity with regard to the duration of each stage of the cell cycle, the individual radiosensitivity for each stage, the key indicator performances, the hour of commencement of the radiotherapy treatment, and the number of healthy/cancer cells at this hour. Numerous simulations have been conducted with higher and lower radiosensitivities than those presented in Table 6.7, but they have not been included in this report to maintain clarity. As anticipated, reduced radiosensitivity makes it more difficult for radiation therapy to eradicate cells, whereas increased radiosensitivity enhances the therapy's effectiveness, as each dose carries a greater likelihood of killing cells.

A distinct trend can be observed between radiosensitivity and treatment duration. Naturally, as radiosensitivity decreases, the treatment duration will be longer due to the reduced effectiveness of radiation doses. This relationship is also reflected in the survival rate of healthy cells, which decreases as radiosensitivity decreases. As a reminder, the survival rate is the ratio of the final number of healthy cells remaining to the number of healthy cells at the beginning of radiotherapy treatment.

We can notice that the TCP decreases extremely fast as the radiosensitivity decreases. From a radiosensitivity of 0.7 to 0.65, the TCP goes from about 90% to only between 50% and 70%. The same extreme decrease is observed between 0.65 and 0.6 of radiosensitivity, with a TCP of 28% at the end.

[1] Note that radiosensitivity and sum of radiosensitivities may be used in similar way.

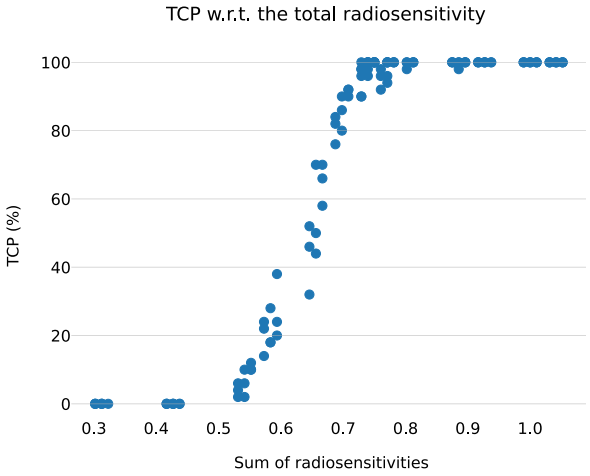


Figure 6.11: TCP w.r.t. the sum of radiosensitivities among the cell cycle for a pre-trained agent (Expected Sarsa).

As in the section discussing the robustness of agents facing variations in cell cycle duration, it is evident that there also exists a threshold where the combined radiosensitivities of the tumor cells become too low to achieve a TCP of 100%. For the sake of simplicity, we will assume that this threshold value is 0.8. When the sum of radiosensitivities falls below this value, the environment is considered too hostile for the agent to effectively eradicate the tumor.

The aggressiveness of the environment in this context is primarily a result of the decreasing sensitivity of cancer cells to radiation therapy doses. As the radiosensitivity of the tumor cells decreases, it becomes increasingly challenging for radiation therapy to achieve the desired therapeutic effect. This is particularly concerning, as radiation therapy plays a crucial role in the treatment of many cancer types.

Table 6.7 also highlights the influence of the quiescent mode. Indeed, one can observe that for the same "Sum of radiosensitivities", different TCP values are evident. This variation can be attributed to the influence of the quiescence mode on radiosensitivity. It has been demonstrated that the TCP can fluctuate between 2 to 6% due to changes in the radiosensitivity of the quiescent mode. It is essential to note that only healthy cells possess this quiescent mode, while cancer cells do not. This implies that it is the sensitivity of healthy cells to radiotherapy that affects the TCP in this context.

Understanding the precise mechanism of how TCP is influenced by the radiosensitivity of the quiescent mode is challenging. One goal of our therapeutic approach is to minimize the number of healthy cells damaged or destroyed by radiotherapy treatment. Intuitively, one might assume that having lower radiosensitivity is preferable in order to protect more healthy cells. However, this is not always the case in practice. The complexity of this mechanism lies in the number of healthy cells that enter the quiescent mode and how their radiosensitivity impacts the overall treatment outcome.

Summary

In this section, we have made several key observations regarding the robustness of our agent in relation to radiosensitivity. To summarize, these findings include:

- Reducing the radiosensitivity leads to a decrease in the TCP.
- The treatment duration increases with the decrease of the sum of radiosensitivities.
- Current selected agent is considered robust to environments with maximum a sum of radiosensitivity equal to 0.8.
- Quiescent mode influences also the TCP value. It is an underlying mechanism difficult to really understand.

6.4.1 Radiosensitivity robustness through re-traininng

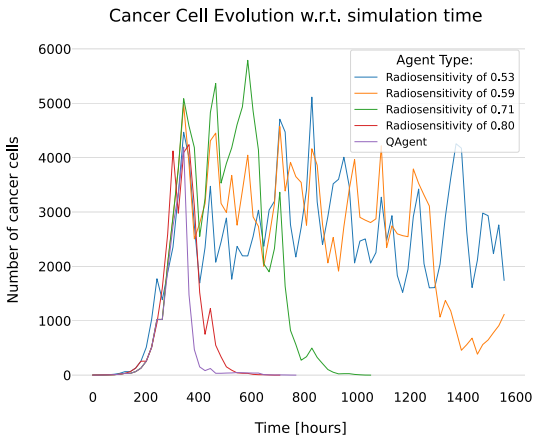


Figure 6.12: Number of cancer cells w.r.t. time for different agent types.

Figure 6.12 illustrates the number of cancer cells in relation to simulation time for various types of agents. It shows a smoothened version of the cancer cells curves. The default radiosensitivity value is approximately 0.96. The "radiosensitivity" agents refer to those that have been adapted from the agent described in Section 6.1 for environments with corresponding radiosensitivity levels. The figure demonstrates that a radiosensitivity of 0.8 exhibits a similar pattern to the default value. This level of radiosensitivity aligns with the threshold identified earlier, beyond which the performance of non-robust agents begins to deteriorate. In the case of a radiosensitivity of 0.7, the impact of low radiosensitivity becomes evident, as the radiation therapy is less effective compared to scenarios with higher radiosensitivity.

The improved robustness of retrained radiosensitivity agents consistently leads to a substantial extension in treatment duration. Table 6.8 and Table 6.9 showcase the results obtained by the agent discussed in Section 6.1, which demonstrates increasing robustness as the retraining process progresses.

Fractions (-)	Doses (Gy)	Duration (h)	TCP (%)	Survival (%)	Epoch (-)
48.30	102.70	1159.20	30.0	0.859025	0
50.90	89.62	1221.60	2.0	0.719414	1
49.80	101.34	1195.20	12.0	0.775148	2
50.66	100.80	1215.84	8.0	0.774551	3
51.00	102.48	1224.00	2.0	0.777975	4
50.94	80.50	1222.56	2.0	0.579464	5
51.00	95.84	1224.00	0.0	0.720544	6
47.18	95.06	1132.32	24.0	0.739334	7
48.22	103.34	1157.28	36.0	0.788282	8
50.84	104.40	1220.16	8.0	0.746878	9
49.82	105.66	1195.68	10.0	0.836206	10
48.02	96.74	1152.48	22.0	0.722568	11
48.68	100.22	1168.32	16.0	0.727170	12
49.90	109.04	1197.60	12.0	0.840961	13
50.44	89.24	1210.56	6.0	0.607437	14
48.36	96.60	1160.64	28.0	0.738419	15
37.46	96.50	899.04	94.0	0.928180	16

Table 6.8: Robust agent results for a radiosensitivity of 0.71 (first training epochs).

Fractions (-)	Doses (Gy)	Duration (h)	TCP (%)	Survival (%)	Epoch (-)
40.50	99.90	972.00	84.0	0.907140	17
41.30	103.06	991.20	80.0	0.937118	18
38.40	98.22	921.60	84.0	0.923954	19
40.08	98.08	961.92	78.0	0.943790	20
43.14	103.26	1035.36	68.0	0.917536	21
36.24	92.46	869.76	96.0	0.889443	22
34.62	90.16	830.88	94.0	0.908216	23
34.38	90.72	825.12	96.0	0.900740	24
33.48	90.72	803.52	96.0	0.934446	25
29.42	82.74	706.08	100.0	0.910634	26
29.06	81.90	697.44	100.0	0.925168	27
27.30	78.84	655.20	98.0	0.908195	28
31.72	84.24	761.28	96.0	0.948888	29
28.36	81.36	680.64	100.0	0.933882	30
29.76	82.64	714.24	98.0	0.928146	31
33.88	90.16	813.12	100.0	0.937559	32
24.48	73.06	587.52	100.0	0.926567	33

Table 6.9: Robust agent results for a radiosensitivity of 0.71 (last training epochs).

The key performance indicators demonstrate consistent improvement as the re-training process progresses. As shown in Table 6.9, the re-training process starts to stabilize in the final iterations. Notably, the TCP is almost consistently around 100%, indicating the onset of convergence. Consequently, we now possess an agent that can effectively cope with an environment exhibiting a radiosensitivity of 0.71 and can be repurposed for similar scenarios.

We have also attempted to train agents for environments with lower radiosensitivities. However, even after 70 iterations of training, these agents never reach a TCP of 100%. This suggests that it becomes increasingly challenging to develop an agent robust enough to handle extremely hostile environments. No detailed convergence analysis has been conducted, but the training time required for an agent to adapt to an environment with a radiosensitivity lower than 0.7 is significantly increasing. This implies that as the environment becomes more aggressive, the training complexity and time investment grow substantially.

6.4.2 Comparison with agents trained on environments with various radiosensitivity

This section is dedicated to comparing the selected agent (Sarsa 17, see section 6.1) retrained on an environment with a radiosensitivity of 0.7 and an expected sarsa agent directly and only trained on an environment with a radiosensitivity of 0.7.

For the sake of clarity, we define the "Robust agent" that is the selected Sarsa 17 agent re-trained on an environment with a radiosensitivity of 0.7 and we define the "Radiosensitivity agent" that is an agent directly trained on an environment with a radiosensitivity of 0.7.

Agent	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Robust agent	100.0	73.06 ± 12.8	24.48 ± 5.86	587.52 ± 140.7	91.57
Radiosensitivity agent	100.0	72.31 ± 16.5	23.27 ± 6.82	558.5 ± 163.7	93.43

Table 6.10: Best performances achieved by the Robust agent and the Radiosensitivity agent. For each agent, 2000 simulations have been conducted to compute the mean and standard deviation properly.

Table 6.10 details the key performance indicators for the highlighted agents. Notably, the convergence of the training process was not achieved within the same epoch range for the different agents. In fact, the process of training to achieve the performance metrics presented in Table 6.10 with our chosen retrained agent, the Robust agent, necessitated over 30 epochs. In contrast, the radiosensitivity agent achieved these levels in less than 20 epochs.

A noticeable point is the slightly higher standard deviations of the key performance indicators associated with the radiosensitivity agent. This heightened variability can be tied back to the agent's relatively brief training period. With fewer epochs, the radiosensitivity agent had less time to stabilize and reduce the variation in its performance.

Upon comparison, the performance metrics of the radiosensitivity agent appear marginally superior to those of the robust counterpart. However, this difference is subtle and may not have significant practical implications.

In a separate experiment, we attempted to train an agent in an environment with a radiosensitivity parameter set to 0.6. Unfortunately, this relatively low radiosensitivity level proved challenging for the agent's training. The agent was unable to reach convergence within a reasonably small number of epochs, indicating that it's more difficult to identify optimal treatment strategies for tumors that demonstrate low sensitivity to radiation doses.

6.4.3 Conclusion

In summary, the agent described in Section 6.1 exhibits robustness up to a radiosensitivity of 0.8. With additional re-training tailored to specific environments, the agent can be further enhanced to cope with environments displaying radiosensitivities as low as 0.7. However, it is important to note that achieving complete robustness in extremely aggressive environments remains a challenging endeavor due to the escalating complexity and training time required.

6.5 Mean value of first and second radiation dose

This section is dedicated to describing the agents' behavior for the first two doses of radiation. We want to understand the importance of the two first doses with regard to the type of environment.

	Radiosensitivity of 0.8	Radiosensitivity of 0.7	Radiosensitivity of 0.6	16-hour cell cycle	18-hour cell cycle	20-hour cell cycle
1st dose	2.13	1.91	1.64	3.25	3.4	3.55
2nd dose	3.06	1.74	2.31	3.4	3.5	3.35

Table 6.11: Mean of the first and second radiation dose administered with regard to different types of agents. For each agent, 2000 simulations have been conducted to compute the mean value properly.

Table 6.11 presents the average values of the first and second radiation doses, segmented by different types of agents. These agents are variants of the Sarsa 17, each retrained in its corresponding environment. This approach offers valuable insights into the optimal strategy for interacting with a specific environment. The table reveals a pronounced trend: cell cycle environments require a higher dosage for both the first and second radiation doses. In contrast, radiosensitivity environments necessitate lower dosage values. This finding not only paves the way for future observations but also enriches our understanding of the strategies an agent should adopt to successfully navigate a given environment.

Chapter 7

ML-based Adaptive Decision-Making: Utilizing Tumor Observations for Robust Radiotherapy Treatments

A crucial question that has emerged throughout this work is determining, based on observations during treatment, when we should decide when a different, more aggressive agent should be employed. This decision becomes necessary when, during treatment, the tumor continues to grow in size and mass, which may indicate that the current treatment is suboptimal. In such cases, using agents that have been trained and rendered robust can be beneficial, as they are better equipped to handle more challenging environments where baseline treatments or non-robust [1] agents may fail.

It is important to note, however, that employing a robust agent often entails administering higher doses to the patient (or to induce more side effects due to the large doses administered), consequently increasing the total dose delivered. As a result, there is an inherent trade-off between keeping the total amount of dose administered as low as possible while still effectively eradicating the tumor. The balance in this trade-off is determined by the precise timing when the treatment agent should be switched to a more aggressive one.

To optimize this trade-off, it is essential to establish a well-founded monitoring strategy that tracks the tumor's response to treatment. Regular assessments, such as imaging tests, can provide valuable information regarding tumor growth and the effectiveness of the current treatment plan. The frequency of these observations should be determined based on factors such as tumor type, growth rate, and the patient's overall health.

7.1 Simulation tendencies

In order to establish a criterion for selecting a more aggressive agent in cancer treatment simulations, it is crucial to gain insights into the factors contributing to successful or failing outcomes. By understanding the complex dynamics that lead to the eradication or growth of a tumor, we can better determine the optimal criteria for agent selection.

In the following sections, we present two distinct methods for assessing the effectiveness of different treatment agents in cancer simulations. The first method is based on analyzing the evolution of cancer cells throughout the simulation. While this approach can yield valuable insights, it is important to note that obtaining such detailed observations in real-life patients is significantly more challenging. As a result, it may not always be feasible to apply this method in clinical settings.

To address this limitation, we introduce a second method that relies on medical imaging data of the tumor. By using machine learning algorithms, we can analyze these images to predict the likelihood of achieving a

[1] Non-robust agents are agents that have been trained on the benchmark environment (see Table 4.1) but that have not been trained again on a more aggressive environment.

TCP of 100% following a given treatment strategy.

The development and refinement of these machine learning algorithms will play a critical role in determining the success of this second method. By leveraging advances in artificial intelligence and computational modeling, we can create powerful tools to assist healthcare professionals in making more informed decisions about cancer treatment plans. In turn, this can lead to more personalized and targeted interventions that are better suited to each patient's unique circumstances, ultimately improving treatment outcomes and overall survival rates.

7.2 Cancer Cells' Evolution Observations

As mentioned just before, we will use the cancer cells' evolution as observations made during treatment to perform a prediction about the outcome of the treatment.

To achieve this, a dataset of 2,600 simulations is utilized to study the behavior of the tumor under different conditions. These simulations will help identify the factors influencing the success or failure of cancer treatments. In this dataset, half of the simulations (1,300) are successful, resulting in the complete eradication of the tumor, while the other half (1,300) are considered failures, as the tumor continues to grow at the end of the simulation.

Cancer cell populations are plotted with respect to simulation time, allowing for clear visualization of the tumor's growth or regression patterns. This will help in the identification of crucial time points or specific factors contributing to the tumor's response to treatment. It is important to note that these simulations may involve various treatments, combinations of treatments, or different agent aggressiveness levels.

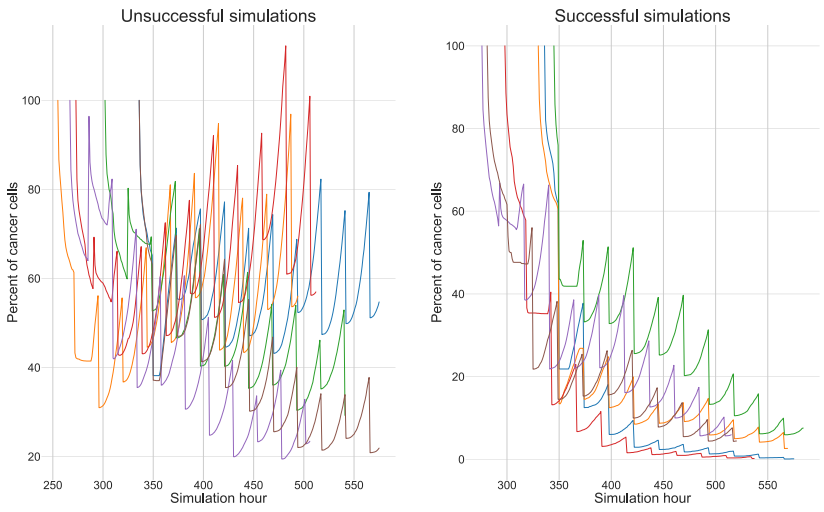


Figure 7.1: Simulation tendencies for fail/successful outcomes.

Figure 7.1 presents the trends observed in both successful and unsuccessful simulations, with 6 examples for each category. These graphs illustrate the changes in the number of cancer cells throughout the course of the simulations, specifically focusing on the effects of radiotherapy treatments. Two distinct patterns can be observed in these simulations, which are as follows:

1. Unsuccessful simulations are characterized by a higher degree of oscillations in the cancer cell population after the first two radiotherapy doses. This suggests that the cancer cells in these simulations are more resilient and continue to proliferate despite the treatment. These oscillations may be due to the treatment not being potent enough to effectively target and eliminate the cancerous cells.

2. On the other hand, successful simulations demonstrate a more straightforward trajectory, as the majority of cancer cells are eradicated after the initial two doses of radiotherapy. This indicates that the treatment is effective in targeting and destroying the cancer cells, ultimately leading to a decline in their population.

7.2.1 TCP classifier and TCP regressor

TCP binary classifier

In the quest to find a criterion that will determine whether a more aggressive treatment approach should be used or not, both a classifier and a regressor are utilized. The primary objective is to create a predictive model that can determine if the simulation will be successful based on the number of cancer cells present at various stages in the simulation. In other words, the model takes as input an array containing the number of cancer cells, starting from the beginning of radiotherapy up to a specific time instant (e.g., after 48 hours of simulation, signifying that two doses have already been administered to the patient). To achieve this, we employed a Random Forest (RF) Classifier that outputs a binary classification decision: whether or not the TCP of the current simulation will reach 100% or not. Each training example provided to the model is constructed using the following steps:

1. Select a window size that will serve as the length of the array of cancer cells.
2. Extract features from the time series data: We extract essential statistical measures such as the mean, median, minimum, and maximum values of the windowed array.
3. Provide this set of features to the RF model as a training example (along with the corresponding TCP).
4. Repeat the process with varying window sizes for each training example to ensure a diverse dataset and improve the model's robustness.

This methodology generates an abundant number of training examples (nearly a million) using only 2,600 simulations. The dataset is constructed based on a diverse range of simulations, incorporating varying cell cycle durations, radiosensitivities, and treatment types. To maintain balance in the dataset, we ensured an equal number of positive and negative samples.

The binary RF classifier has only a single parameter: it is composed of 1000 estimators. The specific choice of 1000 estimators is arbitrary but we want a high enough value. Indeed, we want to be able to produce a class probability. This probability is determined by calculating the mean predicted class probability across all trees within the forest. The class probability for each individual tree is computed as the ratio of samples belonging to the same class within a leaf.

Figure 7.2 showcases the Receiver Operating Characteristic (ROC) curve of the binary RF classifier. The Area Under the Curve (AUC) for this model is about 0.822, which indicates a good classification performance. A perfect model would have an AUC close to 1, suggesting an excellent measure of separability between the classes. In our specific use case, we aim to detect every single instance where the TCP is not 100%, in order to employ an alternative, a more aggressive agent.

Specificity, or the True Negative Rate, and Sensitivity, also known as the True Positive Rate, are key metrics in understanding the performance of our classifier. The ROC curve effectively visualizes the trade-off between these two measures.

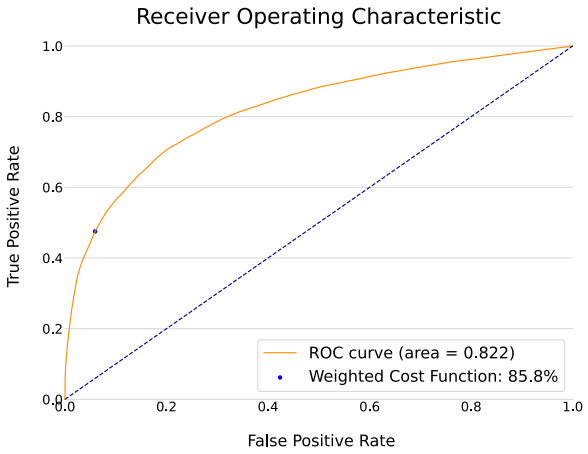


Figure 7.2: ROC curve of the binary RF classifier.

To achieve our goal, we must maximize specificity in order to capture all instances where the TCP is not 100%. However, it is essential to note that increasing specificity often results in reduced sensitivity, which may lead to more false negatives (i.e., predicting that TCP is not 100% when it indeed is).

By analyzing the ROC curve, we can determine an optimal threshold that strikes a balance between specificity and sensitivity. We use an approach that entails defining a weighted cost function with sensitivity and specificity. For our purpose, we assign different weights to specificity and sensitivity, such as a weight 3 times larger for specificity than sensitivity (e.g., 0.75 for specificity and 0.25 for sensitivity). Subsequently, for each point on the ROC curve, we calculate the weighted cost and find the threshold that minimizes this cost. Figure 7.2 also highlights this threshold. When the weight for specificity is 3 times larger than the weight for sensitivity, the resulting threshold is approximately 85.8%, meaning that if the predicted probability of an observation being positive (TCP of 100%) is greater than or equal to the threshold, it will be classified as positive; otherwise, it will be classified as negative (TCP different of 100%).

TCP regressor

Now that we have successfully predicted the outcome of the simulation from a binary perspective (i.e., whether the outcome is a TCP of 100% or not), we can take this a step further by attempting to predict the exact TCP value based on the same features. In doing so, we will transition from a binary classifier to a regressor that can predict the TCP of the simulation, taking into account the characteristics of the cancer cells. To achieve this, we will utilize a Random Forest Regressor (RFR).

The use of an RFR offers several advantages, including the ability to construct a confidence percentage around the decision of the regressor. This is achieved by utilizing each decision tree within the RFR to make individual predictions. Subsequently, we can calculate the number of trees that produce the same value as the RFR. For example, if all the decision trees arrive at the same answer, then the RFR is considered to have high confidence in its decision [2].

This approach was tested on a dataset made of 200 simulations. The RFR demonstrated outstanding performance, achieving an R2 score of over 0.99, which is considered excellent. This high R2 score indicates that the model is capable of accurately predicting the TCP values based on the cancer cell features, and the use of RFR provides an added layer of confidence in the predictions. The lecturer has to be aware that the high performances of the regressor is also due to the fact that not a lot of data has been used.

Threshold Quest

As previously mentioned, in order to effectively classify samples with our binary random forest classifier, we need to determine a suitable threshold for deciding the class membership. In section 7.2.1, we were given a mathematically-derived threshold to use. However, this method suffers from information loss during feature selection, which is a critical aspect that influences the quality of our classifier's output.

In our current approach, only cancer cells have been considered, despite the fact that other factors could potentially impact the classifier's decision. These factors may include healthy cells, the administered dosage, the time evolution of cancer and healthy cells, or even the influence of nutrients. Since it is impossible to accurately predict which factors will have the most significant impact on the simulation outcome, our feature selection process inevitably leads to a loss of information. Furthermore, feature extraction also contributes to this information loss, as the time evolution of the series and fluctuations are no longer present.

Consequently, using the threshold from section 7.2.1 may not accurately represent the simulation's behavior. Therefore, we will choose our thresholds empirically, based on the output of both the binary RF classifier and the RFR. This approach requires us to find two thresholds:

1. A threshold to determine if a more robust agent should be used or not. This threshold will be employed to compare the output of the binary RF classifier, deciding whether the value is above or below the threshold, and consequently, whether to employ a more robust agent.

[2] It does not provide the model uncertainty.

2. A second threshold will help us ascertain the aggressiveness of the next agent to be selected. We have already identified some robust agents, and our goal is to categorize them into three classes: slightly more aggressive (category I), moderately more aggressive (category II), and significantly more aggressive (category III). This threshold will indicate which class the agent should be selected from.

In practice, our classifier does not effectively account for the time dimension. As a result, it is unable to accurately process time series data, meaning that the model fails to capture the progression of events over time. To address this limitation, we need to adopt a more relaxed criterion for two main reasons:

1. In reality, it is not practical to conduct numerous tests on patients at frequent intervals, such as every hour or every few hours. Doing so would be invasive, time-consuming, and resource-intensive.

2. We cannot constantly change the treatment based on the classifier's recommendations. Once a treatment plan has been adjusted to a more robust approach, it is essential to allow sufficient time for evaluating the effectiveness of the new treatment. This evaluation process ensures that informed decisions can be made regarding any further adjustments.

To better accommodate these practical considerations, we will take into account the classifier's decisions every 48 hours. This timeframe ensures that patients receive two radiation doses before the treatment plan is reassessed. By doing so, we can make more informed decisions about the efficacy of the current treatment and any potential adjustments.

Benchmark environment with baseline treatment

This section is designed to offer insights into the performance of our binary RF classifier. We employ the benchmark environment (see Table 4.1), accompanied by the baseline treatment, as the context for our analysis. This approach allows us to understand the behavior of our classifier within a familiar environment and treatment scheme.

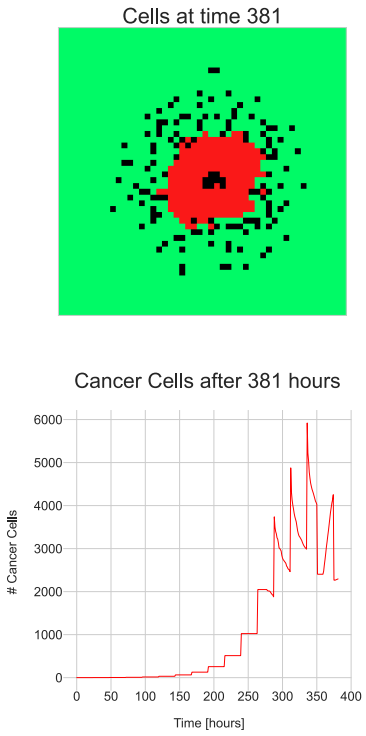


Figure 7.3: Benchmark simulation with baseline treatment.

Figure 7.3 depicts the benchmark simulation using the baseline treatment, which administers 2 Gy of radiation every 24 hours. By time 381, the patient has already received two doses of radiotherapy. The graph shows a significant reduction in the number of cancer cells at this point in time.

To predict the outcome of this simulation, we employ the binary random forest classifier at time 350, shortly after the initiation of radiotherapy. The classifier predicts that the TCP will reach 100% with a probability of 60.0%.

As previously mentioned, it is too early to determine whether we should select a new agent, given that the classifier has insufficient data after only 24 hours. As a result, we decide to wait for the second dose of radiotherapy, which is administered 48 hours after the treatment begins. Following the second dose, the binary random forest classifier predicts that the TCP will not achieve 100% with a probability of 98.9%. It is worth noting that the baseline treatment in the benchmark environment attains a mean TCP of 99%. This means that 99 out of 100 simulations, the baseline treatment will work for the benchmark environment. However, as the mean TCP is not 100%, the baseline treatment is considered to be failed for the benchmark environment. Thus, it is classified as a negative output by our classifier.

In this study, we adopt a stringent approach and regard even a 99% TCP as insufficient. Consequently, the classifier will produce a negative output (indicating that the simulation won't result in a TCP of 100%), prompting a potential change to a new agent, even if the current agent could be effective. This concept is discussed in section 7.2.1 concerning the sensitivity and specificity trade-off. However, we can mitigate the risk of changing the agent too hastily by examining the confidence level of the classifier's prediction. By establishing a threshold for this value, we can employ a more robust agent only when we have high confidence in the output. Nevertheless, this strategy contradicts our objective of detecting every commencement of a simulation that does not lead to a TCP of 100%, and the subsequent prediction is 48 hours later. This delay means the tumor could grow exponentially during this period and could become too large to effectively treat with a more robust agent.

18-hours simulation with baseline treatment

This section is dedicated to presenting the output from the binary random forest classifier in a simulation. The simulation is set within an environment that uses an 18-hour cell cycle, with the baseline treatment being maintained consistently throughout the entire duration of the simulation. No alterations are made to the treatment protocol, thus the focus of this section is solely on the analysis of the classifier's output.

Figure 7.4 illustrates a simulation in which the cancer cells have an 18-hour cell cycle duration, presenting a challenging environment for the baseline treatment. This scenario is known to be too difficult for the baseline treatment to effectively address. In fact, as discussed in section 6.3, we have observed that environments with a cell cycle duration (T_c) of 18 hours are particularly challenging for the Sarsa 17 agent. Consequently, the baseline treatment is anticipated to be unsuccessful in this environment.

As expected, the treatment outcome was indeed a failure, with 62 cancer cells remaining after the application of the baseline treatment. After administering the first radiation dose, the classifier's output indicated that the simulation would not achieve a TCP of 100% with a confidence level of approximately 72.6%. Following the second dose of radiation, the classifier's assessment shifted, suggesting that the treatment would succeed with a confidence level of 70.1%. Based on these results, a change in the current agent is recommended to increase the likelihood of success.

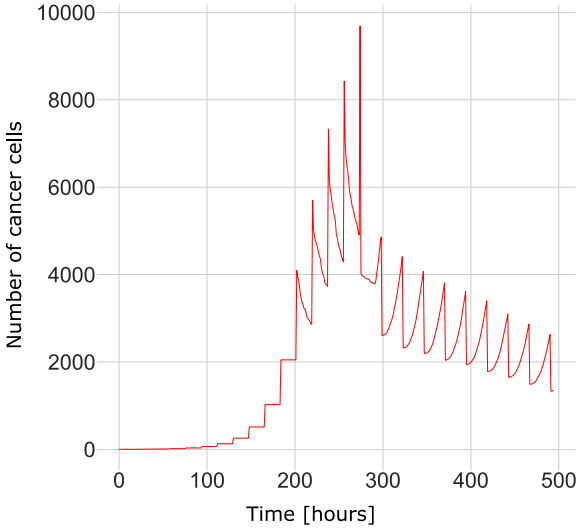


Figure 7.4: 18-hours Cell Cycle simulation with baseline treatment. Cancer cells number evolution over a period of 494 hours

After four doses of radiation therapy, the classifier once again expressed doubt regarding the treatment's ability to achieve a 100% TCP, with a confidence level of 80.5%. At this stage, it is advisable to change the agent. The challenge in this particular simulation lies in the fact that the environment is not overly aggressive, resulting in outputs that are not distinctly inclined towards either success or failure. In this case, while it is not absolutely necessary to replace the agent, there is no guarantee that a 100% TCP will be achieved with the current one.

Summary To wrap up our discussion on threshold selection, we have evaluated our classifier output within two distinct environments. The process of determining these thresholds draws upon insights gathered not just from these two scenarios, but also from other simulations that aren't detailed in this work. Nevertheless, the final decision regarding the threshold remains, to some extent, a subjective one. The thresholds are investigated even more, in Section 7.5.1.

7.3 Convolutional Neural Network Classifier

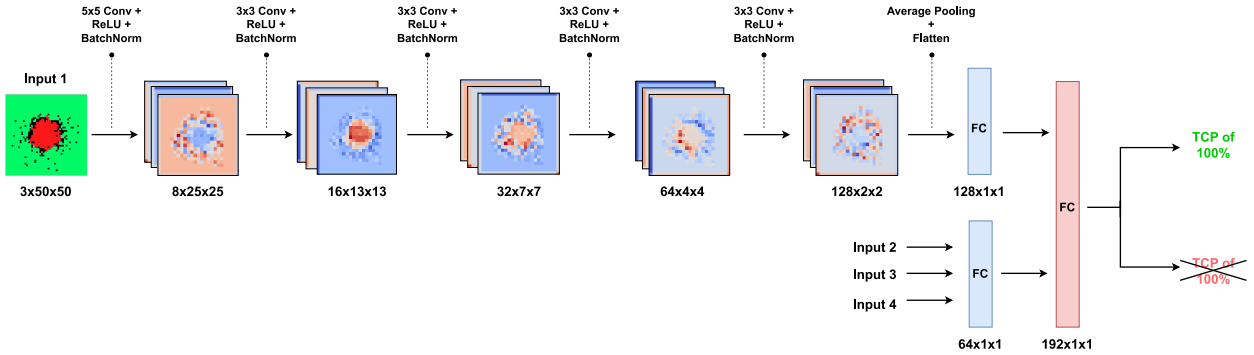


Figure 7.5: Convolutional Neural Network Classifier Architecture. FC blocks corresponds to fully connected layers.

Figure 7.5 illustrates the architecture of the Convolutional Neural Network (CNN) classifier. This classifier has four inputs: an image of the tumor, the hour of commencement of radiotherapy treatment, the last dose administered, and the total dose already administered. The reason behind using such a model is to provide a more realistic approach concerning the observations that can be made on a patient. Indeed, determining the number of cancer cells for a specific time interval (as discussed in section 7.2.1) is practically impossible, since we cannot track the number of cancer cells hourly for each patient. In fact, even estimating the number of cells within a tumor is a challenging task, which often relies on density estimation methods [1]. If we were able to accurately determine the number of cancer cells within a tumor, we would still need to interpolate the evolution between different observations. Therefore, developing a more realistic method to compute predictions based on accessible information is essential.

Our research involves training a CNN classifier on a multitude of simulations in the cellular environment. It's important to note that in our model, we do not account for time evolution. This stipulates that the training data samples originate from the same time instant. In simpler terms, we train one unique CNN for each specific time instant. For example, we accumulate multiple images of tumors post-administration of the first radiation dose. We then use this data to train a dedicated CNN. This procedure is repeated for the initial five doses, consequently resulting in the creation of five unique CNN binary classifiers, each corresponding to a specific dose.

Given the nature of this process, we anticipate that the accuracy of the classifier trained on data from the fifth dose will significantly surpass that of the CNN trained on the first dose data. This expectation arises from the assumption that as the simulation progresses, the ability to predict whether the simulation will culminate with a TCP of 100% improves. Therefore, with each successive dose, our model should exhibit enhanced predictive accuracy.

The effectiveness of the classifiers must increase in line with the number of doses given. This is primarily due to two reasons. Firstly, after the initial dose, the treatment might not yet exhibit its full effectiveness due to the inherent treatment time. Therefore, making a treatment change at this point might not be as critical. Secondly, the importance of making a treatment change escalates as the treatment progresses. This is why the Area Under the Curve (AUC), which measures the classifier's performance, should experience an upward trend as we examine images of tumors that have been exposed to treatment for an extended period. The AUC indicates the model's ability to distinguish between classes, with a higher AUC signifying a better model. Hence, as the treatment progresses, it becomes increasingly important to ensure that the classifier's AUC is improving, as this could directly affect the efficacy of the treatment.

Performances

Figure 7.6 illustrates the Receiver Operating Characteristic (ROC) curves for each of the CNN classifiers, corresponding to each dosage level. Observing the ROC curves, one can discern that the Area Under the Curve (AUC) incrementally escalates with the progression of the treatment duration. For instance, the AUC at the fifth dosage is markedly higher than that of the second dosage.

This is exactly the anticipated pattern because as the treatment persists, our aim is to incrementally enhance the accuracy of the simulation outcomes. Specifically, we aim to ascertain with greater precision whether the treatment will likely be successful or not. After the application of the second dosage, the AUC has already equaled the value obtained in section 7.2.1.

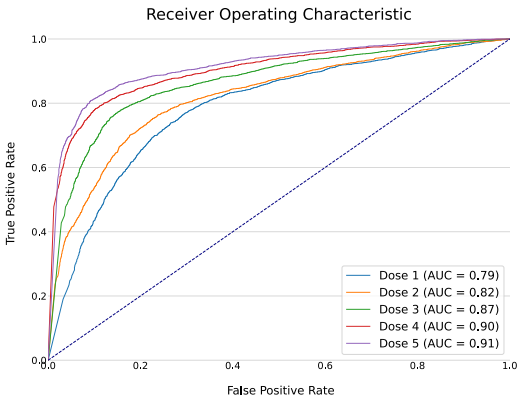


Figure 7.6: Receiver Operating Characteristics of the CNN classifiers.

The final three classifiers display near-optimal classification performances, reflecting an extremely high level of prediction accuracy. Depending on the timing of the observation, indicated by the tumor images, the classifier employed will differ, thus adjusting our confidence level in the prediction. Essentially, as we progress through the treatment and as the classifiers become more accurate, our confidence in their predictions will also increase.

However, it is crucial to note that the performance of the classifiers can be further augmented by integrating additional input data. While our current model solely utilizes tumor images as input, integrating other parameters such as cell density, oxygen concentration, and glucose concentration could potentially enhance the precision and reliability of the predictions. These parameters are known to have a direct influence on tumor growth and response to treatment, thus incorporating these into the CNN model could provide a more holistic understanding of the tumor's behavior and potentially improve the classifier's predictive ability.

Numerous architectures, as Inception [31] and ResNet [32], have been tested in our quest to perfect the task of outcome prediction. This is a really challenging task even for deep learning models, given the nature of the input data and the complexities of the classification output. The tumor's images are composed of just three colors, representing healthy cells, cancerous cells, or void areas within the grid. This simplicity belies the profound complexity underlying the task at hand. In real-world data, tumor images carry an array of additional information, detailing the shape of the tumor, its precise location within the human body, and numerous other potentially relevant factors. However, integrating such intricate real-world data is beyond the scope of our current project. On the other hand, the output for the binary classifier presents its own set of challenges. Conducting feature extraction to predict the outcome of a treatment is far from a straightforward task. Identifying which features of a tumor's image could be linked to the treatment outcome is highly complex.

Despite these challenges, CNNs demonstrate a unique prowess for detecting intricate patterns. Even with the simplicity of our input data and the complexity of the required output, CNNs have been able to discover significant correlations in this context. It's important to highlight that these results are achieved through an iterative learning process, where the models are continuously trained and refined. We are essentially teaching our CNNs to 'learn' from each image, each dosage, and each outcome, gradually enhancing their predictive capabilities over time.

CNN's Feature Extraction

It is important to note that the feature maps highlighted in Figure 7.5 do not represent the actual feature maps of the CNN, except for the first one, which appears after the initial convolution block composed of a convolutional layer, a ReLU activation function, and batch normalization. Only the first group of feature maps is interpretable for humans, as it represents the low-level understanding of our CNN. Essentially, these feature maps capture information from specific regions within the tumor image, which the CNN then utilizes to perform classification.

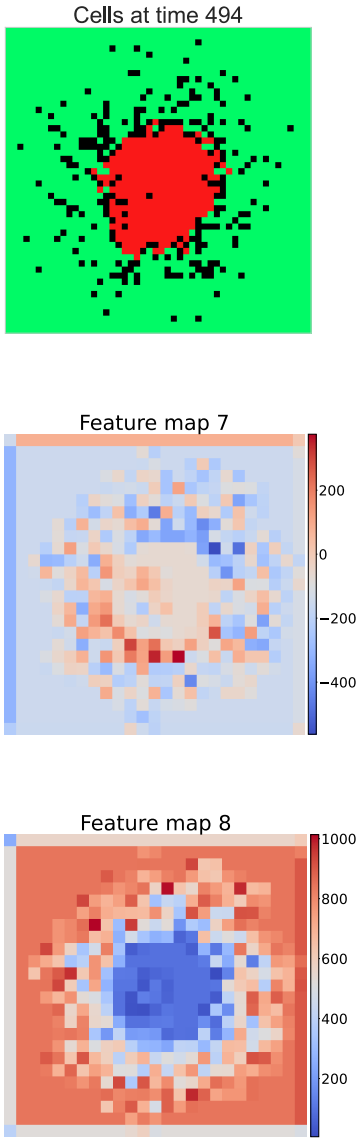


Figure 7.7: Original tumor image and feature maps from the CNN.

Figure 7.7 presents the initial image of the tumor alongside two distinct feature maps, which are associated with the inaugural convolution layer of the CNN, as illustrated in Figure 7.5. Feature maps are instrumental in shedding light on the process of feature extraction carried out autonomously by the CNN.

In the showcased architecture, the first layer (low-level) is comprehensible to humans. Essentially, this layer is designed to highlight the core information first. As we move up the layers, comprehension becomes more complex, transcending human understanding. These layers serve to attain a high level of comprehension. Here, the CNN is programmed to detect hidden patterns which are difficult to catch from a human perspective.

For instance, if we delve deeper into the feature maps presented, the seventh feature map places a greater emphasis on the center of the grid, which is where the tumor is situated. This suggests that this particular feature map is adept at identifying and highlighting abnormalities within cellular structures. Conversely, the eighth feature map draws attention to the periphery of the grid, a region predominantly inhabited by healthy cells. This feature map seems to specialize in recognizing and accentuating the normal, healthy cellular patterns.

It is worth noting, however, that it is inherently challenging to evaluate the relative importance of each feature map in the overall decision-making process of the CNN. The individual contribution of each feature map to the final decision made by the model remains obscure, as they combine and interact in complex ways that can be hard to trace.

One way to gain insight into this issue is through a method known as Gradient-weighted Class Activation Mapping (GradCam) [33]. This technique uses the gradients of any target concept, flowing into the final convolutional layer to produce a spatial map highlighting the important regions in the image for predicting the concept. However, a detailed discussion of GradCam is beyond the scope of this particular work.

7.3.1 Classifier comparison

This section is dedicated to make a brief comparison of the two classifiers presented in the previous sections. Thus, we will consider the binary random forest classifier and the binary convolutional neural network classifier to perform the task of predicting the outcome of the radiotherapy treatment.

Criterion	Binary Random Forest Classifier	Binary Convolutional Neural Network Classifier
Type of Data	The data utilized is the cancer cells' evolution through the simulation.	The data utilized is images of tumors. It is also possible to use cell density, oxygen concentration, and glucose concentration.
Feature Extraction	Feature extraction is performed on the time series of the cancer cells through the simulation. It characterizes a time series by its statistical properties as the mean, the median, the minimum, and the maximum.	Feature Extraction is performed by the CNN itself. It does not need any preprocessing of the data, except the normalization by the mean and the standard deviation on each batch of training.
Area Under Curve	The model achieved an AUC of 0.822	The different models achieved an AUC from 0.79 for the lowest performance, up to 0.91 for the best performance.
Behaviors of the data	The cancer cells' evolution can be discriminated between the successful and unsuccessful simulations.	No particular behavior or trend is observed in the data.

Table 7.1: Summary of classification tasks for prediction RT treatment outcome.

7.3.2 Conclusion

While the proposed approach demonstrates substantial promise, there are a number of limitations that require careful consideration and potential future investigation. Primarily, we have reached a plateau in the accuracy improvement of our current convolutional neural network model. Although a significant amount of data can be readily generated through simulations, it's clear that the quest for enhanced performance might necessitate a more intricate and comprehensive methodological approach.

Indeed, it may be beneficial to design a systematic pipeline that facilitates a deeper connection between the raw input data, comprised of tumor images, and the output of the classifier, which pertains to the treatment outcome. However, it's essential to clarify that this task of refining the model goes beyond the scope of our work. However, we have successfully accomplished satisfying performances to perform accurate prediction, forming a robust foundation that we can build upon in subsequent phases of this research.

In addition, the treatment response time, a crucial aspect in cancer therapy, has not been integrated into our model. Our current system, post-application of the radiation dose, instantaneously determines the death of cancer cells that satisfy the death criterion. This is a significant discrepancy from the actual biological process, as radiation therapy does not result in immediate cancer cell death.

In reality, the damage to DNA, which precipitates the demise of cancer cells, is not instantaneous. Rather, it is a gradual process that unfolds over the course of days or weeks, contingent on the intensity and duration of radiation therapy [34]. Post-treatment, the destruction of cancer cells continues to occur for weeks, sometimes even months, a phenomenon not accounted for in our current model.

Hence, the model's design potentially facilitates instantaneous decision-making following radiation dose administration, a scenario misaligned with real-world applications. This disjunction necessitates future exploration and appropriate modifications to our model, to more accurately reflect the time-delayed response of cancer cells to radiation therapy. Such modifications could involve the incorporation of temporal factors into the model or even the development of a time-series model, which might more closely mirror the biological reality of radiation treatment.

7.4 Agent robustness classes

In practice, testing the robustness of an agent can be quite time-consuming and resource-intensive to reach a precise conclusion about an agent's robustness. To properly assess robustness, we need to perform tests on each agent that has been constructed. A robustness test could involve exposing the agent to a wide variety of environments, and its ability to successfully solve these environments would provide insight into its robustness.

In this work, we will not perform tests on every single agent due to the aforementioned challenges. However, we need to select at least one agent per category to ensure we have a diverse representation, including a category I agent, a category II agent, and a category III agent. A priori, agents that have been trained only once (see Section 5.1.1) will belong to the category I. In this way, the selected agent Sarsa 17 belongs to this class. These agents consistently display impressive performance metrics within the benchmark environment that is inherently non-aggressive.

Now, we need to verify an assumption: radiosensitivity changes render the environment even more aggressive than cell cycle changes. This assumption is based on the fact that, in the previous section, we found agents capable of eradicating the tumor for cell cycle changes, even with 16-hour cell cycles, while we did not find any agent capable of doing so for very low radiosensitivities. This disparity might be because we made an arbitrary choice for radiosensitivities, which could be significantly different from reality, while we used relatively realistic values for cell cycle duration.

The most evident clue supporting this assumption is the presence of convergence among the cell cycle robust agents, whereas no such convergence exists for the low radiosensitivity agents. The convergence here is based on the TCP value. If an agent maintains a constant TCP over its training epochs, it can be considered to have converged. This convergence suggests that the agents are becoming more stable and reliable in their performance, indicating a higher level of robustness.

In this study, we've selected five unique artificial intelligence agents. These agents were initially the Sarsa 17 model (refer to section 6.1), but they've been retrained in different conditions: three of them in environments with varying durations of cell cycles (16, 18, and 20 hours), and two in environments with differing levels of radiosensitivity (0.7 and 0.8). Our goal is to see how each of these retrained agents performs when they're exposed to the conditions of the other agents' environments.

Here's an example to clarify this process: we first took an agent that was originally trained in an environment where the cell cycle lasted 20 hours. Next, we'll be testing this agent in various other environments - those with 16-hour and 18-hour cell cycles, and those with radiosensitivities of 0.7 and 0.8.

Likewise, we'll take another agent which was initially trained in an environment with a radiosensitivity level of 0.8. This agent will be tested in environments with cell cycle lengths of 16, 18, and 20 hours, and an environment with a radiosensitivity of 0.7.

As a reminder, the Sarsa 17 agent has been trained on an environment with a cell cycle of 24 hours and a radiosensitivity of 0.96875. The environments with modified radiosensitivity use a cell cycle of 24 hours and the environments with modified cell cycles use the default radiosensitivities, depicted in section 4.1.2.

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Cell Cycle of 16h	98.0	50.52	15.58	373.92	90.3
Cell Cycle of 18h	100.0	42.14	12.26	294.24	92.8
Radiosensitivity: 0.7	100.0	62.89	18.15	435.60	93.2
Radiosensitivity: 0.8	100.0	55.45	16.11	386.64	93.5

Table 7.2: Sarsa 17 agent retrained on an environment with a cell cycle duration of 20 hours. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.2 highlights the Sarsa 17 agent retrained on an environment with a cell cycle of 20 hours, assessed on various different environments. As expected, the Sarsa 17 agent has some difficulties to find the optimal treatment for environments with smaller cell cycle duration. However, surprisingly, it achieves a 100% of TCP on low radiosensitivity environments.

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Cell Cycle of 16h	100.0	51.72	15.68	376.32	95.2
Cell Cycle of 20h	100.0	40.06	12.44	298.56	92.9
Radiosensitivity: 0.7	100.0	68.26	20.96	503.04	94.7
Radiosensitivity: 0.8	100.0	55.84	17.01	408.24	93.9

Table 7.3: Sarsa 17 agent retrained on an environment with a cell cycle duration of 18 hours. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.3 highlights the Sarsa 17 agent retrained on an environment with a cell cycle of 18 hours, assessed on various different environments. The considered agent achieves perfect robustness against all types of environments.

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Cell Cycle of 18h	100.0	43.82	13.16	315.84	93.3
Cell Cycle of 20h	100.0	38.84	11.48	275.52	90.7
Radiosensitivity: 0.7	100.0	65.09	19.51	468.24	93.9
Radiosensitivity: 0.8	100.0	56.07	16.65	399.60	94.1

Table 7.4: Sarsa 17 agent retrained on an environment with a cell cycle duration of 16 hours. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.4 showcases the performance of the Sarsa 17 agent, retrained in an environment with a 16-hour cell cycle duration, when tested across a variety of environments. As anticipated from previous results, the TCP consistently hits a maximum of 100%, regardless of the environmental conditions. Interestingly, the agent, when retrained on a 20-hour cell cycle, demonstrates the highest performance in an environment with a radiosensitivity of 0.7 and 0.8.

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Cell Cycle of 16h	98.0	62.62	21.86	524.64	94.5
Cell Cycle of 18h	100.0	50.06	16.24	389.76	94.2
Cell Cycle of 20h	100.0	43.04	14.36	344.64	93.6
Radiosensitivity: 0.7	100.0	68.31	21.41	513.84	94.6

Table 7.5: Sarsa 17 agent retrained on an environment with a radiosensitivity of 0.8. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.5 showcases the Sarsa 17 agent that has been retrained in an environment possessing a radiosensitivity of 0.8, and subsequently evaluated across a diverse range of environments. It is particularly challenging for this retrained agent to effectively provide great performances in an environment where the cell cycle duration spans 16 hours. Interestingly, despite the difficulties, the agent displays a remarkable performance, achieving a TCP of 100% even in an environment that has a lower radiosensitivity of 0.7. The performances of this agent are similar to the one retrained on an environment with a cell cycle duration of 20 hours (see Table 7.2).

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Cell Cycle of 16h	100.0	56.06	18.94	454.56	89.9
Cell Cycle of 18h	100.0	52.08	17.70	424.80	90.4
Cell Cycle of 20h	100.0	44.58	15.56	373.44	90.0
Radiosensitivity: 0.8	100.0	64.49	21.35	512.40	93.2

Table 7.6: Sarsa 17 agent retrained on an environment with a radiosensitivity of 0.7. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.6 showcases the performance of the Sarsa 17 agent that has been retrained in an environment with a radiosensitivity of 0.7, evaluated across diverse conditions. This retrained agent exhibits a perfect TCP score of 100% irrespective of the environment type, mirroring the results achieved by an agent retrained in an environment with a cell cycle duration of 16 hours. However, upon examining other essential performance indicators, the agent retrained on a 16-hour cell cycle duration demonstrates superior overall performance.

7.4.1 Conclusion

In our evaluation of agent robustness, we classify the agents into three distinct categories: (1) slightly more aggressive, (2) moderately more aggressive, and (3) significantly more aggressive. The classifications are based on how the agents respond in different simulated environments. We define the slightly more aggressive category as category I, the moderately more aggressive as category II, and the significantly more aggressive category as category III.

The category III agent is identified as the one retrained on an environment where the cell cycle lasts for 16 hours. This agent achieves a higher level of performance, highlighted in the key performance indicators.

Next, we identify the category II agent. For this category, we have two contenders: the agent retrained in an environment with a 20-hour cell cycle and with a radiosensitivity of 0.8. The agent retrained on the 20-hour cell cycle exhibits superior performance in most respects, except survival rate. Given that both agents maintain a survival rate exceeding 90%, and considering that we're not factoring in any side effects, we deem the agent retrained on the 20-hour cell cycle as the reference for the category II.

Finally, we need to determine the agent that fits into the category I. For this, we select the agent mentioned in section 6.1, which was not retrained. This agent showcases commendable performance without displaying an overly aggressive behavior.

Having categorized each agent, we can proceed to establish an automatic robustness decision-making mechanism. This allows us to deploy our classifiers to autonomously make decisions, adjusting the level of agent aggressiveness as needed based on classifier predictions. In essence, we're building a self-regulating decision-making process that can adjust agent selection based on the analysis of our classifiers.

The reader should note that while we utilized specific criteria to categorize the robustness of each agent, these criteria were selected arbitrarily. This implies that there is a multitude of other possible methods and criteria that could be employed for this classification. The current analysis necessitated some simplifications. For instance, the first criterion we employed was the TCP, and the second was the amount of dose administered. We could have opted for other factors such as the standard deviation of key performance indicators, the number of states explored in the observation space, or even a more in-depth analysis of the correlation between cell cycle duration variation and radiosensitivity. Bear in mind that if the cell cycle duration fluctuates, the summation of radiosensitivities is likewise impacted due to the weighted sum, as detailed in Section 6.4.

Furthermore, we could have experimented with different types of agents. For instance, agents specifically trained on a particular environment could have been chosen. We might also have considered progressively trained agents (those trained multiple times in increasingly challenging environments).

7.5 Automatic Robustness Decision Making

This section aims to establish a comprehensive decision-making framework for determining whether to utilize a more robust agent based on observations during treatment. Initially, we'll illustrate a complete methodology derived from the output of the random forest classifier discussed in Section 7.2.1. Subsequently, we'll modify this approach slightly to accommodate the output from the convolutional neural network classifier. Lastly, we will contrast the results derived from these two classification algorithms, both against each other and against findings from previous sections.

Figure 7.8 illustrates the framework designed for automatic robustness decision-making. This framework is primarily based on a standard reinforcement learning process where an agent interacts with the environment by taking certain actions in response to received observations. In this context, the actions correspond to administered radiation doses while the observations consist of the count of healthy and cancer cells resulting from the previous action taken.

An additional layer of complexity is added with the incorporation of a classifier component. This component processes input in the form of tumor images or time-series data of cancer cells, depending on whether a random forest classifier or the convolutional neural network classifier is employed. It subsequently generates a binary output, which represents a prediction regarding the success of the simulation. This output, serving as a pivotal piece of information, is then channeled into the RL agent. This allows for an adaptive decision-making process wherein the RL agent, informed by the output from the classifier, can transition to a more robust model if required.

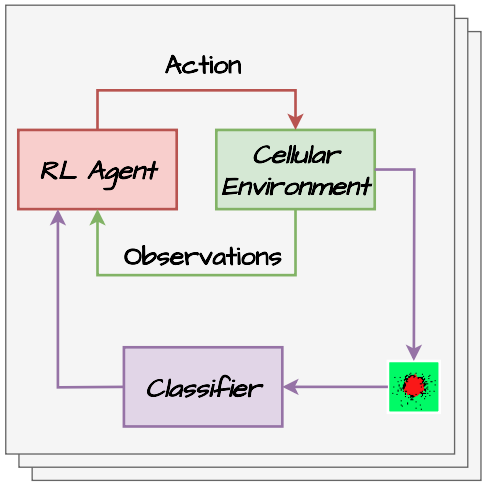


Figure 7.8: Automatic Robustness Decision Making framework.

7.5.1 Methodology

As mentioned before, this section is dedicated to determining the decision-making process based on the random forest classifier (see Section 7.2.1). We will end this section by extending the process to the convolutional neural network classifier.

In the Section 7.2.1, we observed that the benchmark simulation did not provide sufficient information regarding the thresholds that need to be established. As a result, multiple simulations were conducted in various environments. It is worth noting that the more challenging the environment, the easier it is to make a decision. This is because it becomes more apparent that the treatment will fail earlier in aggressive environments. As a reminder, we need to determine two thresholds: one to decide if a more robust agent should be used, and another to determine the level of robustness for the new agent.

These two thresholds were empirically determined. The threshold to decide if a more robust agent should be used is set to 59% and the threshold to determine the level of robustness of the new agent is set to 85%. The framework automatically checks, by using the random forest classifier, whether the simulation will end up with a 100% TCP, after every two radiotherapy doses. This approach avoids checking the simulation at every single hour, which would be impractical for a real-time application testing since it would require too much infrastructure, time, and qualified personnel.

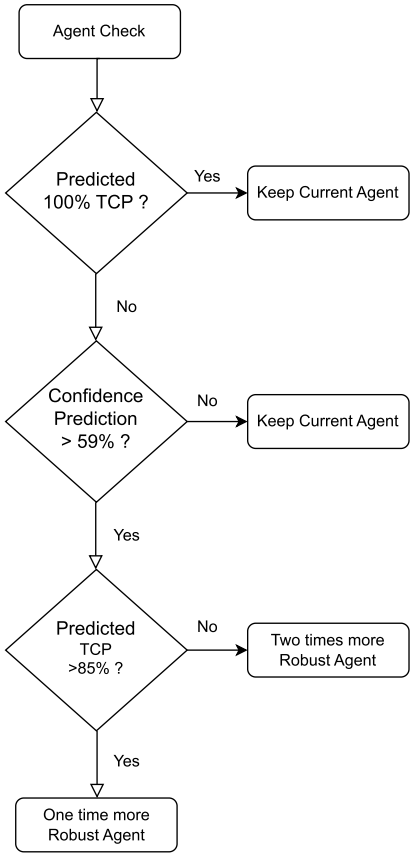


Figure 7.9: Automatic Robustness Decision Making.

After two doses, the first test involves evaluating the prediction of our binary random forest classifier. If the classifier's output indicates that the simulation will reach a TCP of 100%, no changes are made to the agent. However, if the prediction suggests that the simulation will not achieve a TCP of 100%, we then assess the confidence of this binary classification [3]. If the confidence is below 59%, the simulation continues with the current agent. If the confidence is above 59%, one final check is performed using the predicted TCP from the Random Forest Regression (RFR).

As previously mentioned, potential replacement agents are divided into three categories: category I, category II, and category III. The RFR prediction helps determine which category to select a new agent from. If the predicted TCP is below 85%, we choose an agent from a category that is two times stronger than the current one. For instance, if the simulation starts with a treatment not belonging to any of the three categories (assuming we do not begin with overly aggressive treatments), and the binary classifier suggests, with a confidence of over 59%, that we should change the current agent, we then consult the predicted TCP. If the predicted TCP is below 85%, we directly select an agent from the category II. If the predicted TCP is above 85%, the selected agent will be chosen from the category I. Another example, if the current agent is already under the category I and the binary classifier advises a change, the same process is followed. If the predicted TCP is below 85%, we choose an agent from the category III, and if not, we select an agent from the category II.

We can design an analogous decision-making process using the Convolutional Neural Network (CNN). The distinguishing factor here is the absence of a confidence percentage about our classification and a predicted TCP. To address this, we arbitrarily set them to 100% at all times. This implies that if the CNN classifier's output indicates that the simulation will be unsuccessful, we immediately deploy an agent that is one time more robust than the current one. However, this simplification in the decision-making architecture can potentially lead to suboptimal results. For instance, it could lead to treatment where each dose has a high value, meaning that more side effects could occur.

7.5.2 Results

In this section, we will meticulously analyze the results gathered from our automatic decision-making architecture. The architecture begins with the baseline treatment, updating the treatment as necessary based on its decision scheme. It will selectively integrate more robust agents that we have previously defined into the three aforementioned categories. These agents are re-trained version of Sarsa 17 agent for specific environments (refer to Section 7.4).

We aim to contrast the outcomes of our specifically designed architecture with the performance of the most successful agent in each environment. For this purpose, we will consistently select the agent with the highest performance as the standard for comparison. These agents can be of different origins: (1) It could be a directly Sarsa agent on the corresponding environment, (2) it can be the retrained Sarsa 17 agent for the respective environment, or (3) it might be an agent that has demonstrated superior performances than the previous two types, as we discovered in Section 7.4.

[3]Section 7.2.1 explains how the confidence of the classification is constructed. This confidence percentage is only available with the random forest classifier.

Our comparison extends to the automatic robustness decision-making framework, which employs both the random forest classifier and the convolutional neural network classifier. To ensure statistical accuracy, we subjected each environment to our architecture a hundred times. This approach enables us to report the mean and standard deviation results with confidence.

Furthermore, we will quantify the improvement offered by our architecture compared to the selected agent using a defined percentage, which is calculated as follows:

$$\delta = -100 \times \left(1 - \frac{KPI_{ARA}}{KPI_{SEL}}\right) \tag{7.1}$$

In this equation, δ symbolizes the percentage of improvement. The term KPI_{ARA} refers to the mean value of the key performance indicator (such as the total dose, the number of fractions, or the treatment duration) for the Auto-Robust Agent (ARA). KPI_{SEL} denotes the mean value of the key performance indicator for the selected agent (i.e., the agent that has demonstrated superior performance in a given environment).

Interpretation of δ is straightforward: a negative value indicates that our automatic decision-making architecture has outperformed the selected agent. Conversely, a positive δ signifies that the selected agent has maintained its superior performance. Essentially, our goal is to minimize the key performance indicator as much as possible. The values of δ will be in the following tables, next to the value of the key performance indicator.

We plan to include the mean value of δ across the key performance indicators. We provide the average value of δ over the metrics of Dose, Fractions, and Duration. We'll also compare the automatic decision-making architecture results with the baseline treatment, represented as an average δ value. Section 9.3.2 provides an in-depth analysis of these results and their implications in comparison to the baseline. Similarly to the δ value, the mean of δ has the same signification: a negative value indicates that our automatic decision-making architecture provided better overall performances.

We will also incorporate visualizations of the treatment patterns to facilitate a better understanding of our agents' behavior. These will involve a comparative analysis of the chosen agent's treatment pattern against that of the ARA agents. All the comparisons of the treatment patterns can be found in Section 9.3.1 in the annex.

Figure 7.10 provides a visualization of the treatment strategies that will be encountered in the next sections. The best-performing agent, represented in grey, and the strategies of the auto-robust agent (ARA), represented in blue, are clearly depicted. The graph displays the dose delivered across the duration of treatment, with the mean dose value pinpointed by a dot, and an adjoining bar that indicates the standard deviation.

As stated earlier, the ARA agent consistently initiates with the baseline treatment, resulting in the first two radiation doses being fixed at 2 Gy. Consequently, the mean dose value is positioned at 2 Gy, without any corresponding standard deviation, due to this uniformity in initial treatment.

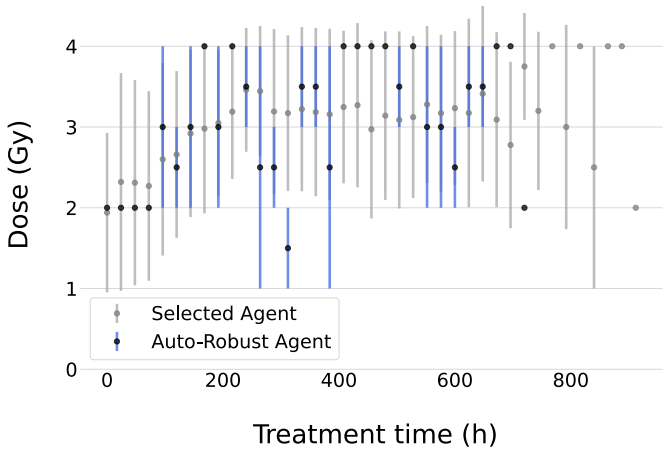


Figure 7.10: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.5.

Radiosensitivity of 0.8

This section describes the results obtained for simulations based on an environment with a radiosensitivity of 0.8. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is the Sarsa 17 agent retrained on an environment with a cell cycle duration of 20 hours. This agent demonstrated the best performances on this environment as explained in section 7.4.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Cell Cycle 20 hours	100.0	55.45 (-)	16.11 (-)	386.64 (-)	93.5
Auto-Robust Agent (RF)	100.0	59.54 (+7.38%)	18.25 (+13.3%)	438.0 (+13.3%)	93.7
Auto-Robust Agent (CNN)	100.0	62.8 (+13.3%)	20.7 (+28.5%)	495.8 (+28.2%)	93.7

Table 7.7: Best results and improvements for an environment with a radiosensitivity of 0.8. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.7 succinctly showcases the resultant outcomes. The "Cell Cycle 20 hours" agent pertains to a model that was initially trained in a benchmark environment, and subsequently, in an environment characterized by a cell cycle of 20 hours. As demonstrated in Section 7.4, this agent, when subjected to an environment with a radiosensitivity value of 0.8, managed to achieve optimal performance. Furthermore, the insights garnered from Table 7.7 underscore a noteworthy point: the automated decision-making framework, irrespective of the employment of the Random Forest or CNN classifier, falls short in surpassing this level of performance.

The mean improvement over the key performance indicators [4] with the CNN classifier compared to the selected agent is +23.3% [5]. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -50.3% (see Section 9.3.2).

Radiosensitivity of 0.7

This section describes the results obtained for simulations based on an environment with a radiosensitivity of 0.7. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is a Sarsa agent directly trained on an environment with a radiosensitivity of 0.7, as it demonstrated the best performances on this environment.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Sarsa directly trained	100.0	72.31 (-)	23.27 (-)	558.5 (-)	93.4
Auto-Robust Agent (RF)	100.0	61.26 (-15.3%)	18.96 (-18.5%)	455.0 (-18.5%)	93.9
Auto-Robust Agent (CNN)	100.0	64.7 (-11.1%)	20.9 (-10.1%)	500.9 (-10.3%)	93.4

Table 7.8: Best results and improvements for an environment with a radiosensitivity of 0.7. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.8 effectively delineates the gathered results. The chosen agent, which was explicitly trained within an environment exhibiting a radiosensitivity of 0.7, emerged as the one with the highest overall performance. Remarkably, the automated decision-making framework managed to outstrip the performance of this agent.

[4] We compute the mean over the key performance indicators to have a single percentage representing the improvements. For instance, with respect to the environment with a radiosensitivity of 0.8, we obtain: $\frac{13.3+28.5+28.2}{3} = 23.3\%$

[5] As the value of δ, a positive percentage means that our automatic decision-making architecture underperformed compared to the selected agent.

In specific terms, the auto-robust agent deploying the Random Forest classifier demonstrated a mean performance boost of -17.4% over the key performance indicators. On the other hand, the auto-robust agent that utilized the CNN classifier achieved a more modest mean increase of -10.5%. Yet, it is important to note that both agents maintained a survival rate commensurate with that of the selected agent.

Figure 7.11 illustrates the radiotherapy treatment strategies adopted for the selected agent in comparison to the ARA agent using the CNN classifier. A distinct improvement can be observed in terms of treatment duration, with the ARA's protocol demonstrating a noticeably shorter span.

The initial phase of the ARA treatment reveals a certain level of volatility in its approach, indicated by a substantial standard deviation across each radiation dose up to the 384-hour mark. This suggests an unpredictable or adaptable dosing strategy during this phase. However, a sharp contrast is observed beyond this point, where five doses of 4 Gy are delivered with no standard deviation. This means that across 100 simulations, the ARA agent consistently selected the same dose at these particular time intervals.

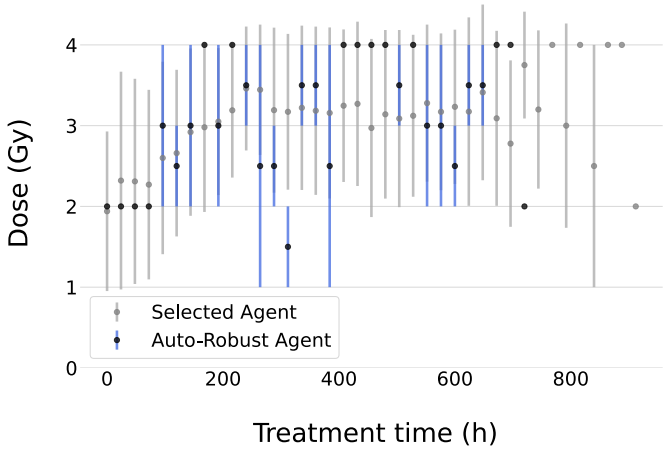


Figure 7.11: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.7. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

Our hypothesis is that this signifies the agent's attempt to escalate the treatment's intensity. We infer that the initial 384 hours were dedicated to administering smaller, possibly safer doses, with a subsequent shift towards a more aggressive treatment approach, as indicated by the consistent high-dose delivery.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is -10.5%. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -51.0% (see Section 9.3.2).

Radiosensitivity of 0.6

This section describes the results obtained for simulations based on an environment with a radiosensitivity of 0.6. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is the Sarsa 17 agent retrained on an environment with a radiosensitivity of 0.6, as it demonstrated the best performances on this environment.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Selected agent retrained	88.0	109.8 (-)	37.8 (-)	906.2 (-)	89.7
Auto-Robust Agent (RF)	100.0	85.1 (-22.5%)	26.2 (-30.6%)	628.8 (-30.6%)	93.9
Auto-Robust Agent (CNN)	100.0	90.0 (-18.0%)	28.3 (-25.1%)	678.5 (-25.1%)	94.2

Table 7.9: Best results and improvements for an environment with a radiosensitivity of 0.6. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.9 presents the results derived from simulations set in an environment with a radiosensitivity of 0.6,

a challenging setting in which no single agent has yet attained a perfect 100% TCP. Interestingly, in this context, the automated decision-making architecture reaches the elusive 100% TCP, leading to significant enhancements in performance. As a natural consequence of achieving a TCP of 100%, there is a subsequent increase in the survival rate. All the other key performance indicators have also been improved thanks to the automated architecture.

Figure 7.11 shows the treatment strategies employed by the selected agent and the ARA agent, which uses the CNN classifier. The ARA agent notably applies a treatment that successfully achieves 100% TCP within an environment demonstrating a radiosensitivity of 0.6. A striking feature of this treatment is the requirement of high-dose radiation in the initial phase up to 432 hours. Post this point, there is a switch to the application of doses of more moderate intensity on average. Unexpectedly, certain dosages towards the end of the treatment appear to be obligatory. It is deduced based on the absence of standard deviation in the last four radiation doses, indicating their uniformity. Figure 7.11 underscores the fact that a treatment approach involving incrementally increasing radiation doses is not apt for environments with extremely low radiosensitivity.

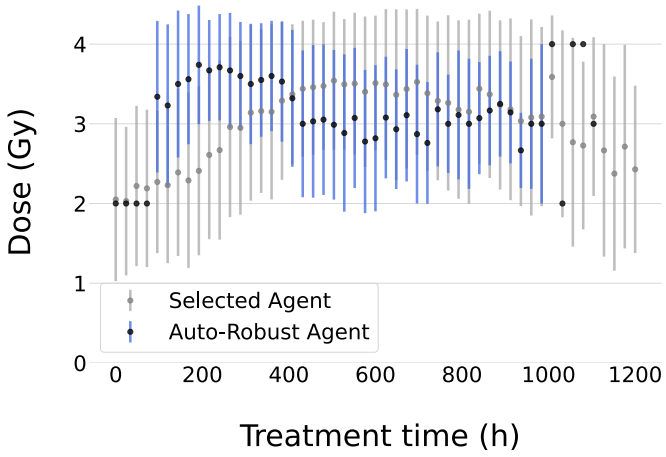


Figure 7.12: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.6. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is -22.7%. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -33.2% (see Section 9.3.2).

Radiosensitivity of 0.5

This section describes the results obtained for simulations based on an environment with a radiosensitivity of 0.5. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is the Sarsa 17 agent retrained on an environment with a radiosensitivity of 0.5, as it demonstrated the best performances on this environment.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Selected agent retrained	60.0	133.24 (-)	45.4 (-)	1089.6 (-)	78.1
Auto-Robust Agent (RF)	100.0	113.7 (-14.7%)	34.7 (-23.6%)	832.1 (-23.6%)	93.5
Auto-Robust Agent (CNN)	100.0	125.5 (-5.8%)	40.2 (-11.5%)	965.5 (-11.4%)	90.9

Table 7.10: Best results and improvements for an environment with a radiosensitivity of 0.5. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.10 presents the findings from environments characterized by a radiosensitivity of 0.5, indicative of an extreme environment. Much like the preceding section, no agent managed to realize a TCP of 100% within such extreme conditions. Yet, the automated decision-making architecture surmounts this challenge, successfully achieving a TCP of 100%. Nonetheless, the severity of this environment results in a substantial

increase in the total dose administered. Every administered dose averages around 3 Gy, marking a rather aggressive approach to treatment. It's important to underscore that such an intensive treatment regimen might precipitate numerous unwanted side effects.

The comparison of the treatment strategies can be found in Section 9.3.1, in the annex. Similar results are observed to environments with a radiosensitivity of 0.6.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is -9.57%. It is not representative since the selected agent do not achieve a TCP of 100%.

Cell cycle duration of 24 hours

This section describes the results obtained for simulations based on an environment with a cell cycle duration of 24 hours (benchmark environment). We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is the Sarsa 17 agent, as it demonstrated the best performances on this environment.

	TCP (%)	**Dose (Gy)**	**Fractions (-)**	**Duration (h)**	**Survival (%)**
Selected agent (Sarsa 17)	100.0	41.8	15.28 (-)	366.7 (-)	96.4
Auto-Robust Agent (RF)	100.0	42.6 (+1.91%)	15.4 (+0.79%)	369.6 (+0.79%)	93.7
Auto-Robust Agent (CNN)	100.0	40.1 (-4.07%)	13.7 (-10.3%)	327.6 (-10.7%)	93.3

Table 7.11: Best results and improvements for an environment with a cell cycle of 24 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.11 shows the findings acquired from the benchmark environments, deploying the Sarsa 17 agent as chosen in Section 6.1. While the automatic decision-making system using the RF classifier fails to yield any significant enhancements, the architecture employing the CNN classifier indicates an average improvement of -8.36%. Even though there is a slight decrease in the survival rate, the resultant figure is deemed acceptable as it surpasses the 90% threshold, as discussed in Section 5.1.2.

After just 96 hours of treatment, the ARA utilizing the RF classifier has already identified the most robust agent from the category III. This suggests that it is making swift decisions regarding the projected outcomes of the simulation. Conversely, the ARA employing the CNN classifier identifies this agent only after 192 hours of radiotherapy treatment. This indicates a more cautious decision-making approach when considering transitioning to a more robust agent. This could be seen as a more desirable approach as it could potentially mitigate the aggressiveness of the treatment on the patient. Therefore, we encounter a trade-off between reducing the intensity of the treatment, thereby minimizing exposure to high radiation doses, and achieving the optimal therapeutic outcome, a 100% TCP.

Figure 7.13 illustrates the comparative treatment strategies devised by the chosen agent (Sarsa 17) and our ARA agent, utilizing the CNN classifier. As explained before, the ARA agent invariably starts with the baseline treatment for the initial two radiation doses and can choose a more robust agent every two doses.

As depicted in Figure 7.13, the opening four radiation doses are consistently set at 2 Gy. The Sarsa 17 agent, on the other hand, initiates treatment with radiation doses exceeding 3 Gy. Interestingly, although the ARA agent begins with conventional dosages, it gradually escalates to higher dosages during the remainder of the treatment, surpassing those of the Sarsa 17 agent. A striking pattern emerges in the ARA treatment strategy, where the last three doses are invariably identical.

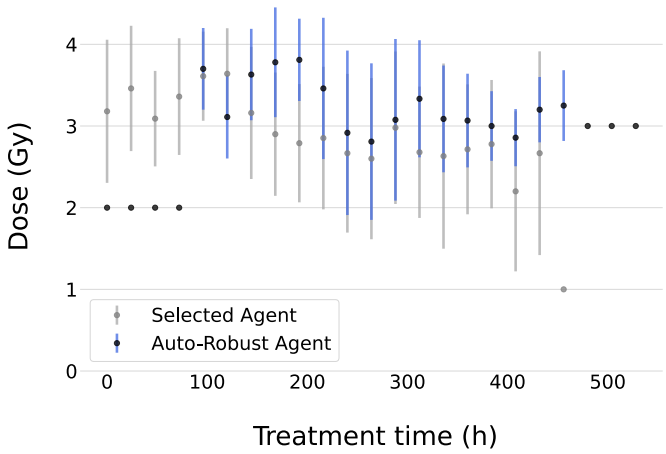


Figure 7.13: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for the benchmark environment. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is -8.36%. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -55.0% (see Section 9.3.2).

Cell cycle duration of 20 hours

This section describes the results obtained for simulations based on an environment with a cell cycle duration of 20 hours. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is the Sarsa 17 agent retrained on an environment with a cell cycle duration of 20 hours, as it demonstrated the best performances on this environment.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Selected agent retrained	100.0	41.8 (-)	12.06 (-)	289.44 (-)	91.4
Auto-Robust Agent (RF)	100.0	47.9 (+14.6%)	16.2 (+34.3%)	387.8 (+33.9%)	91.3
Auto-Robust Agent (CNN)	100.0	46.8 (+12.0%)	16.1 (+33.5%)	386.6 (+33.6%)	90.6

Table 7.12: Best results and improvements for an environment with a cell cycle of 20 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.12 discloses the results procured from an environment with a cell cycle duration of 20 hours. As illuminated in the previous section, the automated decision-making paradigm does not surpass the performance metrics of the selected agent. The chosen agent exhibits stellar performance, thereby setting a challenging benchmark to exceed.

Figure 7.14 shows the comparison between the treatment strategies of the selected agent and the ARA agent. From these simulation behaviors, we can extract significant insights. Notably, the importance of the initial three radiation doses in environments with modified cell cycle durations. These doses necessitate being of high value. The selected agent demonstrably adheres to this principle, consistently opting for significantly high-value radiation doses at the commencement of the treatment.

Moreover, an intriguing pattern emerges following the first three radiation doses. It appears that the standard deviation experiences an increase for subsequent doses. This observation underscores the relatively lesser importance of the mid-treatment radiation doses in the context of the entire simulation.

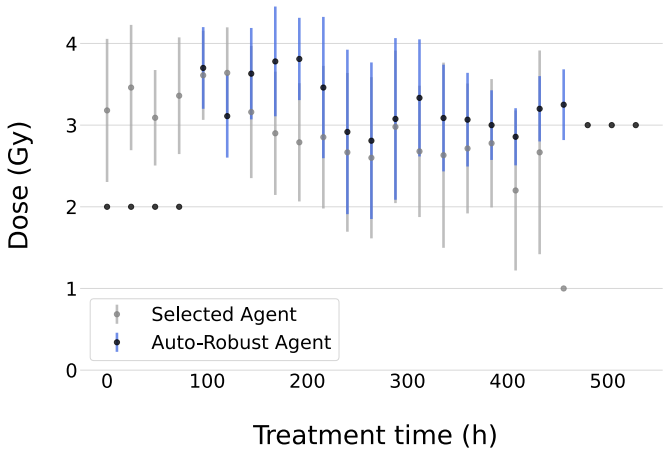


Figure 7.14: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for an environment with a cell cycle duration of 20 hours. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

As depicted in Figure 7.14, the ARA agent adopts a less aggressive treatment approach. However, it compensates for this by extending the duration of the treatment, hence underscoring a strategic difference between the two agents.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is +26.4%. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -62.3% (see Section 9.3.2).

Cell cycle duration of 18 hours

This section describes the results obtained for simulations based on an environment with a cell cycle duration of 18 hours. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is the Sarsa 17 agent retrained on an environment with a cell cycle duration of 18 hours, as it demonstrated the best performances on this environment.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Selected agent retrained	100.0	46.64 (-)	13.66 (-)	327.84 (-)	91.5
Auto-Robust Agent (RF)	100.0	47.0 (+0.77%)	15.0 (+9.81%)	360.2 (+9.81%)	90.7
Auto-Robust Agent (CNN)	100.0	44.9 (-3.73%)	15.7 (+14.9%)	375.6 (+14.6%)	90.4

Table 7.13: Best results and improvements for an environment with a cell cycle of 18 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.13 outlines the results obtained from an environment with an 18-hour cell cycle duration. In line with the observations from prior sections, the automated decision-making mechanism fails to outperform the selected agent.

The comparison of the treatment strategies can be found in Section 9.3.1, in the annex. Similar results are

observed to environments with a cell cycle duration of 20 hours.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is +8.59%. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -59.0% (see Section 9.3.2).

Cell cycle duration of 16 hours

This section describes the results obtained for simulations based on an environment with a cell cycle duration of 16 hours. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process. The selected agent for this section is a Sarsa agent trained directly on an environment with a cell cycle duration of 16 hours, as it demonstrated the best performances on this environment.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Sarsa directly trained	100.0	46.75 (-)	13.88 (-)	333.1 (-)	91.5
Auto-Robust Agent (RF)	100.0	44.53 (-4.7%)	14.25 (+2.7%)	342.0 (+2.7%)	91.0
Auto-Robust Agent (CNN)	100.0	48.9 (+4.6%)	16.7 (+2.0%)	400.1 (+20.1%)	88.9

Table 7.14: Best results and improvements for an environment with a cell cycle of 16 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Table 7.14 showcases the results obtained from an environment characterized by a 16-hour cell cycle. As anticipated, the automated decision-making system does not surpass the performance of the selected agent, even though it comes relatively close.

The comparison of the treatment strategies can be found in Section 9.3.1, in the annex. Similar results are observed to environments with a cell cycle duration of 20 hours.

The mean improvement over the key performance indicators with the CNN classifier compared to the selected agent is +8.90%. The mean improvement over the key performance indicators with the CNN classifier compared to the baseline treatment is -50.3% (see Section 9.3.2).

7.5.3 Summary and discussion

Summary

The collected data across various environments, each characterized by unique attributes like differing radiosensitivity levels and cell cycle durations, were encapsulated in tables 7.7, 7.8, 7.9, 7.10, 7.11, 7.12, 7.13, and 7.14.

In environments with radiosensitivity levels of 0.8 and higher, a model trained on a benchmark environment and then a 20-hour cell cycle environment yielded optimal performance. Automated decision-making architecture, despite its use of Random Forest or CNN classifiers, was unable to achieve superior performance in these settings.

However, in an environment with radiosensitivity of 0.7, the automated decision-making architecture outperformed the agent explicitly trained within this environment. The auto-robust agents deploying Random Forest and CNN classifiers demonstrated mean performance boosts of -17.4% and -10.5% respectively compared to the selected agent, whilst maintaining comparable survival rates.

In a challenging setting with a radiosensitivity of 0.6, where no agent had previously achieved a 100% TCP, the automated decision-making framework managed to attain this elusive goal, subsequently resulting in an increase in the survival rate.

Under extreme conditions of 0.5 radiosensitivity, the automated architecture once again reached a TCP of 100%. However, due to the severity of the environment, the total dose administered significantly increased, with an average dose of 3 Gy, which may lead to potential side effects due to its intensity.

In the benchmark environments, using the Sarsa 17 agent, the automated decision-making architecture employing the CNN classifier indicated an average improvement of 8.36%, though with a slightly reduced survival rate, but still considered acceptable as it surpassed the 90% threshold.

	Radio of 0.8	**Radio of 0.7**	**Radio of 0.6**	**Cell cycle of 24h**	**Cell cycle of 20h**	**Cell cycle of 18h**	**Cell cycle of 16h**
Baseline	-52.4%	-51.0%	-33.2%	-55.0%	-62.3%	-59.0%	-50.3%
Selected Agent	+23.3%	-10.5%	-22.7%	-8.36%	+26.4%	+8.59%	+8.90%

Table 7.15: Mean delta value representing the improvement of the ARA agent using the CNN classifier compared to the baseline treatment and the selected agent on modified cellular environments. The selected agent corresponds to the most suitable agent for each environment. These agents have been presented in the previous sections. A negative percentage indicates that the ARA agent outperforms the baseline or the selected agent.

Table 7.15 provides the mean δ values obtained with the ARA agent using the CNN classifier compared to the baseline treatment and the selected agents depicted in the previous sections. The ARA agent outperforms in every single type of environment the baseline treatment.

Discussion

As outlined in Section 6.5 and in Section 7.5.2, environments characterized by modified cell cycle duration necessitate high-dosage treatment in the early stages. Indeed, the most effective approach for managing this type of tumor is the immediate application of high-dosage treatments within the initial hours.

This scenario reveals a limitation of our current automatic decision-making architecture, which fails to outperform the selected agent. The reason for this is the use of the baseline treatment in the early stages, which invariably employs radiation doses of only 2 Gy. This approach demonstrates a constraint in our system as we often must wait to gather relevant observations before deciding to transition to a more intensive treatment strategy. This is the main reason for the suboptimal performances provided by our architecture on modified cell cycle environments compared to the selected agents.

Conversely, as illustrated in Section 6.5, environments exhibiting modified radiosensitivity do not necessitate the early application of high-dosage treatments. Consequently, this provides an opportunity for our automatic decision-making architecture to acclimate to such environments. Even if this type of environment does not specifically requires high-dosage treatment in the early stage, we observe an improvement in the key performance indicators using a dose bigger than 2.5 Gy during the first 400 hours of treatment. This type of environment also allows the use of a large variety of doses in the middle of treatment between 2 and 4 Gy.

7.5.4 Ways of improvements

As delineated in Section 7.3.2, there is significant potential for augmenting the efficiency of the classifiers. Our exploration has covered various CNN architectures, nevertheless, it is plausible that a superior structure could be formulated for this model. Additional research and exploration are required to harness more effectively the raw data at our disposal.

One pivotal aspect to consider would be the integration of real-world data, such as CT scans or MRI images of tumors. A potentially groundbreaking approach could involve the use of multiple tumor images for decision-

making. However, this would necessitate the modification and construction of a specific CNN architecture tailored to this task.

Furthermore, the automatic robustness decision-making process offers considerable scope for enhancement. The current decision scheme is quite simple and may not be generating the most optimal results. Nonetheless, we have seen substantial improvements over the chosen agents in several environments.

Figures 7.15 and 7.16 present histograms that elucidate the decision-making process regarding the utilization of a more robust agent. To summarize, each environment underwent 100 simulations. The specific environment type is indicated in the legend.

We have two distinct plots: one pertaining to the cell cycle environments (left) and another concerning the radiosensitivity environments (right). Both plots demonstrate the initial decision to be made (i.e., 48 hours post-treatment initiation) following the administration of two radiation doses. As previously outlined, there are three possible decisions: selecting an agent with one-time increased robustness, opting for an agent with two-time increased robustness, or retaining the current agent (refer to Section 7.5.1). Each simulation commences with the baseline treatment as the first agent. The automatic decision-making process uses the random forest classifier for the results presented in Figure 7.15 and Figure 7.16 because, as explained earlier, the CNN will always select a one-time enhanced agent (see Section 7.5.1).

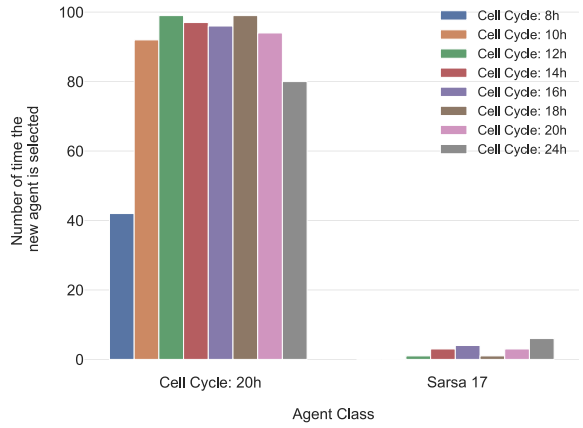


Figure 7.15: Histogram of the decision about the agent change for cell cycle environments.

Figure 7.16: Histogram of the decision about the agent change for radiosensitivity environments.

Figure 7.15 and Figure 7.16 show the histogram of the decision taken by our automatic decision-making architecture after the first radiation dose. 100 simulations have been conducted for each environment to see the decision taken by our architecture. After the first radiation dose, the automatic decision-making process has the choice of selecting an agent from the category I, from the category II, or keeping the current agent.

Figure 7.15 and Figure 7.16 reveal a clear preference for an agent with twice the robustness of the current model. This approach to improvement is intuitive in highly aggressive environments, such as those with an 8-hour cell cycle duration or a radiosensitivity of 0.5, where the adoption of a significantly more robust agent is justifiable.

On the other hand, in less challenging environments, such as those with a 24-hour cell cycle, the decision to directly upgrade to an agent with doubled robustness might be premature. It might be worth considering other strategies before resorting to this drastic step.

This pattern of decision-making, favoring a two-time more robust agent, is consistently reflected across the histograms for different time steps where choices about agent robustness are to be made. A detailed analysis of these decisions can potentially highlight more nuanced strategies for agent selection based on specific

environmental factors.

7.5.5 Conclusion

An automatic decision-making process has been developed to assess the efficiency of a treatment agent based on tumor observations during treatment. This process incorporates two distinctive architectural designs: one utilizes a random forest classifier to make decisions based on time series data of the cancer cells, while the other employs a convolutional neural network classifier to make decisions informed by tumor images.

Upon comparison with earlier results, it's evident that our novel architectures tend to enhance treatment outcomes, particularly for tumors demonstrating reduced radiosensitivity. The one exception lies in environments with a radiosensitivity of 0.8. Conversely, our architectures did not surpass the already major results from previous sections when compared in environments with varying cell cycle durations. These results were already optimized across all key performance indicators, making it challenging to exceed their performance. Nevertheless, our architectures' performances closely align with those of the most effective agents. This alignment is ideal, as the treatment strategy aims to commence with a baseline treatment and escalate to a more aggressive regimen if necessary. It leads to treatments that are less aggressive for the patient as our architecture tends to choose, on average, smaller doses. The counterpart is that the treatment duration increases.

Moving forward, the focus of improvements should be the decision-making thresholds. At present, the process seems to favor swapping agents in pursuit of a more robust alternative too readily. Future explorations should aim to discover the optimal balance between ensuring 100% TCP and avoiding premature agent substitution. One solution could involve incorporating the Normal Tissue Complication Probability (NTCP) as a criterion, a measure of potential side effects resulting from radiation dose administration. Understanding these side effects better will help us refine several elements of our architectures, including the robustness classes of agents, the classification process, and the thresholds for deciding whether to change the current agent.

7.6 Further Works

This section is dedicated to resuming the further work that could be done from this master thesis. Some of the following ideas have already been discussed in previous sections.

General Improvements

- A significant enhancement to this work would be the inclusion of a criterion that considers the side effects resulting from radiation therapy. For example, the Normal Tissue Complication Probability (NTCP) criterion was not integrated into this study. The model distinctly lacks a measure of side effects, which is crucial for achieving balance in the development of a robust agent. Incorporating this metric would preclude the necessity of exclusively seeking the minimum number of fractions coupled with high radiation doses in every fraction.
- An important factor in cancer treatment is the treatment response time and it hasn't been incorporated into our model yet. Our present system calculates the death of cancer cells that meet the death criterion immediately after the first radiation dose is applied. This is a substantial deviation from the real biological process, as radiation treatment doesn't cause cancer cells to die instantly. Possible improvements could include integrating time-related elements into the model or even crafting a time-series model. Such adjustments might better emulate the true biological response to radiation therapy.
- The optimal frequency of the observations during treatment has to be found. In this work, we took observations every 48 hours, arbitrarily. Further explorations should be made concerning the optimal frequency to perform these observations.
- We should explore the combination of cell cycle changes with radiosensitivity changes. The cumulative effect of radiosensitivity, when viewed in the context of cell cycle modifications, has not yet been fully characterized. As the sum of radiosensitivity essentially represents a weighted average of the radiosensitivities of each stage of the cell cycle, it inherently changes as the duration of the cell cycle is altered. Therefore, further investigation should focus on understanding the implications of these combined factors, specifically the impact of varying cell cycle durations combined with radiosensitivity.
- The limitations of the cellular model discussed in Section 5.2 can be addressed with several adjustments to the model. These modifications would enable the exploration of more intricate environments regarding the adjustment of the nutrient consumption rate. Additionally, it would be worthwhile to investigate alterations to the critical nutrient level by adjusting the nutrient consumption of healthy cells, instead of solely focusing on cancer cells.

Training of agents

- In the present study, we have employed two distinct training methods for agents. Initially, the agent is trained in a benchmark environment (refer to Table 4.1) before being exposed to a specialized environment for the development of robustness. Conversely, in the second approach, the agent is directly trained within the specific environment, without any initial exposure to the benchmark. An alternative strategy could involve incrementally challenging the agent with increasingly demanding environments. For instance, if the objective is to cultivate an agent robust enough to handle an environment with a radiosensitivity level of 0.5, we could envisage a gradual training approach. This would entail exposing the agent to environments where radiosensitivity is sequentially reduced towards the target of 0.5. This method could potentially yield superior outcomes, especially in the context of extremely hostile environments. Moreover, it could also minimize the standard deviation of key performance indicators, given that the agent will undergo a more comprehensive training regime.
- The reward function is a critical aspect of the training process, modifying it can significantly transform how agents are trained. It serves as the compass that guides an agent's learning journey, making it

instrumental in achieving optimal results. The potency of the reward function extends to the possibility of enhancements. By exploring and experimenting with diverse reward functions, we can potentially unlock new avenues for improvement. This underscores the fundamental role that the reward function plays in advancing the efficacy of the training process.

Classifiers improvements

- The performances of the classifiers are sufficiently good to be used. However, an improvement of their Area Under the Curve (AUC) could lead to more accurate prediction about the simulation outcome and then lead to more accurate treatment. A potential way of improvement would be to give way more training samples to the classifiers. We have at our disposal an environment that can be exploited and modified to encounter a large variety of scenarios. Even if we have already tried to give an overview of all the possible simulation outcomes, we could certainly go further and handle way more specific situations. Convolution neural networks are particularly suited to perform generalizations about data but also to detect specific patterns.
- It may be beneficial to optimize our classification architecture to strengthen the correlation between the initial data of the CNN, inclusive of tumor images, and the final output from the CNN. We have reservations about the current architecture's efficacy in fully leveraging the potential of neural networks. Furthermore, the quality of the input data could be enhanced to provide our CNN classifier with the opportunity to achieve superior generalization and performance.

Agent Robustness Classes

The categorization of each agent's robustness was based on specific, yet arbitrarily selected criteria. This means there are countless other methods and criteria that could be used for similar categorization purposes. We had to make a few simplifications to conduct our current analysis. For example, we first considered the TCP and then the dosage administered as our main criteria.

However, we could have alternatively considered factors like the standard deviation of key performance indicators, or the extent of exploration in the observation space. We could have even conducted a more comprehensive study on the relationship between variations in cell cycle duration and radiosensitivity. Remember, any fluctuation in the cell cycle duration can affect the cumulative radiosensitivity due to the weighted sum, as we discussed in Section 5.4.

Automatic Decision-Making Process

- The process of automatic decision-making is heavily dependent on the output produced by the classifier. This procedure involves the use of arbitrarily selected thresholds to determine when a more robust agent needs to be engaged. There's room for further investigation into these thresholds, as elaborated in Section 7.5.4. Enhancements can be made to the decision-making process which could potentially integrate additional observations of the tumor, thereby enabling more precise decision-making.
- Significant insights into tumor responses to administered treatments have been discovered. Notably, these insights pertain to scenarios where the cell cycle duration has been modified. Our findings suggest that the initial doses of radiation should be particularly high, while subsequent doses can vary more significantly without compromising effectiveness. This discovery opens up new possibilities for exploring treatments that are handcrafted based on these observations. To illustrate, we might consider a treatment strategy where the first four radiation doses each deliver 4 Gy. The subsequent doses can then be carefully modulated to be as low as possible without compromising the radiotherapy efficacy.

In order to facilitate future contributions and development on this project, we have made our code open-source, similar to the approach taken by Moreau et al. [19] with their work. To make the use of the cellular environment along with the reinforcement learning algorithms more accessible, we have designed a Graphical User Interface (GUI). We encourage readers and fellow researchers to utilize our open-source code, which is readily available on our GitHub repository: `https://github.com/martinflor/RL_for_radiotherapy_treatment`. For an understanding of the GUI interface and how it could assist in carrying this work forward, please refer to Section 9.4.

Chapter 8

Conclusion

This work primarily seeks to underscore the profound value of applying reinforcement learning in identifying optimal radiotherapy treatments. Through our approach, we've shown that it's not only feasible but highly effective, to achieve significant reductions in key performance indicators, such as the total dose administered, the number of fractions, and the treatment duration. Consequently, we can deliver treatments with fewer fractions, reduced dosage, and shorter treatment durations. These findings align with the work of Moreau and al. [19]

Moreover, our research takes a step beyond, aiming to create robust reinforcement learning agents suitable for different types of cellular environments. These environments were modified in nutrient consumption rates, cell cycle durations, and radiosensitivities. Our agents were tested and tailored to provide the most effective treatments based on the specificities of each environment. Given the uncertainty surrounding the exact type of environment a tumor belongs to in real-world scenarios, it's impractical to determine the most appropriate agent beforehand. Consequently, we strived to devise an automatic decision-making process that adapts the ongoing treatment based on tumor observations, enabling automatic adaptation, circumventing the need to identify the environment type.

A crucial aim of this work was to construct an automated decision-making architecture. This system, capable of determining when to employ a more robust agent based on tumor treatment observations, begins with a baseline treatment and adjusts the treatment as and when decision criteria are satisfied. The resulting architecture led to a remarkable 42.8% decrease in the total dose administered to the patient, a 61.0% reduction in the number of fractions, and a 61.1% reduction in treatment duration compared to the baseline treatment.

Our architecture achieves similar performance with the most suitable agents in environments with modified cell cycle durations. Nevertheless, it enables the formulation of less aggressive treatments in such environments, as evidenced by the lower average dose administered each time compared to the best-performing agent. However, a slight drawback is that it results in longer treatment durations.

Our automatic decision-making process also provides improved performances on environments with modified radiosensitivity. It highlights that this type of environment does not require high-dosage treatment in the early stages even if the dose should be higher than 2.5 Gy in the first 400 hours, on average. The rest of the treatment allows using a large variety of doses between 2 and 4 Gy.

The insights gained from the treatment strategies investigated in this master thesis offer an understanding of tumor behavior within a cellular environment, particularly in relation to radiotherapy treatment. This research can lead to numerous valuable contributions. Foremost among them is the potential development of a more sophisticated and realistic cellular model that can accurately represent real-world scenarios.

Chapter 9

Annex

9.1 Reinforcement learning algorithms

9.1.1 Dynamic Programming

Dynamic programming (DP) is a general problem-solving framework for breaking down complex problems into smaller subproblems that can be solved independently and then combined to find the solution to the original problem. Dynamic programming algorithms rely on the assumption of complete knowledge of the environment. Such algorithms serve as theoretical references rather than practical solutions for real-world problems. DP algorithms are used to find the value functions defined earlier. So, the goal of such algorithms is to find a solution to the bellman optimality equations :

$$\begin{aligned} v_*(s) &= \max_{a \in \mathbf{A}} \mathbb{E}[\mathbf{R}_{t+1} + \gamma v_*(\mathbf{S}_{t+1}) | \mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \max_{a \in \mathbf{A}} \sum_{s',r} p(s', r|s, a)[r + \gamma v_*(s')] \end{aligned} \tag{9.1}$$

$$\begin{aligned} q_*(s, a) &= \mathbb{E}[\mathbf{R}_{t+1} + \gamma \max_{a\prime} q_*(\mathbf{S}_{t+1}, a') | \mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a\prime} q_*(s', a')], \quad \forall s, s' \in \mathbf{S}, \ \ \forall a \in \mathbf{A}(s) \end{aligned} \tag{9.2}$$

Policy Evaluation

Policy evaluation describes how to compute the state-value function v_π for an arbitrary policy π. Remind that the bellman equation is :

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi[\mathbf{G}_t | \mathbf{S}_t = s] \\ &= \mathbb{E}_\pi[\mathbf{R}_{t+1} + \gamma \mathbf{G}_{t+1} | \mathbf{S}_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma \, \mathbb{E}_\pi[\mathbf{G}_{t+1} | \mathbf{S}_{t+1} = s']] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_\pi(s')], \quad \forall s \in \mathbf{S} \end{aligned} \tag{9.3}$$

Once again, if the dynamics are known, the bellman equation is a system of linear equations with n equations, n being the number of states. Iteratives methods can be used. v_0 is chosen arbitrarily and then :

$$\begin{aligned} v_{k+1}(s) &:= \mathbb{E}_\pi[\mathbf{R}_{t+1} + \gamma v_k(\mathbf{S}_{t+1}) | \mathbf{S}_t = s] \\ &= \sum_a \pi(a|s) \sum_{s'} \sum_r p(s', r|s, a)[r + \gamma v_k(s')], \quad \forall s \in \mathbf{S} \end{aligned} \tag{9.4}$$

It can be shown that the sequence of v_k's will converge toward v_π. This algorithm is called **iterative policy evaluation**. In order to compute $v_{k+1}(s)$, one needs to know the next state $s^{'}$. In practice, the iterative

process can be done backward, starting from the last state and going toward the first one. However, other techniques consist to compute with all the states at once. Pseudo-code of the iterative policy evaluation algorithm 4 is described below :

Algorithm 4 Iterative Policy Evaluation

π : policy to be evaluated
$V(s) \leftarrow 0, \quad \forall\, s \in S$
$\theta \leftarrow threshold$
while $\Delta > \theta$ **do**
  $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ **do**
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_a \pi(s,a) \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
    $\Delta \leftarrow max(\Delta, |v - V(s)|$
  end for
end while
Ensure: $V \approx V_\pi$

Policy Improvement

Computing the value function $v_\pi(s)$ has the goal to help finding better policies. For a given state s, it is possible that an action $a \neq \pi(s)$ may be better than simply following the initial policy. The action-value function can be used to compare between following the current policy or to follow an action out of this policy :

$$\begin{aligned} q_\pi(s,a) &:= \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(\mathbf{S}_{t+1}) | \mathbf{S}_t = s, \mathbf{A}_t = a] \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{9.5}$$

The policy improvement theorem states that if :

$$q_\pi(s, \pi^{'}(s)) \geq v_\pi(s), \quad \forall s \in \mathbf{S} \tag{9.6}$$

then, the policy $\pi_{'}$ must be better or equal to π. It means that :

$$v_{\pi^{'}}(s) \geq v_{\pi(s)}, \quad \forall s \in \mathbf{S}$$

Following this, the policy $\pi^{'}$ is the same as π, except that at state s : $\pi^{'}(s) = a \neq \pi(s)$. For others states than s, the following equality is true : $q_\pi(s, \pi^{'}(s)) = v_\pi(s)$. So, equation 9.6 translates that if it is better to follow $\pi'(s) = a$ once in state s and therafter follow π, than it would to simply follow π, then the new policy π' is better than π. This is why the changed policy is better than the previous one. The development of the policy improvement theorem is described in the annex 9.1.3.

Now that evaluating a change in the policy at a given state is defined, one can extend it to consider changes at all states for all possible actions :

$$\begin{aligned} \pi'(s) &:= \arg\max_a q_\pi(s,a) \\ &= \arg\max_a \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1} \mid S_t = s, A_t = a] \\ &= \arg\max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{9.7}$$

where π' is a new **greedy** policy. This policy chooses the best actions in short term look ahead (one step) with regards to v_π. Since this greedy policy respects the policy improvement theorem, it ensures that this policy will be as good as, or better than the original one. Suposing the new greedy policy π' is as good as but not better than the old policy π, then $v_\pi = v_{\pi'}$, it follows that $\forall s \in S$:

$$
\begin{aligned}
v_{\pi'}(s) &= max_a \mathbb{E}[R_{t+1} + \gamma v_{\pi'}(S_{t+1}) \mid S_t = s, A_t = a] \\
&= max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_{\pi'}(s)]
\end{aligned} \tag{9.8}
$$

This last equation is exactly the **Bellman optimality** equation 2.5, and therefore, $v_{\pi'}$ must be the optimal policy v_* (see section 2.1.3), and both π and π' are optimal policies. It highlights that policy improvement gives a strictly better policy than the original policy except when the original policy is already the optimal one. Algorithm 5 shows the pseudo-code of the policy improvement.

Algorithm 5 Policy Improvement

π : policy to be improved
$V(s)$, *the value function*
$policy_stable \leftarrow True$
for each $s \in \mathcal{S}$ **do**
  $old_action \leftarrow \pi(s)$
  $\pi(s) \leftarrow argmax_a \sum_{s',r} p(s', r|s, a)[r + \gamma V(s')]$
  if $old_action \neq \pi(s)$ **then**
    $policy_stable \leftarrow False$
  end if
end for
if *policy_stable* **then**
  $V \approx v_*$
  $\pi \approx \pi_*$
else
  Repeat
end if

Policy Iteration

Policy Iteration provides a baseline to converge toward an optimal policy. Repeating policy evaluation and policy improvement, in a finite Markov Decision Process (only a finite number of policies), this alternating process must converge. So, using v_π to improve π, the new value function $v_{\pi'}$ can be computed and used to improve once again the policy. The drawback is that each policy iteration needs policy evaluation which is also an iterative process. The step of policy evaluation can be pruned without loosing the convergence guarantee of the policy iteration process. **Value iteration** is an algorithm where the policy evaluation is stopped after just one update of each state. It avoids the iterative process of policy evaluation and only performs sweeps between evaluation and improvement.

9.1.2 Monte Carlo Methods

Monte Carlo (MC) Methods are a way of solving problems in reinforcement learning that do not assume the agent knows the complete environment. These methods rely on samples of states, actions, and rewards, either from a real or simulated environment, to estimate the value functions and policies. Unlike Dynamic Programming, Monte Carlo Methods estimate the value of each state-action pair by averaging returns from visits to that state. This approach does not require knowledge of the dynamics p of the environment, which can be difficult or impossible to compute in some cases. Additionally, Monte Carlo Methods allow for state-independent estimation and do not require the entire state space, unlike DP, meaning that averaging returns can be made for only one or a subset of states.

MC Policy Prediction

First, the purpose is, as in Dynamic Programming, considering the learning of the state-value function v for a given policy π. The value of a state is the expected cumulative discounted reward. It is possible to estimate it by **experience**, meaning estimating it by averaging the returns observed after visiting a state. Of course, the average of returns will converge to the expected value, as more returns are accounted. It is important to precise that only episodic tasks are considered such that it ensures to have access to well-defined returns.

Visiting state s means that we encounter an occurrence of state s in an episode. Hence, a state can be visited multiple times during an episode. So that, **first-visit** Monte Carlo method estimates v_π based on the average of the returns following the first visit to s and **every-visit** Monte Carlo method estimates v_π based on the average of the returns following all visits to state s. Thus, $v_\pi(s)$ is computed as the average of returns after visiting the state s.

Action Value estimation

As mentioned before, Monte Carlo can get rid of a model of the environment, such that computing v_π is no more sufficient to determine a policy. Equation 9.4 highlights the needs of a model of the environment to know the transition probabilities. Without any model of the environment, one needs to estimate the action values, in other words, the values of different action in a state. In Dynamic Programming, looking one step ahead was sufficient thanks to the transition probabilities and the recursion in Bellman equations, that allow to compute the estimation of a state based on the next state only. For Monte Carlo methods, the transition probabilities are not considered to be known such that computing $v_\pi(s)$ is practically impossible. Then, Monte Carlo Methods aim to find q_*.

Estimating $q_\pi(s, a)$ (the expected return when starting in state s and taking action while thereafter following policy π) is the same principle as estimating the state value function but now, we are visiting state-action pairs rather than only a state (visiting a state-action pair means the state s is visited and action a is taken in it). Following π means that, if the policy is deterministic, a lot of state-action pairs won't be visited because the same action at same sate will always be taken. Then, Monte Carlo methods won't be able to average anymore and won't improve with experience. Ideally, one needs to estimate all the state-action pairs and this is the so called problem **maintaining exploration**. It is absolutely necessary to keep exploring new state-action pairs. A way to achieve it, is to start episodes in a given state-action pair and setting a nonzero probability to each pair to be selected. In practice, to ensure the exploration, the policy won't be deterministic but rather stochastic.

Monte Carlo Control

Monte Carlo control is the purpose of approximating optimal policies as in Dynamic Programming with the section Policy Iteration. The same principle is used such that two alternating components are performed, policy evaluation and policy improvement. Policy evaluation is performed following what is described in the section 9.1.2. Policy Improvement is performed by making the policy greedy with regards to the current value function. Thus, an action is chosen by the maximal action-value : $\pi(s) := \arg\max_a q(s, a)$. As discussed in the Dynamic Programming section, the policy improvement theorem states that π_{k+1} will be an improved version of π_k except if π_k is already an optimal policy. Unfortunately, two assumptions have been made : episodes start in a state–action pair, and that every pair has a nonzero probability of being selected as the start such that it guarantees that all state–action pairs will be visited an infinite number of times in the limit of an infinite number of episodes. These assumptions are practically impossible to assume. As described in the section of Policy Iteration, a similar process to Value Iteration can be used for Monte Carlo Methods. In this way, a natural way to reproduce Value Iteration is to alternate between evaluation and improvement on an episode-by-episode basis such that the infinite number of episodes hypothesis is not needed anymore. Note that this method converges when there is stability, meaning that both the policy and the value function are optimal.

The second assumption is called **Exploring Starts** and highlights the need that all actions are selected an infinite number of time. To ensure this hypothesis, the agent need to continue to select them and this is achieved via **on-policy** and **off-policy** methods. On-policy methods learn from the actions that an agent takes in the environment (the agent updates its policy as it interacts with the environment), whereas in off-policy methods, the agent follows a separate behavior policy while learning from the actions taken by that policy or from a replay of past experiences.

On-policy methods In on-policy control methods, the agent's policy is initially non-deterministic, meaning that it assigns some probability $\pi(a|s)$ to each possible action a in each state s. Thus, such a policy is said to be **soft** because $\pi(a|s) > 0$. As the agent interacts with the environment and learns, the policy is gradually updated to become more confident in certain actions and less confident in others. The goal is for the policy to converge to a deterministic policy that always selects the best action in each state. ϵ-greedy policies are

well known policies where most of the time they choose an action that has the maximal action value but with a probability of ϵ to select a random action. It can be shown that policy iteration/improvement works for ϵ-greedy policies

Off-policy methods In reinforcement learning, the goal is to find the best policy that tells the agent which action to take in each state. To find the best policy, the agent needs to explore different actions and learn from the resulting environment interactions. However, exploration is not always the best strategy, as it may lead to suboptimal actions and delay learning. On-policy methods balance exploration and exploitation by learning about a policy that is similar to the optimal policy, but still allows for some exploration. In this way, the agent can take actions that are not always the best but that will help it learn more about the environment.

Off-policy methods, on the other hand, separate the exploration and learning processes. The agent uses a **behavior policy** to take actions and explore the environment, and a separate **target policy** to learn from the actions taken by the behavior policy. The agent uses the knowledge it gains from the behavior policy to improve the target policy. This approach allows the agent to explore more efficiently, as the behavior policy can be specifically designed to explore different actions, while the target policy can be improved faster as it doesn't need to explore by itself.

Off-policy methods assumes what is called **coverage**. The target policy is denoted by π and the behavior policy is denoted b. We wish to estimate v_π or q_π with episodes from the policy b ($b \neq \pi$), then, the two policies are considered fixed and given. Using the episodes from the behavior policy to estimate values for π require that every action taken under the policy π is, at least, occasionally taken under the policy b. In others words, we require that $\pi(a|s) > 0$ implies $b(a|s) > 0$. This is the coverage assumption leading to the policy b being stochastic in states where it is not identical to π.

In off-policy methods, **importance sampling** is a technique used to adjust the learning signal in order to correct for the difference between the behavior policy and the target policy. When an agent follows a behavior policy to take actions in an environment, it generates a sequence of state-action-reward-next state transitions (also known as "experience"). These experiences are used to update the agent's knowledge of the value of different actions. However, if the behavior policy is different from the target policy, the experiences generated by the behavior policy will not be representative of the value of the actions according to the target policy. Importance sampling corrects for this by adjusting the learning signal based on the probability of the experience under the target policy versus the behavior policy. For example, if an action is more likely to be taken by the target policy than by the behavior policy, the learning signal for that action will be adjusted upward to account for the fact that it is more important according to the target policy. Conversely, if an action is less likely to be taken by the target policy than by the behavior policy, the learning signal for that action will be adjusted downward. In practice, importance sampling is applied by weighting each experience by the ratio of the target policy to the behavior policy at the state and action of the experience. This is called the importance sampling ratio (ISR) :

$$\rho_{t:T-1} := \frac{\prod_{k=t}^{T-1} \pi(A_k|S_K)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_K)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

This represents the relative probability of the trajectory under the target and the behavior policies. For example, if the ISR is 10, it means that the experiences being considered is 10 times more likely to occur under the target policy than under the behavior policy. In this way, a high value of ISR indicates that the experience is valuable for learning about the target policy, since it is unlikely to have been generated by the behavior policy. On the other hand, a low value of ISR means that the experience is unlikely to be informative about the target policy and should not be used to update the agent's knowledge.

Let's define $\tau(s)$ that is the set of all time steps in which state s is visited and T(t), the first time of termination following time t. Then, G_t denotes the return after t up through T(t). In this way, estimating $v_\pi(s)$ is weighting the returns G_t with the ISR and then averaged :

$$V(s) := \frac{\sum_{t\in\tau(s)} \rho_{t:T(t)-1} G_t}{|\tau_s|}$$

where this version of importance sampling is called **ordinary importance sampling**.

Another alternative is **weighted importance sampling** which uses a weighted average :

$$V(s) := \frac{\sum_{t \in \tau(s)} \rho_{t:T(t)-1} G_t}{\sum_{t \in \tau(s)} \rho_{t:T(t)-1}}$$

that is equal to 0 if the denominator is 0.

9.1.3 Policy Improvement Theorem

$$\begin{aligned}
v_\pi(s) &\leq q_\pi(s, \pi'(s)) \\
&= \mathbb{E}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s, A_t = \pi'(s)] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma v_\pi(S_{t+1}) \mid S_t = s] \\
&\leq \mathbb{E}[R_{t+1} + \gamma q_\pi(S_{t+1}, \pi'(S_{t+1})) \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 v_\pi(S_{t+2}) \mid S_t = s] \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 v_\pi(S_{t+3}) \mid S_t = s] \\
&\vdots \\
&\leq \mathbb{E}_{\pi'}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots \mid S_t = s] \\
&= \mathbb{E}_{\pi'}[G_t \mid S_t = s] \\
&= v_{\pi'}(s)
\end{aligned} \tag{9.9}$$

9.2 Early Stopping Possibilities

This section underscores the untapped potential of implementing early stopping strategies to expedite the training process of our RL agents. By scrutinizing the learning curves of these agents, we can identify and establish an effective criterion for early stopping. This methodology can significantly enhance efficiency by curtailing unnecessary expenditure of time in the training process. While we won't delve into an extensive investigation of this technique in this particular context, we aim to emphasize its promising capabilities.

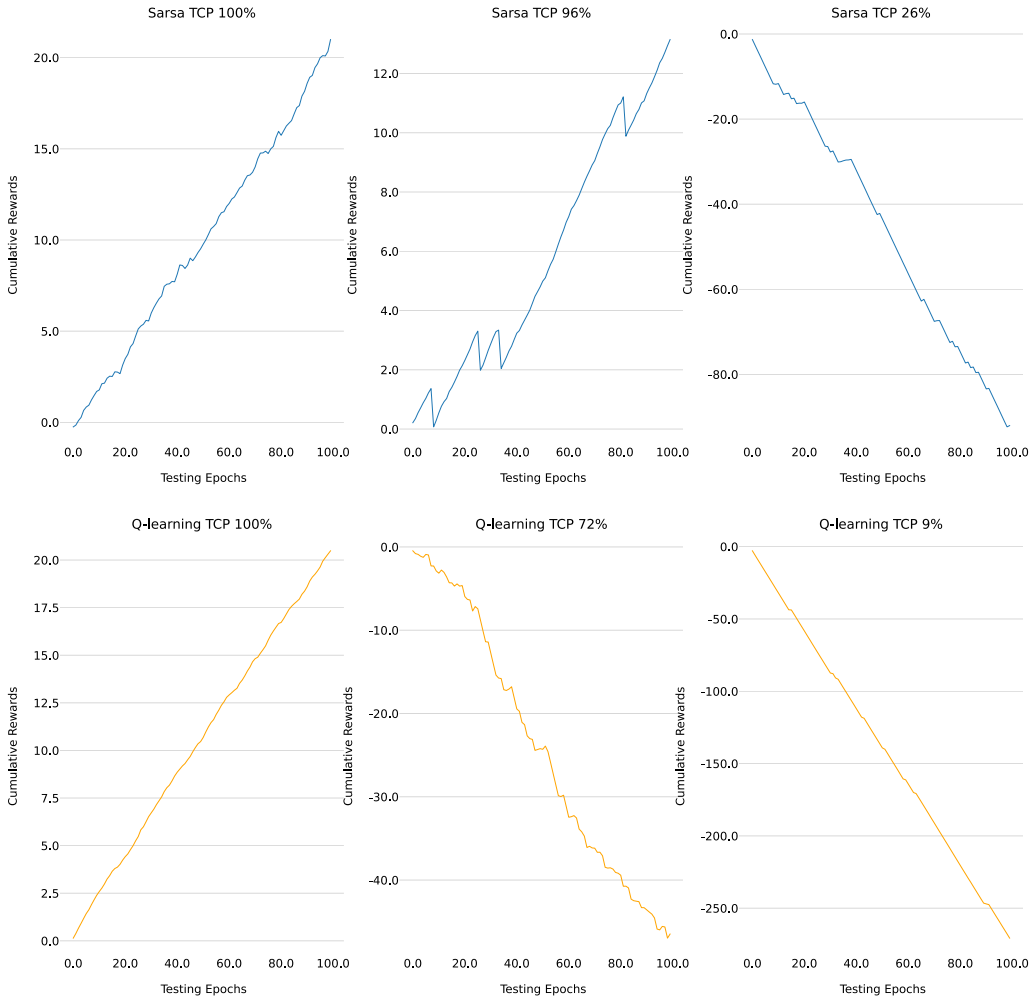


Figure 9.1: Q-learning and Sarsa learning curves.

Figure 9.1 presents the learning curves for two reinforcement learning algorithms, Q-learning and Sarsa agents. These learning curves represent the cumulative sum of rewards obtained over numerous episodes throughout the training process.

The learning curves can be broadly categorized into three distinct types, which are directly related to the agents' ability to achieve optimal TCP values:

1. Constantly increasing curves: These curves depict a steady improvement in the agent's performance, ultimately resulting in a TCP of 100%. This indicates that the agent has effectively learned to deliver the optimal radiotherapy treatment plan ensuring complete tumor control.

2. Curves with minor decreasing segments: These curves generally show upward trends but may have occasional dips, indicating a slight suboptimal performance in certain episodes. As a result, the achieved TCP values are close to, but not quite reaching, 100%. These agents are still considered effective in administering radiotherapy treatment but may require further fine-tuning to achieve maximum tumor control.

3. Consistently decreasing curves: These learning curves show a downward trajectory throughout the training process, suggesting that the agent's performance is steadily declining. Such agents are unable to achieve optimal TCP values, leading to values significantly lower than 100%. This may be due to issues in the learning algorithm or the need for additional training episodes to reach convergence.

Additionally, these learning curves can serve as an invaluable tool for implementing **Early Stopping** during the training of the agents. Early Stopping is a technique employed to prevent overfitting by halting the training process once the agent's performance reaches an optimal or near-optimal state, as indicated by the learning curve. This can not only save computational resources but also help to ensure the generalizability of the learned policy when applied to new environments or test scenarios.

9.3 Results of the automatic robustness decision-making

This section is dedicated to providing a summary of the results obtained in Section 7.5.2. No further explanations will be provided as they are present in the aforementioned section.

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Radiosensitivity of 0.50	90.0	113.7 ± 21.77	34.7 ± 8.49	832.1 ± 203.76	93.5
Radiosensitivity of 0.55	100.0	91.3 ± 16.92	27.6 ± 5.92	662.6 ± 142.06	93.7
Radiosensitivity of 0.6	100.0	85.05 ± 19.30	26.2 ± 7.17	628.8 ± 172.1	93.8
Radiosensitivity of 0.66	100.0	79.52 ± 14.80	24.9 ± 5.85	597.1 ± 140.4	94.6
Radiosensitivity of 0.72	100.0	61.26 ± 9.71	18.96 ± 3.7	455.0 ± 88.2	93.9
Radiosensitivity of 0.80	100.0	59.54 ± 11.22	18.25 ± 3.97	438.0 ± 95.30	93.7
Cell Cycle of 8 hours	0.0	87.59 ± 33.4	35.21 ± 6.6	845.0 ± 159.0	0.0
Cell Cycle of 10 hours	72.0	121.3 ± 36.2	40.52 ± 12.6	972.5 ± 303.4	62.0
Cell Cycle of 12 hours	98.0	76.30 ± 18.68	24.37 ± 7.2	584.9 ± 172.7	84.7
Cell Cycle of 14 hours	100.0	51.93 ± 11.73	16.48 ± 4.6	395.5 ± 109.5	87.3
Cell Cycle of 16 hours	100.0	44.53 ± 7.8	14.25 ±3.0	342.0 ± 72.3	91.0
Cell Cycle of 18 hours	100.0	47.0 ± 6.76	15.0 ± 2.65	360.2 ± 63.54	90.7
Cell Cycle of 20 hours	100.0	47.9 ± 13.19	16.2 ± 7.48	387.8 ± 179.5	91.3
Cell Cycle of 24 hours	100.0	42.6 ± 13.10	15.4 ± 8.55	369.6 ± 205.1	93.7

Table 9.1: Performances achieved by the automatic decision-making based on the output of the binary random forest classifier.

Environment	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Radiosensitivity of 0.50	76.0	125.5 ± 19.82	40.2 ± 8.01	965.5 ± 192.36	90.9
Radiosensitivity of 0.55	96.0	103.1 ± 20.19	32.5 ± 7.37	781.0 ± 176.80	93.9
Radiosensitivity of 0.60	100.0	90.0 ± 17.42	28.3 ± 6.02	678.5 ± 144.47	94.2
Radiosensitivity of 0.66	100.0	79.8 ± 14.14	25.3 ± 5.02	606.8 ±120.5	93.2
Radiosensitivity of 0.72	100.0	64.7 ± 11.28	20.9 ± 4.05	500.9 ± 97.29	93.4
Radiosensitivity of 0.80	100.0	62.9 ± 8.83	20.7 ± 3.36	495.8 ± 80.55	93.7
Cell Cycle of 8 hours	0.0	59.5 ± 2.81	29.8 ± 1.40	714.0 ± 33.7	0.0
Cell Cycle of 10 hours	4.0	75.3 ± 18.34	36.3 ± 5.21	870.5 ± 124.93	3.0
Cell Cycle of 12 hours	58.0	84.6 ± 21.36	33.4 ± 12.20	801.8 ± 292.72	47.3
Cell Cycle of 14 hours	100.0	59.0 ± 11.07	19.9 ± 4.25	478.3 ± 101.89	85.17
Cell Cycle of 16 hours	100.0	48.9 ± 7.98	16.7 ± 3.09	400.1 ± 74.24	88.9
Cell Cycle of 18 hours	100.0	44.9 ± 6.22	15.7 ± 2.42	375.6 ± 58.03	90.4
Cell Cycle of 20 hours	100.0	46.8 ± 8.40	16.1 ± 3.17	386.6 ± 75.96	90.6
Cell Cycle of 24 hours	100.0	40.1 ± 6.99	13.7 ± 2.62	327.6 ± 62.80	93.3

Table 9.2: Performances achieved by the automatic decision-making based on the output of the binary convolutional neural network classifier.

Table 9.1 and Table 9.2 depict the results obtained by the automatic decision-making with, respectively, the binary random forest classifier and the binary convolutional neural network classifier.

9.3.1 Treatment shapes

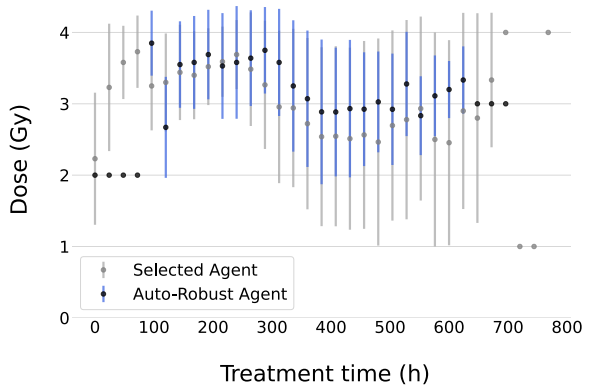


Figure 9.2: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.8. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

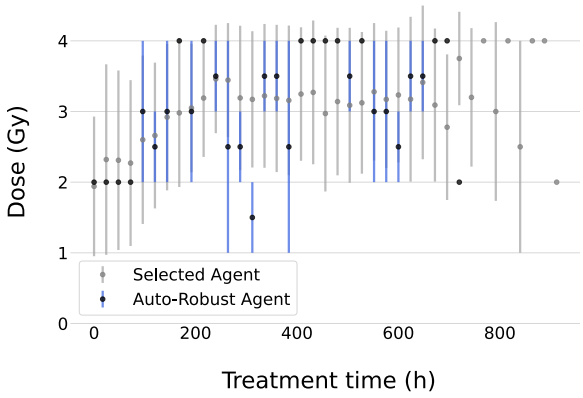


Figure 9.3: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.7. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

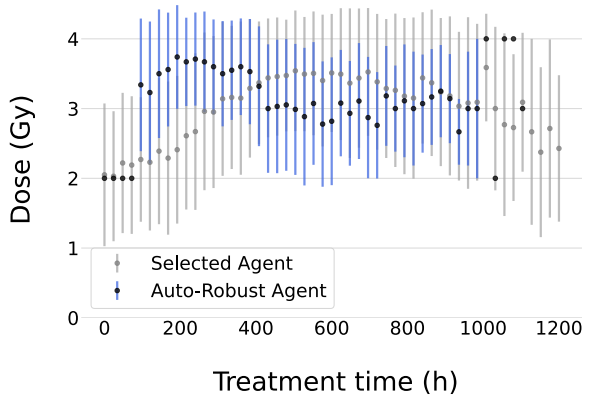


Figure 9.4: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.6. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

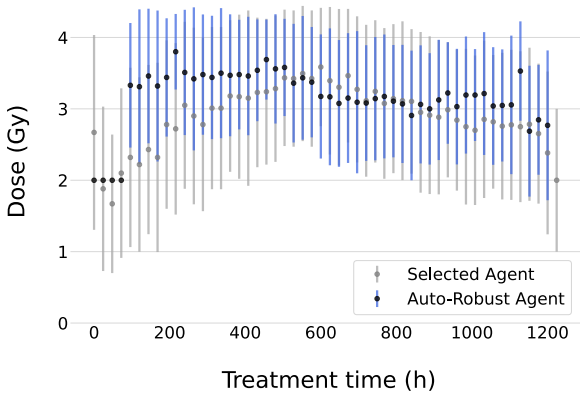


Figure 9.5: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a radiosensitivity of 0.5. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

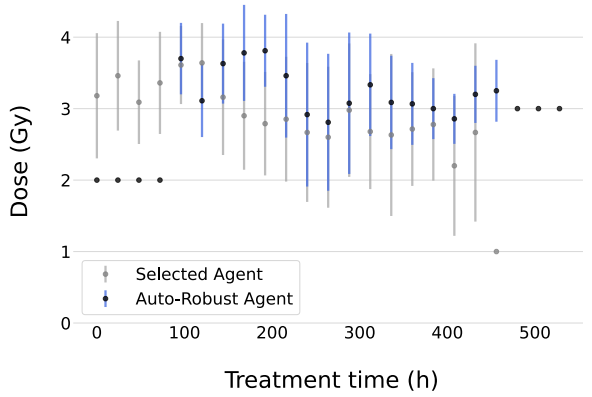


Figure 9.6: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a cell cycle duration of 24 hours. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

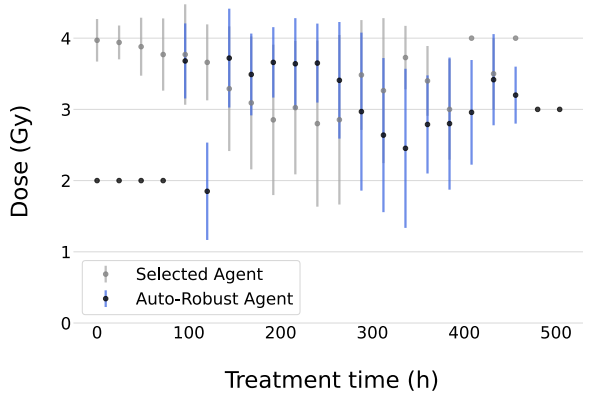


Figure 9.7: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a cell cycle duration of 20 hours. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

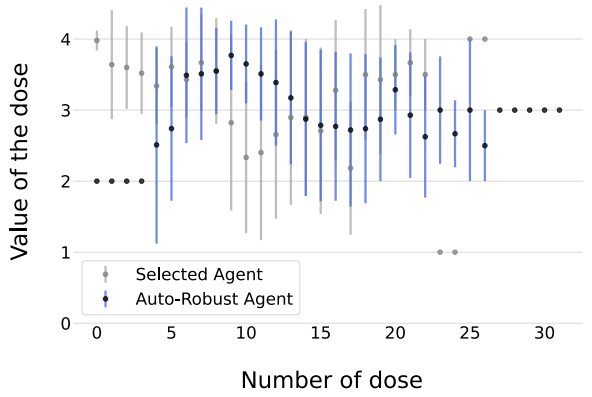


Figure 9.8: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a cell cycle duration of 18 hours. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

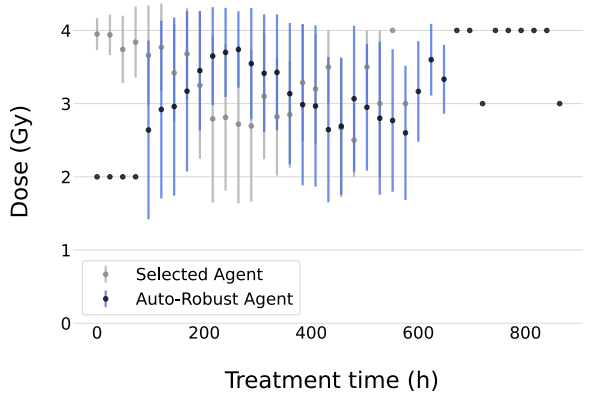


Figure 9.9: Radiotherapy treatments' patterns found by the selected agent and the ARA agent for environments with a cell cycle duration of 16 hours. 100 simulations have been conducted with both agents to provide mean and standard deviation values.

9.3.2 Comparison with the baseline treatment

Radiosensitivity of 0.8

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	100.0	102.06 (-)	51.03 (-)	1224.72 (-)	16.4
Auto-Robust Agent (RF)	100.0	59.54 (-41.6%)	18.25 (-64.2%)	438.0 (-64.2%)	93.7
Auto-Robust Agent (CNN)	100.0	62.8 (-38.4%)	20.7 (-59.4%)	495.8 (-59.5%)	93.7

Table 9.3: Best results and improvements for an environment with a radiosensitivity of 0.8. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 52.4%.

Radiosensitivity of 0.7

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	1.0	100.94 (-)	50.47 (-)	1211.3 (-)	8.1
Auto-Robust Agent (RF)	100.0	61.26 (-39.3%)	18.96 (-62.4%)	455.0 (-62.4%)	93.9
Auto-Robust Agent (CNN)	100.0	64.7 (-35.9%)	20.9 (-58.6%)	500.9 (-58.6%)	93.4

Table 9.4: Best results and improvements for an environment with a radiosensitivity of 0.7. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 51.0%.

Radiosensitivity of 0.6

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	0.0	101.4 (-)	50.7 (-)	1216.8 (-)	2.9
Auto-Robust Agent (RF)	100.0	85.1 (-16.1%)	26.2 (-48.3%)	628.8 (-48.6%)	93.9
Auto-Robust Agent (CNN)	100.0	90.0 (-11.2%)	28.3 (-44.2%)	678.5 (-44.2%)	94.2

Table 9.5: Best results and improvements for an environment with a radiosensitivity of 0.6. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 33.2%.

Cell cycle duration of 24 hours

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	99.0	70.1	35.1 (-)	841.4 (-)	99.9
Auto-Robust Agent (RF)	100.0	42.6 (-39.2%)	15.4 (-56.1%)	369.6 (-56.1%)	93.7
Auto-Robust Agent (CNN)	100.0	40.1 (-42.8%)	13.7 (-61.0%)	327.6 (-61.1%)	93.3

Table 9.6: Best results and improvements for an environment with a cell cycle of 24 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 55.0%.

Cell cycle duration of 20 hours

This section describes the results obtained for simulations based on an environment with a cell cycle duration of 20 hours. We will take the best results in the previous sections and compare them with results obtained from the automatic decision-making process.

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	0.0	98.26 (-)	49.13 (-)	1179.12 (-)	0.5
Auto-Robust Agent (RF)	100.0	47.9 (-51.3%)	16.2 (-67.0%)	387.8 (-67.1%)	91.3
Auto-Robust Agent (CNN)	100.0	46.8 (-52.4%)	16.1 (-67.2%)	386.6 (-67.2%)	90.6

Table 9.7: Best results and improvements for an environment with a cell cycle of 20 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 62.3%.

Cell cycle duration of 18 hours

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	100.0	87.4 (-)	43.69 (-)	1048.56 (-)	0.01
Auto-Robust Agent (RF)	100.0	47.0 (-46.2%)	15.0 (-65.7%)	360.2 (-65.6%)	90.7
Auto-Robust Agent (CNN)	100.0	44.9 (-48.6%)	15.7 (-64.1%)	375.6 (-64.2%)	90.4

Table 9.8: Best results and improvements for an environment with a cell cycle of 18 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 59.0%.

Cell cycle duration of 16 hours

	TCP (%)	Dose (Gy)	Fractions (-)	Duration (h)	Survival (%)
Baseline	0.0	78.2 (-)	38.1 (-)	914.2 (-)	0.01
Auto-Robust Agent (RF)	100.0	44.53 (-43.1%)	14.25 (-63.4%)	342.0 (-62.6%)	91.0
Auto-Robust Agent (CNN)	100.0	48.9 (-37.5%)	16.7 (-57.1%)	400.1 (-56.2%)	88.9

Table 9.9: Best results and improvements for an environment with a cell cycle of 16 hours. δ values are between parentheses. Each experiment conducted has been repeated 100 times to compute the mean values of the key performance indicators.

Mean improvement over the key performance indicators with the CNN classifier : 50.3%.

9.4 Graphical User Interface

This section is devoted to elucidating the intricacies of the Graphical User Interface (GUI) that we have meticulously designed to streamline the application of Reinforcement Learning (RL) algorithms in radiother-

apy treatment. Our intention behind this GUI is to establish an open-source project that is accessible to a global audience, regardless of their technical background.

This user-friendly interface is paired with a selection of RL agents that are readily available on our GitHub repository `https://github.com/martinflor/RL_for_radiotherapy_treatment`. Additionally, a comprehensive cellular model is implemented, creating a holistic platform for effective RL-based simulations. The GUI affords users the flexibility to experiment with a wide array of cellular parameters, facilitating both straightforward and complex simulations with relative ease.

Throughout this section, we will provide a step-by-step guide on how to navigate and optimally utilize the capabilities of this GUI.

9.4.1 Treatment page

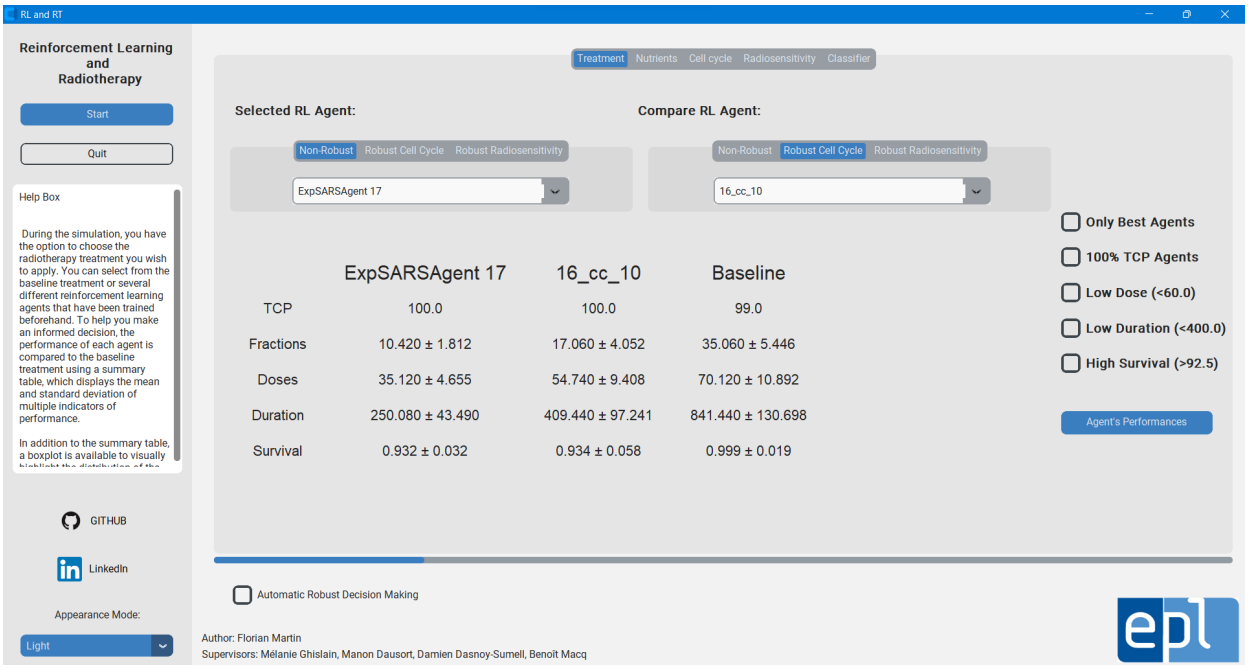


Figure 9.10: Home page of the GUI.

Figure 9.10 displays the home page of the Graphical User Interface (GUI). This interface is primarily designated for the selection of the RL agent that will be implemented during the simulation process. The segment labeled "Selected RL Agent" represents the chosen RL agent, which will administer the treatment throughout the simulation. Conversely, the "Compare RL Agent" section serves to aid users in performing side-by-side evaluations against the selected RL agent within a summary table.

This summary table presents a comparative analysis of key performance indicators across three distinct treatment options. The treatment on the left corresponds to the "Selected RL Agent", while the one situated in the middle is associated with the "Compare RL Agent". The third treatment is a baseline or control treatment, offering a standard for comparison. The purpose of this summary table is to facilitate a direct comparison between various RL agents, highlighting differences and similarities in relation to the defined key performance indicators. This comparison allows users to make more informed decisions regarding the most effective RL agent for their specific use case.

The agent labeled as "Expected Sarsa agent 17" represents the product of 17 iterations in the training process of the Expected Sarsa algorithm. Similarly, the "16_cc_10" agent refers to an agent specifically trained for a cell cycle duration of 16 hours, with "10" signifying the number of iterations completed during its training

process.

Several interactive buttons are available on the page. On the right side, there is a selection button that allows the user to filter agents based on specific characteristics. For instance, one can choose to display only agents that have achieved a 100% TCP. In this case, only those agents will appear in the list on the left.

At the bottom of the page, there is another significant button titled "Automatic Robust Decision Making". This feature is a key component of the decision-making system outlined in this Master's thesis. It triggers the system to evaluate the agent's performance based on 'during treatment' observations of the tumor. If deemed necessary, it then initiates the use of a more robust agent to enhance treatment effectiveness. This sophisticated feature is designed to ensure optimal decision making and treatment planning in real-time.

9.4.2 Nutrients page

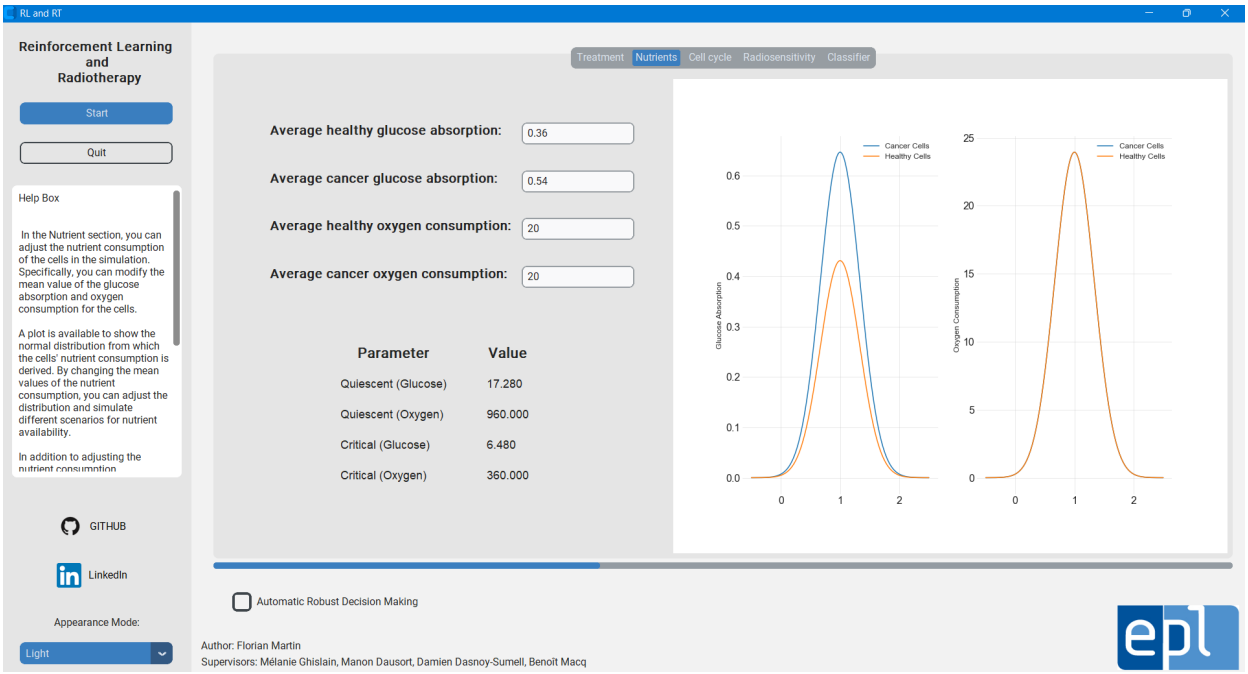


Figure 9.11: Nutrients page of the GUI.

Figure 9.11 displays the nutrients page of the Graphical User Interface (GUI). This page is specifically designed for adjusting the nutrient parameters within the simulation. Here, it's possible to modify the average nutrient consumption of both healthy and cancer cells. Correspondingly, the quiescent and critical consumption levels are calculated automatically when these average values are altered.

On the left side of the page, a dynamic plot visualizes the distribution of nutrients. This plot is updated in real-time whenever the user changes the mean consumption value, offering an immediate visual feedback of how these modifications impact the nutrient distribution.

9.4.3 Cell cycle page

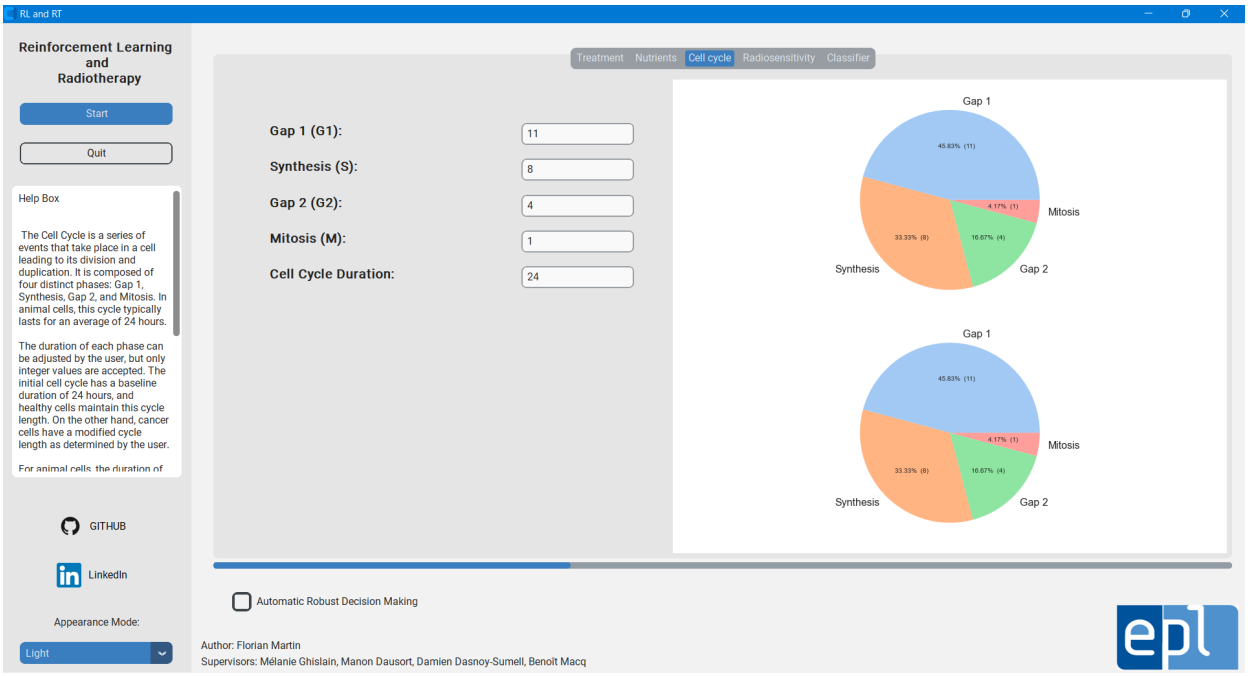


Figure 9.12: Cell cycle page of the GUI.

Figure 9.12 showcases the cell cycle page of the GUI. This interface facilitates customization of the cell cycle parameters for the cancer cells within the simulation. Users can alter the duration of each stage of the cell cycle to meet their specific needs.

The total duration of the cell cycle is dynamically updated as it is calculated automatically as the sum of the durations assigned to the four stages. In addition, a pie chart visually represents the proportion of time taken by each stage of the cell cycle, thus providing a clear and intuitive understanding of how the total cell cycle time is divided across different stages.

9.4.4 Radiosensitivity page

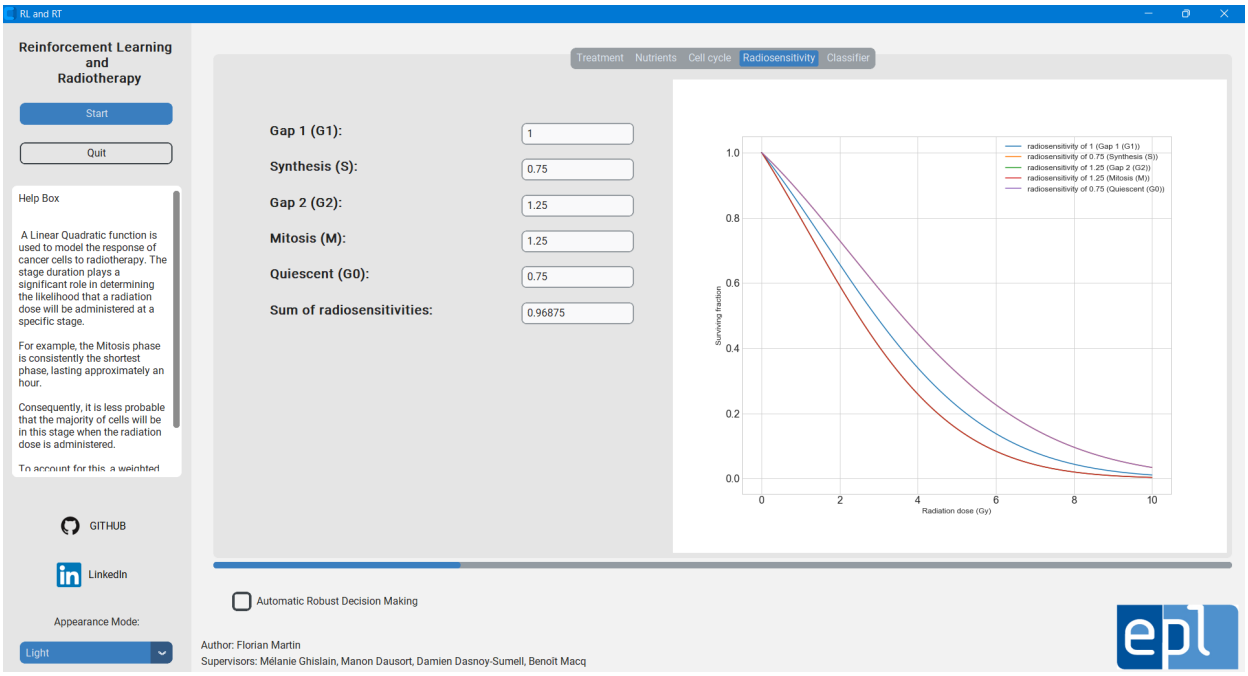


Figure 9.13: Radiosensitivity page of the GUI.

Figure 9.13 presents the radiosensitivity page of the GUI. This page is specifically designed to enable users to adjust the radiosensitivity parameters of cells. Each stage of the cell cycle has a specific radiosensitivity attributed to both healthy and cancer cells, which can be individually modified.

As changes are made, a weighted average of the radiosensitivity parameters is calculated and updated in real-time. This reflects the impact of the alterations across different stages of the cell cycle. Concurrently, survival curves are dynamically represented on an integrated plot, providing a visual and intuitive understanding of the changes as they occur. This real-time update ensures that users can observe and analyze the impact of their adjustments immediately.

9.4.5 Classifier page

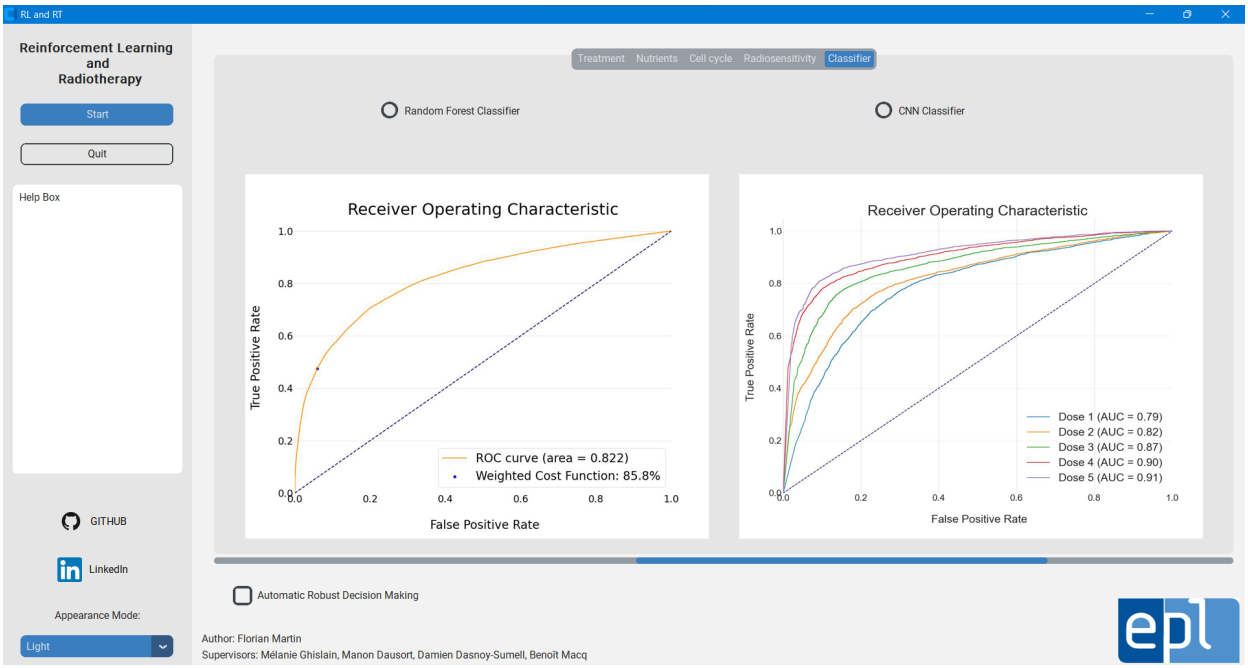


Figure 9.14: Classifier page of the GUI.

Figure 9.14 showcases the classifier page of the GUI. This dedicated interface empowers users to select the type of classifier to be used in the automated robust decision-making process. For a clear visual understanding of classifier performance, Receiver Operating Characteristic (ROC) curves are prominently displayed for both the Random Forest (RF) and Convolutional Neural Network (CNN) classifiers. This offers an intuitive visual comparison of the performance and effectiveness of these classification algorithms within the decision-making process.

9.4.6 Simulation page

After the user has meticulously configured all the necessary parameters and selected the desired treatment, the simulation can be initiated from the top left corner of the GUI. This interface will subsequently prepare the environment and load the previously chosen classifier. Please note, this process may take a few seconds to fully update and start the simulation, ensuring that all modifications have been correctly incorporated.

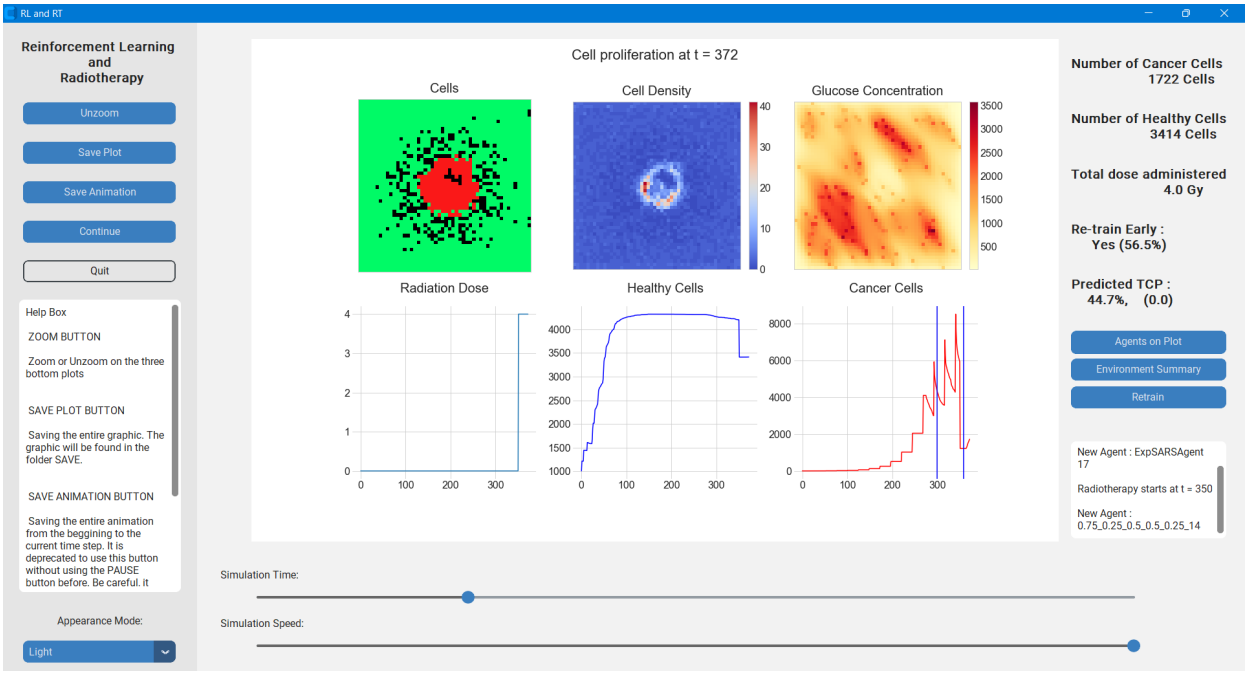


Figure 9.15: Simulation page of the GUI.

Figure 9.15 showcases the simulation page of the GUI, packed with numerous functionalities. The left sidebar allows the user to zoom in/out on the plot of the radiation dose, healthy cells, and cancer cells. There are options for saving all six plots or an animation of the simulation, which runs from the initial step to the conclusion. A button to pause or resume the simulation provides added control, and two interactive sliders at the bottom enhance user involvement. The first slider controls simulation time, permitting the user to rewind or fast-forward the simulation (note that the simulation unfolds in real-time). The second slider adjusts simulation speed, offering five distinct settings to either decelerate or accelerate the simulation.

On the right-hand side, real-time updates on the number of cancer cells, healthy cells, and the total administered radiation dose are available. The label "Re-train Early" represents the output of the chosen classifier, which processes the cancer cells array (or tumor image) and generates a binary output predicting if the simulation will achieve a TCP of 100%. If it reads "Yes", it suggests a transition to a more robust agent. The adjacent percentage indicates the classifier's confidence in its prediction.

Also featured on the right-hand side is a predicted TCP, computed by the TCP regressor described in Section 7.2.1.

Three additional buttons offer more interactivity: "Agents on Plot" denotes vertical lines on the cancer cells plot, indicating moments when the simulation switched to a more robust agent. Clicking this button again will remove the lines. The "Environment Summary" button provides a refresher on the user-selected parameters. The "Retrain" button enables manual agent switching, particularly useful when the "Automatic Robust Decision Making" option on the treatment page is not activated. A click on the "Retrain" button opens a new window, as depicted in Figure 9.16.

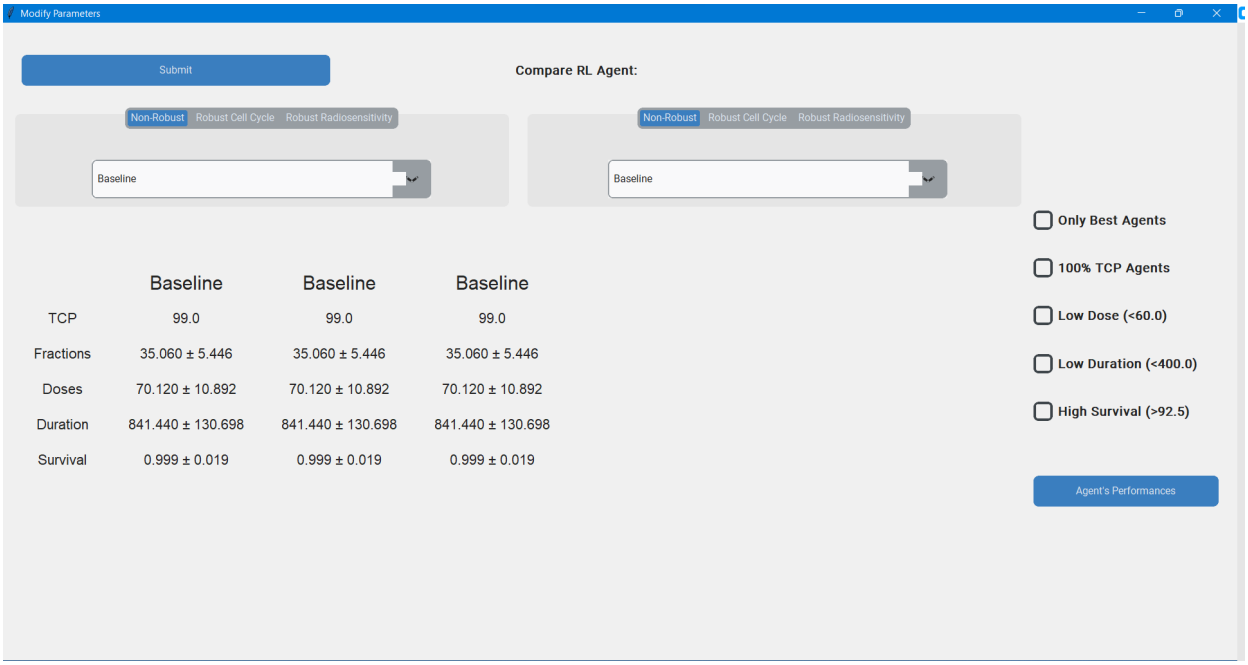


Figure 9.16: Retrain page of the GUI.

The Retrain page bears resemblance to the treatment page, serving a crucial role in agent selection during the course of the simulation. After selecting a new agent, the user can integrate it into the ongoing simulation by clicking the "Submit" button.

Bibliography

[1] Ammar Jalalimanesh et al. "Simulation-based optimization of radiotherapy: Agent-based modeling and reinforcement learning". en. In: *Mathematics and Computers in Simulation*. Biomath 2014 and Biomath 2015 133 (Mar. 2017), pp. 235–248. ISSN: 0378-4754. DOI: 10.1016/j.matcom.2016.05.008. URL: https://www.sciencedirect.com/science/article/pii/S0378475416300878 (visited on 02/08/2023).

[2] Nicole O'Neil. "An Agent Based Model of Tumor Growth and Response to Radiotherapy". In: *Theses and Dissertations* (May 2012). DOI: https://doi.org/10.25772/ZWX8-SN12. URL: https://scholarscompass.vcu.edu/etd/2831.

[3] Richard S Sutton and Andrew G Barto. "Reinforcement Learning: An Introduction". en. In: ().

[4] *Markov property*. en. Page Version ID: 1084143539. Apr. 2022. URL: https://en.wikipedia.org/w/index.php?title=Markov_property&oldid=1084143539 (visited on 10/28/2022).

[5] blackburn. *Reinforcement Learning : Markov-Decision Process (Part 2)*. en. May 2022. URL: https://towardsdatascience.com/reinforcement-learning-markov-decision-process-part-2-96837c936ec3 (visited on 10/29/2022).

[6] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. arXiv:1312.5602 [cs]. Dec. 2013. DOI: 10.48550/arXiv.1312.5602. URL: http://arxiv.org/abs/1312.5602 (visited on 02/15/2023).

[7] Vincent Francois-Lavet et al. "An Introduction to Deep Reinforcement Learning". In: *FNT in Machine Learning* 11.3-4 (2018). arXiv:1811.12560 [cs, stat], pp. 219–354. ISSN: 1935-8237, 1935-8245. DOI: 10.1561/2200000071. URL: http://arxiv.org/abs/1811.12560 (visited on 02/14/2023).

[8] Michael Joiner and Albert van der Kogel, eds. *Basic clinical radiobiology*. en. 4th ed. London: Hodder Arnold, 2009. ISBN: 978-0-340-92966-7.

[9] E. P. Malaise, N. Chavaudra, and M. Tubiana. "The relationship between growth rate, labelling index and histological type of human solid tumours". en. In: *European Journal of Cancer (1965)* 9.4 (Apr. 1973), pp. 305–312. ISSN: 0014-2964. DOI: 10.1016/0014-2964(73)90099-6. URL: https://www.sciencedirect.com/science/article/pii/0014296473900996 (visited on 03/20/2023).

[10] International Atomic Energy Agency. *Radiotherapy in Cancer Care: Facing the Global Challenge*. en. Text. ISBN: 9789201150134 Publication Title: Radiotherapy in Cancer Care: Facing the Global Challenge. International Atomic Energy Agency, 2017, pp. 1–544. URL: https://www.iaea.org/publications/10627/radiotherapy-in-cancer-care-facing-the-global-challenge (visited on 03/22/2023).

[11] Joseph Ragaz et al. "Locoregional Radiation Therapy in Patients With High-Risk Breast Cancer Receiving Adjuvant Chemotherapy: 20-Year Results of the British Columbia Randomized Trial". In: *JNCI: Journal of the National Cancer Institute* 97.2 (Jan. 2005), pp. 116–126. ISSN: 0027-8874. DOI: 10.1093/jnci/djh297. URL: https://doi.org/10.1093/jnci/djh297 (visited on 03/15/2023).

[12] J. F. Ward. "The yield of DNA double-strand breaks produced intracellularly by ionizing radiation: a review". eng. In: *Int J Radiat Biol* 57.6 (June 1990), pp. 1141–1150. ISSN: 0955-3002. DOI: 10.1080/09553009014551251.

[13] B. Marples, B. G. Wouters, and M. C. Joiner. "An association between the radiation-induced arrest of G2-phase cells and low-dose hyper-radiosensitivity: a plausible underlying mechanism?" eng. In: *Radiat Res* 160.1 (July 2003), pp. 38–45. ISSN: 0033-7587. DOI: 10.1667/rr3013.

[14] Steven R. McDougall, Alexander R. A. Anderson, and Mark A. J. Chaplain. "Mathematical modelling of dynamic adaptive tumour-induced angiogenesis: Clinical implications and therapeutic targeting strategies". en. In: *Journal of Theoretical Biology* 241.3 (Aug. 2006), pp. 564–589. ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2005.12.022. URL: https://www.sciencedirect.com/science/article/pii/S0022519305005564 (visited on 02/08/2023).

[15] Alexander R. A. Anderson et al. "Microenvironment driven invasion: a multiscale multimodel investigation". en. In: *J. Math. Biol.* 58.4 (Apr. 2009), pp. 579–624. ISSN: 1432-1416. DOI: 10.1007/s00285-008-0210-2. URL: https://doi.org/10.1007/s00285-008-0210-2 (visited on 02/08/2023).

[16] Dirk Drasdo. "Center-based Single-cell Models: An Approach to Multi-cellular Organization Based on a Conceptual Analogy to Colloidal Particles". en. In: *Single-Cell-Based Models in Biology and Medicine*. Ed. by Alexander R. A. Anderson, Mark A. J. Chaplain, and Katarzyna A. Rejniak. Mathematics and Biosciences in Interaction. Basel: Birkhäuser, 2007, pp. 171–196. ISBN: 978-3-7643-8123-3. DOI: 10.1007/978-3-7643-8123-3_8. URL: https://doi.org/10.1007/978-3-7643-8123-3_8 (visited on 02/08/2023).

[17] Michael Douglass, Eva Bezak, and Scott Penfold. "Development of a randomized 3D cell model for Monte Carlo microdosimetry simulations". In: *Medical physics* 39 (June 2012), pp. 3509–19. DOI: 10.1118/1.4719963.

[18] Kieran Smallbone et al. "Metabolic changes during carcinogenesis: Potential impact on invasiveness". en. In: *Journal of Theoretical Biology* 244.4 (Feb. 2007), pp. 703–713. ISSN: 0022-5193. DOI: 10.1016/j.jtbi.2006.09.010. URL: https://www.sciencedirect.com/science/article/pii/S0022519306004115 (visited on 02/08/2023).

[19] Grégoire Moreau et al. "Reinforcement Learning for Radiotherapy Dose Fractioning Automation". In: *Biomedicines* 9 (Feb. 2021), p. 214. DOI: 10.3390/biomedicines9020214.

[20] Geoffrey M. Cooper. "The Eukaryotic Cell Cycle". en. In: *The Cell: A Molecular Approach. 2nd edition* (2000). Publisher: Sinauer Associates. URL: https://www.ncbi.nlm.nih.gov/books/NBK9876/ (visited on 02/10/2023).

[21] Nicole O'Neil. "An Agent Based Model of Tumor Growth and Response to Radiotherapy". en. In: ().

[22] Paul Held. "Determination of Cell Cycle Time". en. In: ().

[23] Vassilis P. Antipas et al. "A spatio-temporal simulation model of the response of solid tumours to radiotherapy in vivo: parametric validation concerning oxygen enhancement ratio and cell cycle duration". en. In: *Phys. Med. Biol.* 49.8 (Mar. 2004), p. 1485. ISSN: 0031-9155. DOI: 10.1088/0031-9155/49/8/008. URL: https://dx.doi.org/10.1088/0031-9155/49/8/008 (visited on 02/08/2023).

[24] M. Richard et al. "A mathematical model of response of cells to radiation". en. In: *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*. Computer Simulation of Radiation Effects in Solids 255.1 (Feb. 2007), pp. 18–22. ISSN: 0168-583X. DOI: 10.1016/j.nimb.2006.11.077. URL: https://www.sciencedirect.com/science/article/pii/S0168583X06010573 (visited on 02/12/2023).

[25] L. H. Gray et al. "The concentration of oxygen dissolved in tissues at the time of irradiation as a factor in radiotherapy". eng. In: *Br J Radiol* 26.312 (Dec. 1953), pp. 638–648. ISSN: 0007-1285. DOI: 10.1259/0007-1285-26-312-638.

[26] Katy McLaughlin Ph.D. *Aerobic Respiration - The Definitive Guide*. en-US. Oct. 2016. URL: https://biologydictionary.net/aerobic-respiration/ (visited on 03/07/2023).

[27] Barbara Muz et al. "The role of hypoxia in cancer progression, angiogenesis, metastasis, and resistance to therapy". In: *Hypoxia (Auckl)* 3 (Dec. 2015), pp. 83–92. ISSN: 2324-1128. DOI: 10.2147/HP.S93413. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5045092/ (visited on 03/07/2023).

[28] Michael Baumann and Cordula Petersen. "TCP and NTCP: a basic introduction". eng. In: *Rays* 30.2 (2005), pp. 99–104. ISSN: 0390-7740.

[29] *Radiation Therapy Side Effects - NCI*. en. cgvArticle. Archive Location: nciglobal,ncienterprise. Jan. 2018. URL: https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy/side-effects (visited on 04/19/2023).

[30] *Side effects of radiotherapy*. en. Section: conditions. Oct. 2018. URL: https://www.nhs.uk/conditions/radiotherapy/side-effects/ (visited on 04/19/2023).

[31] Christian Szegedy et al. *Going Deeper with Convolutions*. arXiv:1409.4842 [cs]. Sept. 2014. DOI: 10.48550/arXiv.1409.4842. URL: http://arxiv.org/abs/1409.4842 (visited on 06/01/2023).

[32] Kaiming He et al. *Deep Residual Learning for Image Recognition*. arXiv:1512.03385 [cs]. Dec. 2015. DOI: 10.48550/arXiv.1512.03385. URL: http://arxiv.org/abs/1512.03385 (visited on 06/01/2023).

[33] Ramprasaath R. Selvaraju et al. "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization". In: *Int J Comput Vis* 128.2 (Feb. 2020). arXiv:1610.02391 [cs], pp. 336–359. ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-019-01228-7. URL: http://arxiv.org/abs/1610.02391 (visited on 06/01/2023).

[34] *Radiation Therapy for Cancer - NCI*. en. cgvArticle. Archive Location: nciglobal,ncienterprise. Apr. 2015. URL: https://www.cancer.gov/about-cancer/treatment/types/radiation-therapy (visited on 05/19/2023).

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**