UCLouvain

epl

École polytechnique de Louvain

Combining biological proliferation function and proton therapy for a treatment plan based on deep reinforcement learning

Author: **Amanpreet SINGH**
Supervisor: **Benoit MACQ**
Readers: **John LEE, Paul DESBORDES, Alice BORBATH**
Academic year 2020–2021
Master [120] in Electrical Engineering

Abstract

Cancer is among the leading causes of death worldwide. The early diagnosis of the tumor is crucial since the treatment strategy will depend on the advancement of the disease. Most of the time, external beam radiotherapy is a common treatment strategy used in clinics to treat the tumor. Innovations in the field of machine learning in conjuncture with hardware acceleration are being used in industrial applications such as autonomous vehicles, finance, healthcare, etc.

This work aims to combine the proliferation of biological cells in conjuncture with proton therapy in a computer simulation. Deep reinforcement learning agents are then trained and used in this environment to propose a treatment plan to cure cancer.

The first part of this thesis focuses on reviewing theoretical prerequisites to understand the methods used in the second part for the simulation. The third part is dedicated to the training of deep reinforcement learning agents in the tumoral environment. Agents using algorithms such as DQN and DDPG are trained and then used to get a treatment plan. As a final note, the results obtained from the simulation and potential further improvements are discussed.

Acknowledgment

Throughout the writing of my Master's thesis, I have received a tremendous amount of expertise, guidance, and support.

First and foremost, I would like to thank my supervisor, Professor Benoit Macq, whose insightful expertise and feedback allowed me to push my work to such an extent.

I want to acknowledge Paul Desbordes for his constructive comments, which helped improve the simulation quality.

I would also like to express my deepest gratitude towards Professor John Lee and Alice Borbath for giving time and effort to read the thesis.

Finally, a special thank goes to my family for their love and who always pushed me forward to be successful.

Health is not valued till sickness comes.

- Thomas Fuller

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial intelligence
DDPG	Deep Deterministic Policy Gradient
DQN	Deep Q-Networks
DRL	Deep Reinforcement Learning
EM	Electromagnetic
GLOBOCAN	Global Cancer Observatory
LQ	Linear-quadratic
LTE	Linear Energy Transfer
MCS	Monte Carlo Simulation
MDP	Markov Decision Process
ML	Machine learning
NTCP	Normal Tissue Control Probability
OAR	Organ At Risk
OMF	Oxygen Modification Factor
POMDP	Partially Observable Markov Decision Process
RBE	Relative Biological Effectiveness
RL	Reinforcement learning
SF	Surviving Fraction
SOBP	Spread-Out Bragg Peak
TCP	Tumor Control Probability
TV	Target Volume

Glossary

Ten-Year Survival	The probability for an individual to survive at least 10 year after cancer diagnosis
Palliative care	Mitigate suffering among patients with serious, complex illness (end-of-life care)
Phantom	A mass that approximates tissues in its physical properties

Introduction

Generalities on cancer

One in two men and one in three women will be diagnosed with cancer in the U.S [15]. Unfortunately, it is not the end of the story. In 2020, the *GLOBOCAN* estimated 10 million death worldwide due to cancer [16], which makes it one of the most deadly disease. On the one hand, prostate cancer is the most diagnosed cancer for men, and on the other hand, breast cancer for women. Both in men and women, lung and colon cancers are common. *Ten-Year Survival* varies between gender and cancer type, for instance, it is estimated at 10 % for lung cancer [17].

At the most superficial level, cancer cells are cells that have lost the ability to follow the standard control that the body exerts on all cells. They continue to proliferate anarchically and may also spread to form a mass called a tumor. There are two types of tumor, benign and malignant. As the name suggests, a malignant tumor can invade adjacent organs and spread to other tissues in contrast with benign mass, which develops locally without metastasis. It is essential to understand that cancer that occurs in one individual is very different than other individual which makes it difficult to treat.

The early diagnosis of cancer is crucial since once it is diagnosed, the treatment strategy will depend on the advancement of the disease (Figure 1). The local control in Figure 1 refers to the probability of controlling the tumor, i.e., annihilate cancer cells, which

decreases with the stage of the patient. One should note that external beam radiotherapy involves approximately 50% of cancer treatments, which makes it one of the most used methods by cancer centers and hospitals.

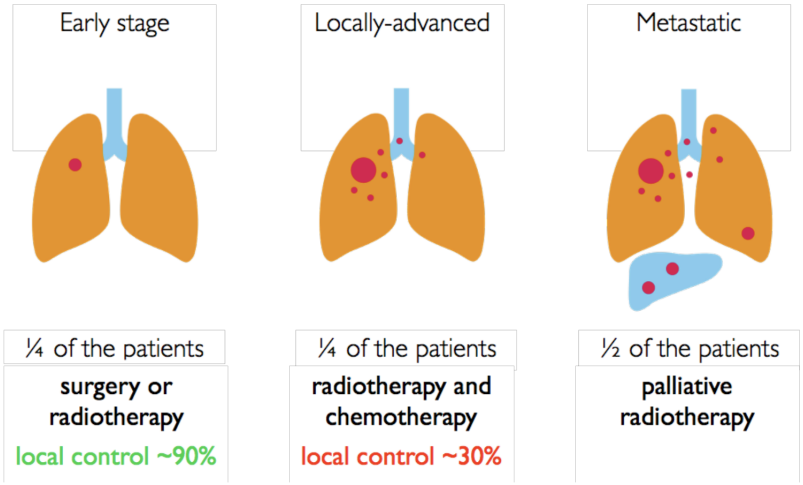

Figure 1: Cancer stage and treatment strategy [1]

Basics of radiotherapy

In the realm of cancer treatment, radiotherapy plays a key role. The basic mechanism of radiotherapy is using ionizing radiation, i.e., an electromagnetic wave carrying enough energy to ionize matter. Radiation such as X-rays are used in radiotherapy (Figure 2).

The absorbed dose by a *phantom* is the amount of energy absorbed per unit mass through the process of ionization and is expressed in Gray [Gy] :

$$1\ Gy = 1\ \frac{J}{Kg} = 1\ \frac{m^2}{s^2} \tag{1}$$

The process of ionization in the body will damage the DNA. Hence, it will also damage the healthy and cancer cells. The convenience in this process is that the normal cells have better DNA repair capabilities than cancer cells [1]. This allows the opening of a therapeutic window, meaning that there is a certain amount of radiation that impact the disease and still allows healthy tissues to recover. Unfortunately, DNA can repair badly in

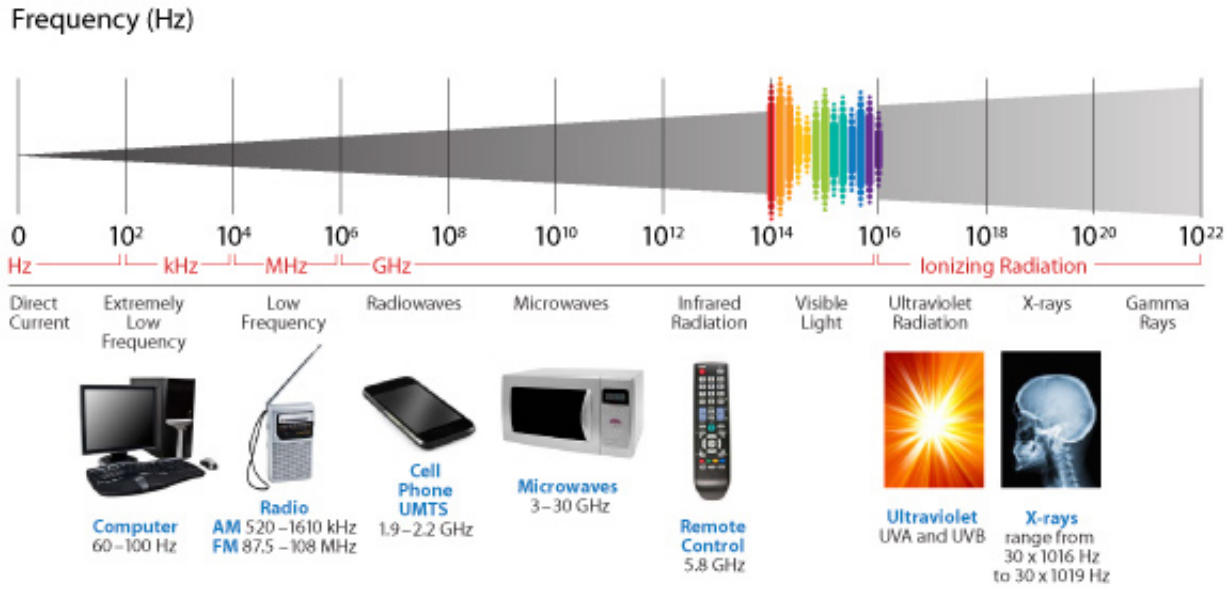

Figure 2: Electromagnetic spectrum [2]

some cases (toxicity), and can lead to mutation and secondary cancer. The whole deal of radiotherapy is to maximize TV dose while minimizing the OAR dose. This is also refereed as maximizing the TCP and minimize $NTCP$. Figure 3 allows to understand that there is a trade-off to be made in terms of the absorbed dose in order to fulfill the constraint stated before.

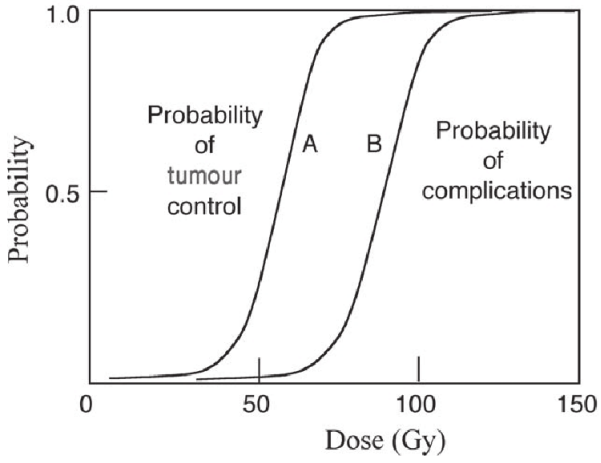

Figure 3: TCP and NTCP in function of the dose [3]

Improving radiotherapy efficiency

There are many ways to improve the therapeutic ratio [1] such as

- Radiosensitization

  Tumor cells have a reduced amount of oxygen, i.e., hypoxia, w.r.t normal cells. This leads to a significant limitation for radiotherapy treatment since hypoxic microenvironments are much more resistant to radiation in contrast to standard oxygen microenvironment [18]. Radiosensitizers are agents that improve the effectiveness of radiotherapy by enhancing the killing of tumor cells [18]. The effectiveness is measured by computing the enhancement ratio (ER) [19] :

$$ER = \frac{\text{Radiation dose for a given biological effect without sensitizer}}{\text{Radiation dose to achieve the same effect with sensitizer}} \tag{2}$$

  The presence of oxygen increases the probability of producing free radicals that are toxic to the cells. The Oxygen Enhancement Ratio (OER) is defined as the ratio of doses under hypoxic to aerated conditions having the same biological effect (Figure 4).

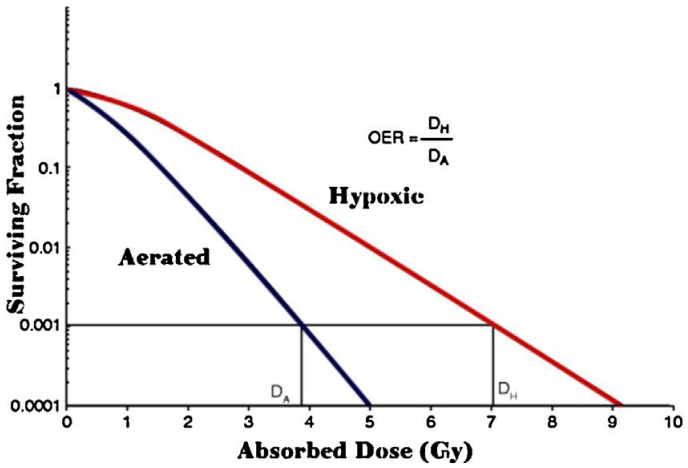

Figure 4: Effect of hypoxia upon cell survival fraction [4]

Therefore, the knowledge of tumor oxygen distribution before the treatment is crucial for the treatment outcome.

- Dose sculpting

  Dose sculpting is referred to as ballistics. It can be achieved with various methods such as modulation of the beam intensity and by increasing the number of beam angles, allowing to target the tumor more accurately [1]. In practice, techniques such as pencil beam scanning is widely used with protontherapy.

- Fractionation

  A standard radiotherapy treatment plan aims to provide 2 Gy per day during 35 days [6]. The treatment plan works by fractionating the total dose, which aims to take advantage of the fact that a smaller administered dose will result in increased cell survival (healthy cells repair capabilities are better than tumor cells) in contrast to administrating the total dose in a single shot (Figure 5). This will allow to minimize the $NTCP$ without degrading the TCP since the dose is cumulative.

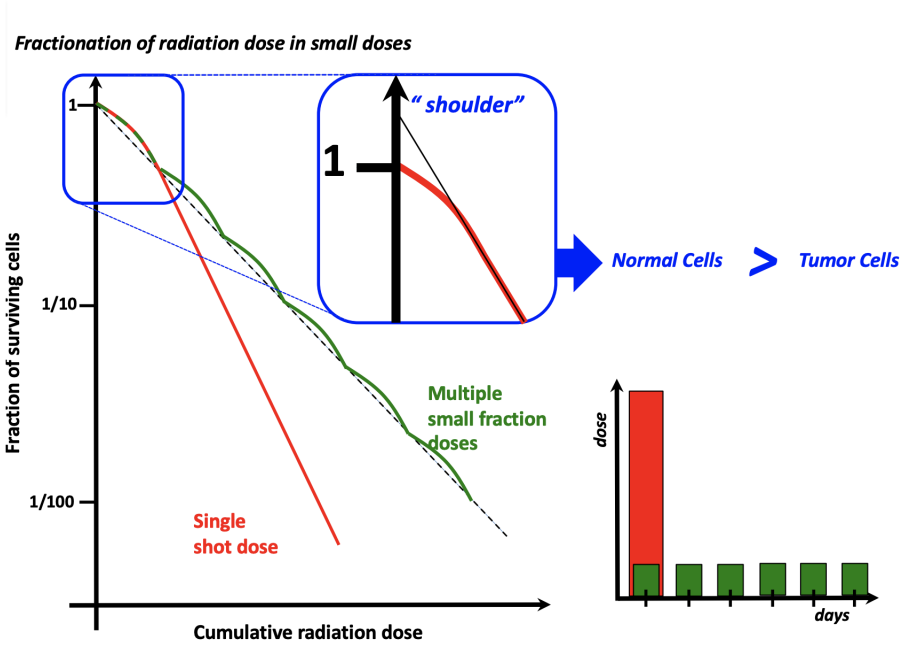

Figure 5: Fractionation of the total dose [1]

Goals and plan of the work

The main focus of this work is to combine the following ingredients in a computer simulation : the proliferation of biological cells, irradiation with a beam of protons, and reinforcement learning. More specifically, the specifications are the following :

- an implementation of an environment which represents cells proliferation (cancer and healthy) on a 3D grid.
- an implementation of actions, such as irradiation of the 3D grid with a modulated beam of protons, (x,y) positioning of the beam, and variable rest time after the irradiation.
- a reinforcement learning agent whose goal will be to find a treatment plan based on rewards functions.

To carry out this work successfully, I will first discuss the theoretical prerequisite and relevant literature to understand the approach used to implement the specifications. The second chapter is about the materials and methods used for the simulation, based on the first chapter. The third chapter focuses on the results obtained from the materials and methods detailed in the second chapter and a wrap-up in the last chapter.

One should note that this work is mainly based on the excellent work of Grégoire Moreau [6], a former student of Université Catholique de Louvain, in Belgium. I hope that my work will contribute, somehow, to academia and even in clinics with further development of this simulation.

Link to the GitHub repository :

https://github.com/amanpreetsingh-BE/DeepRL-Protontherapy

Chapter 1

Theoretical background

The first step is to look at relevant literature since the main goal is to combine the proliferation of biological cells with proton therapy and propose a treatment plan based on reinforcement learning in a computer simulation. This approach is called in silico in contrast with in vivo and in vitro experiments. Moreover, the usage of deep reinforcement learning in the purpose of this work makes almost mandatory the usage of a computer simulation. Indeed, the agent is first trained on a computer simulation since it is too dangerous to train on a real patient. On the one hand, this method is an ethical and cost-effective method but in the other hand, it is less accurate and the mapping to real situation is approximate. Ideally, the simulation need to be as close to real life in order to achieve best outcome for the treatment. Unfortunately, the complexity of the model is limited by the usage of deep reinforcement learning, since the latter one requires millions of learning examples in order to converge [6].

1.1 State of the art in computational modeling

1.1.1 Cell-based computational models

Metzcar et al. [5] proposed a review of cell-based computational modeling in cancer biology (Figure 1.1)

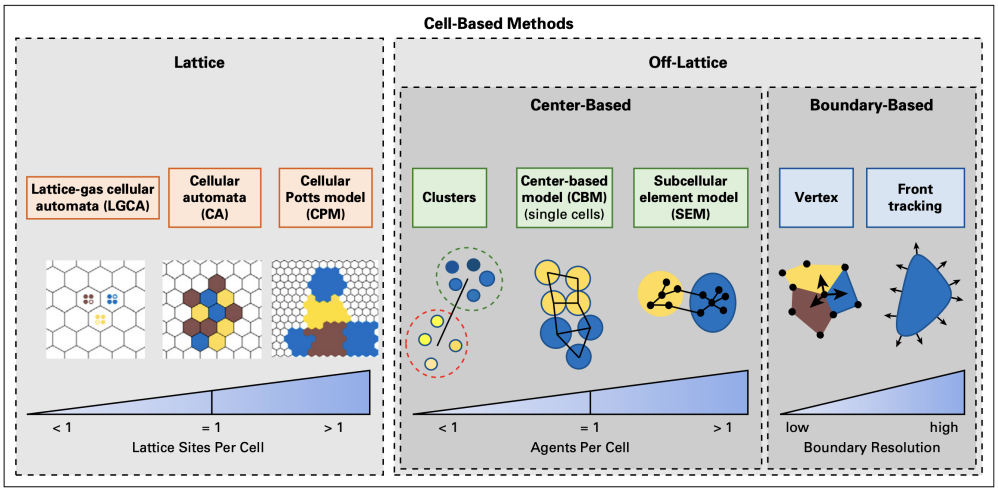

Figure 1.1: Cell-Based Computational Modeling [5]

Basically, there is two approaches :

- Lattice-based methods are the approaches where the cells are placed on a grid. Their spatial resolution further categorizes them :
  - Cellular automata : each site on the grid contains a single cell.
  - Lattice-gas cellular automata : each site on the grid can contain multiple cells.
  - Cellular Potts model : multiple sites on the grid are used to model a cell, including its shape.
- Off-lattice methods are the approaches where the cells are in free positions :
  - Center-based : focus on the center of mass or volume for each cell.
  - Boundary-based : model cell morphology in more significant details.

The main problem with off-lattice methods is that they are very computationally intensive and make it not suitable for *DRL*.

1.1.2 Agent-based simulation

Off-lattice methods are too extensive for deep reinforcement learning purposes. Simulation-based on less complex model and suitable for *DRL* have been presented in the literature. Moreau's simulation [6] is based on O'Neil [20] and Jalalimanesh et al. [21] works. It is slightly modified to make the simulation more realistic and will be the basis for this work. It is an agent-based simulation, the idea that one can model an environment (e.g biological proliferation) using agents. An agent is an autonomous individual element (e.g biological cell) with properties and actions in a computer simulation. The dynamics of the model is dictated by local rules, meaning that the agents are impacted by nearest neighbors and surrounding environments [20]. The simulation is summarized in Figure 1.2.

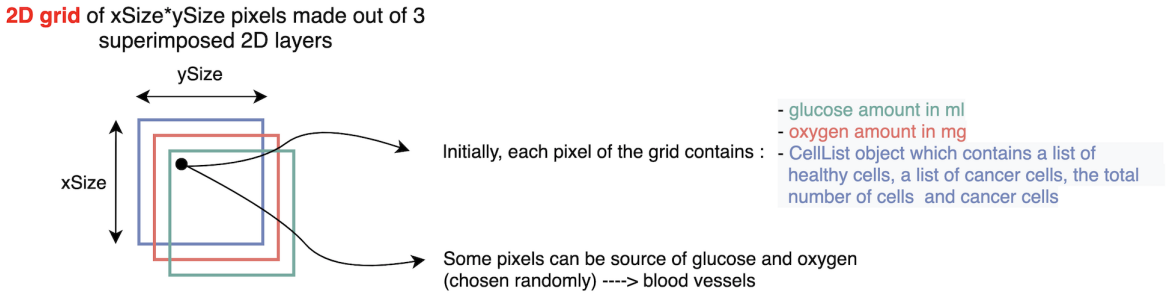

Figure 1.2: Basis of Moreau's simulation, which is a 2D grid

- The program will first initialize a 2D grid of size xSize*ySize which defines the number of pixels. A certain number of healthy and sources (glucose and oxygen) cells representing blood vessels are placed randomly on the grid and only one cancer cell is place in the center.

- Each hour or tick, the grid will go through various steps sequentially :

  – Replenishment of sources in glucose and oxygen with fixed values.

- Feed the cells, handle mitosis of cells, update the stage
- Diffuse the glucose and the oxygen to neighbors with a fixed value (diffusion rate).

In essence, that's it for the simulation, at least the biological proliferation part, but if the reader wishes to know the details and the values of parameters, it can be found in [6]. The result of biological proliferation by healthy and cancer cells is presented in Figure 1.3.

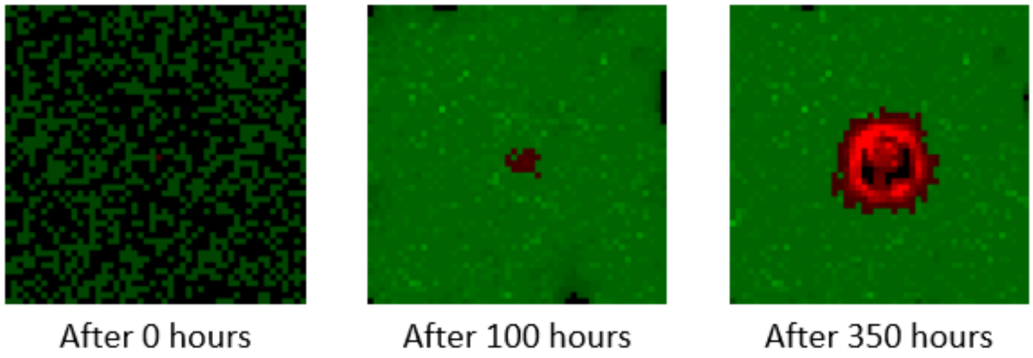

Figure 1.3: Moreau's simulation after 350 ticks [6]

1.2 The proliferation of biological cells

1.2.1 Healthy cells

The proliferation of biological cells is modeled by "The Eukaryotic Cell Cycle" [22] and is more complete than the description given in this work. In the simulation, the cell cycle is based on the description given in O'Neil [20] and can be summarized in Figure 1.4 :

The cycle of the cell (Figure 1.4) can be described by 4 stages :

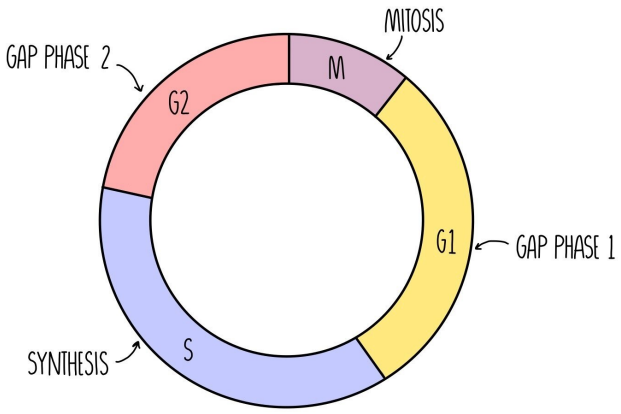

Figure 1.4: Cell cycle [7]

- M := Mitosis, i.e., cell division $\sim$ 11 hours : is the stage where the mother cell divides in order to produce two daughter cells
- G1 := Gap 1, i.e., cell growth through feeding $\sim$ 8 hours : follows mitosis and is the stage between cell division and DNA replication
- G1 := Gap 1, i.e., cell growth through feeding $\sim$ 8 hours : is the stage were DNA synthesis occurs and the amount of DNA doubles in the cell
- G2 := Gap 2, i.e., cell growth $\sim$ 1 hour : is the stage where the cell continues to grow and prepares for mitosis by producing necessary proteins

Healthy cells do not enter in the cycle, i.e no replication, due to [20] :

- Physical space (density of the cells exceeds a certain threshold).
- Nutrients availability (not enough nutrients).

Those limitations are usually checked in gap 1, and if one of the conditions is true, the cell enters in Quiescence G0 mode. In this mode, the cell remains active (feeding and growing), but there is no replication. It can happen that cells exit G0 to G1 mode if there is physical space and enough nutrients.

1.2.2 Cancer cells

Generally, the development of a tumor is due to a mutation of a single cell. The main difference between healthy and cancer cells relies on faster growth without inhibition and aggressive feeding of available nutrients even if the physical space and nutrients availability is low [20]. The reasons for mutation can be environmental and genetic factors. The tissue around the tumor can develop blood vessels in order to supply in nutrients the tumor. This phenomenon is called angiogenesis.

1.3 Radiobiology

1.3.1 Survival curve, linear energy transfer and relative biological effectiveness

Survival curve

Radiobiology is the study of biological effects produced by ionizing radiation on living organisms. Such effects can be assessed via a cell survival curve which is a plot of surviving fraction SF of cells against dose D of radiation. Figure 1.5 is a survival curve of irradiated tissue culture plotted on a linear and logarithmic scale. The relation is not linear and increasing the dose yields in smaller survival fraction, meaning that the effect of killing cells is increasing. Two points are defined to indicate the sensitivity of the cells to radiation. ED50 and ED90 are the effective dose at which 50% and 90% of the cells dies, respectively.

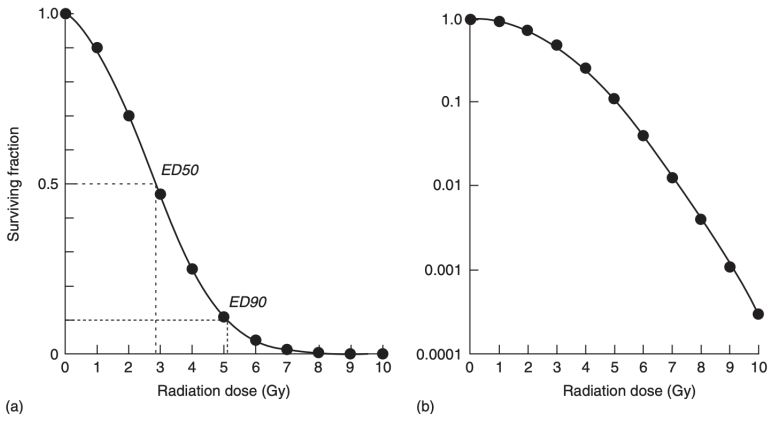

Figure 1.5: Survival curve for irradiated tissue culture, (a) linear scale (b) logarithmic scale [8]

The *SF* curve can be mathematically modeled using target theory or linear quadratic (LQ) model [8].

Linear energy transfer

Linear energy transfer (LET) describes the density of ionization in particle tracks (Figure 1.6), i.e., average radiation energy in keV (dE_{local}) given up by a charged particle to target atoms along its path (dx) through tissue [23].

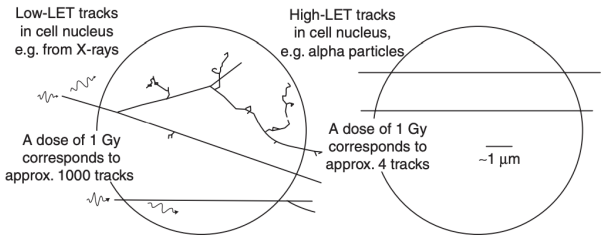

Figure 1.6: Structure of particles tracks for Low-LET and High-LET [8]

Particles such as electrons, photons or protons are low-LET and neutrons, α or light ions are high-LET [1]. Mathematically, it is defined as :

$$L = \frac{dE_{local}}{dx} \, [KeV/\mu m]$$

Relative biological effectiveness

The Relative Biological Effectiveness (RBE) is defined as the ratio of the doses required by two radiations to cause the same biological effect [8]:

$$RBE = \frac{\text{Dose of reference}}{\text{Dose of test}} \tag{1.1}$$

The widely available 250 kVp X-rays or ^{60}Co γ-rays make them suitable for reference dose to evaluate the RBE [8].

SF-LET-RBE relation

Biological effect increases (steeper SF curve) as LET value increases leading to higher values of RBE until it becomes inefficient. Figure 1.7 shows this behavior and the overkill effect, meaning that energy deposited on a cell is much higher than energy needed to kill the cell leading to reduced likelihood of killing other cells [8].

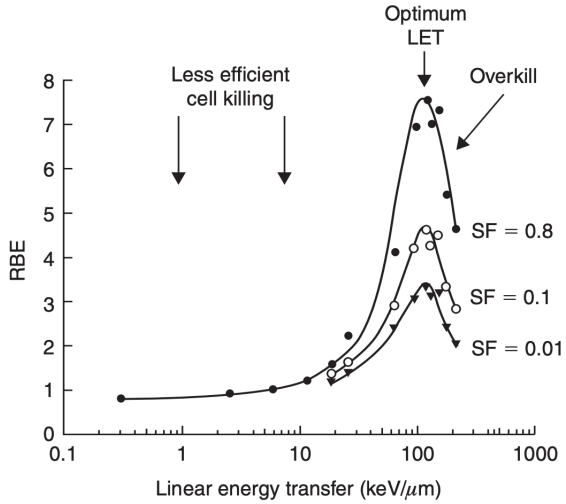

Figure 1.7: Relation of RBE to LET for various SF [8]

1.3.2 Target theory

Secondary charged particles and free radicals produced by the radiation cause damage to the DNA. Target theory proposes that to kill cells, the damaging must happen on regions of the DNA responsible for the reproductive ability of the cells. The most commons versions of this theory are [8] :

- Single-target single-hit : one hit by radiation on a single sensitive target leads to the death of the cell.

$$p(survival) = exp\left(\frac{D}{D_0}\right) \tag{1.2}$$

- Multitarget single-hit : one hit by radiation on n sensitive targets leads to the death of the cell.

$$p(survival) = 1 - \left(1 - exp\left(\frac{D}{D_0}\right)\right)^n \tag{1.3}$$

D_0 is the dose that gives an average of one hit per target. The derivations of the equations are based on Poisson distribution and can be found in [8].

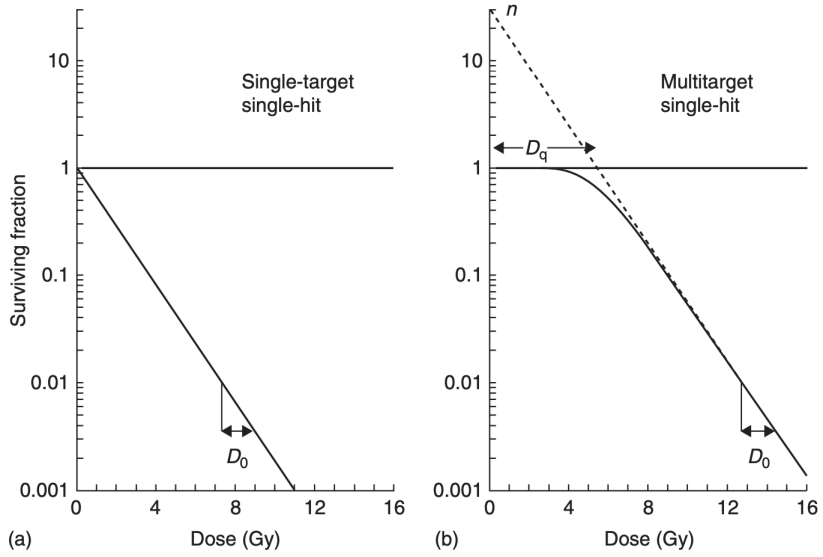

Figure 1.8: The two most common types of target theory. (a) Single-target inactivation; (b) multitarget inactivation [8]

Intuitively, the first plot (a) in Figure 1.8 describes the situation in which a cell receiving a dose greater than D_0 dies, otherwise, it survives. The second plot (b) describes

a "shouldered" curve characterized by D_q, the Quasi-threshold dose [8]. Intuitively, the presence of the shoulder can be interpreted as the result of events accumulation (radiation hitting n sensitive targets) or efficiency reduction in the repair mechanism while increasing the dose [8]. The problem with the target theory is that the presence of specific regions have not been identified. Moreover, the "flat" response for multitarget model in low doses range makes it not suitable clinically.

1.3.3 LQ model

A second-order polynomial can fit the survival curve. In order to make sure that for zero dose the SF is 100%, constant term of the polynomial is set to zero [8] :

$$-ln(S) = \alpha D + \beta D^2 \rightarrow p(survival) = exp\left(-\alpha D - \beta D^2\right) \tag{1.4}$$

The equation is constituted by a linear part αD and a quadratic one βD^2. The contribution of the linear term is equal to the contribution of the quadratic one when the dose is equal to the ratio α/β, which characterizes the shape of the curve (Figure 1.9). Therefore, the parameters α and β represent the radiosensitivity of irradiated tissues [6]. The values of those parameters can be found in the literature such as the ones proposed by Van Leeuwen et al. [24].

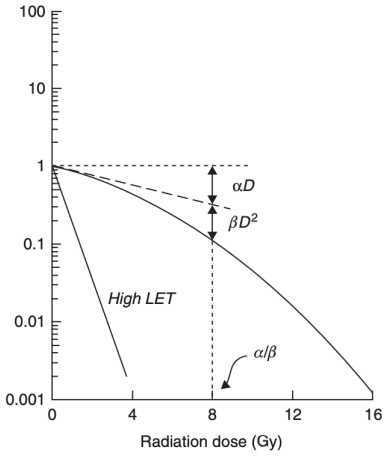

Figure 1.9: LQ model [8]

In contrast with target theory, the main advantage of this model is a better description of radiation response in the low doses range (0-3 Gy). Moreover, the LQ model is widely used in experimental and clinical radiobiology [8] but can be further expanded, such as in Jalalimanesh et al. [21]. In this paper, they used an extended LQ model proposed by Powathil et al. [25] :

$$p(survival) := S(d) = exp\left[\gamma\left(-\alpha.OMF.d - \beta(OMF.d)^2\right)\right] \tag{1.5}$$

The parameter γ represents the current stage of the cell cycle, and OMF is the oxygen modification factor representing the current oxygenation level, which is crucial since hypoxic cells are less sensitive to radiation [6]. The OMF parameter given in Powathil et al. [25] can be computed as :

$$OMF = \frac{OER(pO_2)}{OER_m} = \frac{1}{OER_m}\frac{OER_m * pO_2(x) + K_m}{pO_2(x) + K_m} \tag{1.6}$$

where $pO_2(x)$ is the dioxygen concentration in voxel x, $OER_m = 3$ is the maximum OER, and $K_m = 3mm$ is the oxygen concentration that gives an OER value of 2.

1.4 Proton therapy

1.4.1 Biological differentiation using protons

As discussed in the *introduction*, one can improve the therapeutic ratio by various methods such as dose sculpting, fractionation, biological differentiation using different particles, and radiosensitization. The main difference between proton and radiotherapy is the use of different particles to irradiate patients. In classical radiotherapy, the particles used are photons and protons for proton therapy. Protons are charged particles and considered as direct ionizing radiation, meaning that they deposit doses right away in contrast with neutral particles such as photons, which are considered as an indirect ionizing radiation [26]. Therefore, photons and protons interact differently with matter and deposit energy

differently. This means that at equal dose, the biological effect is different. Photons and protons are considered as Low-LET [1]. A constant $RBE = 1.1$ to relate proton dosimetry is used clinically, even if in practice, the value is not constant [27].

1.4.2 Interaction of protons with matter

Protons interact with matter through several mechanisms [9]:

- (a) Inelastic Coulomb interaction with atomic electrons
  - Collision between incident proton and an atomic electron
  - Incident proton loses some kinetic energy quasi-continuously and it is transferred to target electron allowing it to exit the atom at a certain angle.
  - Incident proton continues in straight line, i.e no deflection since its rest mass is much higher than the rest mass of electrons.
- (b) Elastic Coulomb scattering with atomic nucleus
  - Incident proton undergoes a repulsive force exerted by target atomic nucleus
  - There is no loss energy loss but the incident proton is deflected due to the large mass of the nucleus
- (c) Non-elastic nuclear interaction
  - Nuclear interaction between incident proton and target atomic nucleus
  - Secondary particles such as protons, heavier ions, neutrons and gamma rays are generated by the targeted nucleus.
  - The primary proton is removed from the beam.

The dominating interaction is the inelastic Coulomb interaction with atomic electrons, which is the most important clinically since energy loss determines the range in patient and cancer cells killing ability [28].

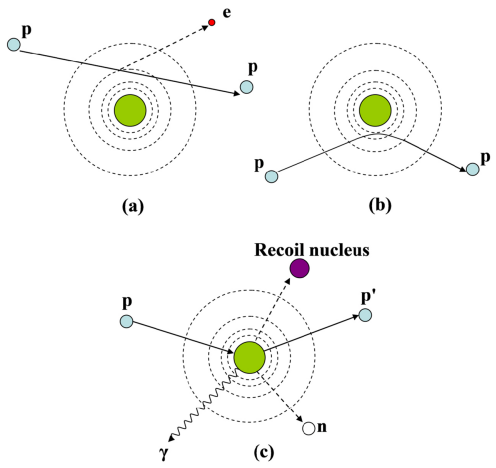

Figure 1.10: Interactions proton-matter [9]

Energy loss rate

The energy loss rate of charged particles, which is also called stopping power (SP), is defined as the average energy loss, dE, per unit distance, dx, along the track of the charged particle. This energy loss rate is expressed in $[MeV/cm]$. A more convenient way to express the SP is to remove the dependency of the mass density. This is referred as mass stopping power [9] :

$$\frac{S}{\rho} = -\frac{1}{\rho}\frac{dE}{dx}\left[\frac{MeV.cm^2}{g}\right] \tag{1.7}$$

Depth-dose distribution

Bethe (1930) and Bloch (1933) provided a more complete formula of mass stopping power (Equation 1.7) by taking into account quantum mechanical effect [9].

$$\frac{S}{\rho} = -\frac{1}{\rho}\frac{dE}{dx} = 0.1535\frac{z^2}{\beta^2}\left[27.675 + 2\ln\frac{\beta^2}{1-\beta^2} - 2\beta^2 - \delta - 2\ln(I) - 2\frac{C}{Z}\right]\frac{Z}{A} \tag{1.8}$$

The parameters of Equation 1.8 are speed of light c, velocity of projectile v, atomic number of projectile z, atomic number of target atom Z, mass number of target atom A, mean excitation potential of absorbing material I, density correction term δ, Lorentz factor $\beta = \frac{v}{c}$ and shell effect $\frac{C}{Z}$

The most important observation of Equation 1.8 is that energy loss is proportional to the inverse square of the proton's velocity, meaning the more proton slows down, the more it releases quadratic energy. Moreover, there is no dependency on projectile mass. Therefore, the peak of a proton depth-dose distribution (energy absorbed while traveling matter) can be explained by this relation. The peak has a name and is called a Bragg peak.

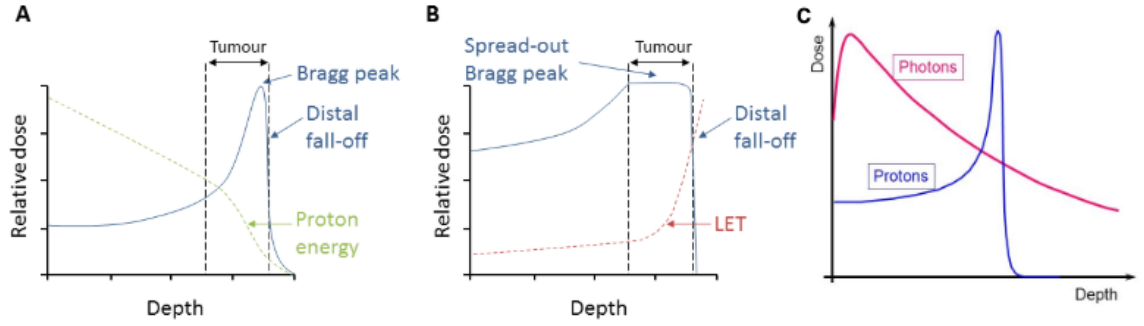

Figure 1.11: An unmodulated Bragg peak produced by a proton beam, (B) Spread-out Bragg peak (*SOBP*) from several modulated proton beams [10]. Difference between photon and proton in depth-dose distribution [11]

- Plot A of Figure 1.11 displays a typical depth-dose distribution of a proton. As the proton loses energy, the deposited dose increases forming the Bragg peak. The location and height of the peak mainly depend on stopping power and energy straggling [9]

- Plot B of Figure 1.11 displays a Spread-out Bragg peak (*SOBP*) which can be produced by using a beam of protons with various initial energies. In contrast with a single Bragg peak, a *SOBP* allows irradiating the entire depth of a large tumor. Note that the (*LET*) is increasing and becomes maximum at distal fall-off of the Bragg peak.

- Plot C of Figure 1.11 displays a typical depth-dose distribution of proton in comparison with photons. In conventional radiotherapy, most of the energy is absorbed in the surface of the patient. Moreover, energy is also absorbed behind the tumor. Therefore, the main advantage of protons over photons is the ability to spare the (OAR) and surrounding tissues close to the tumor.

1.4.3 Dose calculation

Dose calculation allows estimating the dose distribution delivered to the patient given a configuration of the beam. This allows us to assess the quality of the treatment since we want to maximize the dose delivered to the (TV) and minimize it to the (OAR). Therefore, one can also optimize beam configuration, i.e., the treatment plan using dose calculation.

Two types of dose calculation algorithms :

- Analytical algorithms, for instance image processing techniques. It is quite approximate since there is a lot of physical properties that are not simulated but there is the advantage of being fast.

- Monte Carlo simulations, where one simulate the whole physical process. Basically we simulate one particle and we simulate each interaction that it will undergo inside the patient geometry. It is the most accurate method but it is slower.

Currently, Monte Carlo methods begin to be used in clinic since the computational power increases. Monte Carlo is a numerical method based on the random sampling of the numbers to compute a result. This allows to solve very complex problems without the need of large memory and computation time. The drawback is that the result is affected by statistical uncertainties, which decreases with the number N of simulated events.

1.5 Reinforcement learning

1.5.1 Introduction to RL

The exponential growth of data [29] (text, pictures, music, etc.) justify the need of automated systems that can learn from data and more importantly adapt to the change at the input of the system. In the literature, some terms are used interchangeably but one should not confuse artificial intelligence (AI), machine learning (ML) and deep learning (DL). At the highest level, AI refers to leveraging computers to mimic the problem solving and decision making of the human mind. ML is a subset of AI that is more focused on the use of self learning algorithms that derives knowledge from data to predict outcomes. Finally, DL is a subset of ML refereed as a scalable version of ML which deals with big dataset (Figure 1.12).

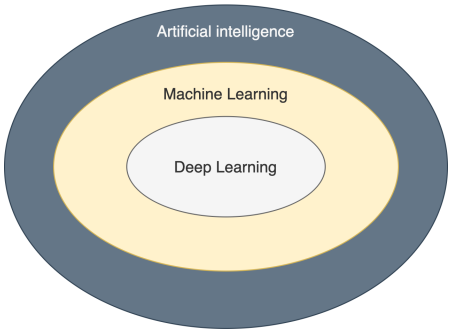

Figure 1.12: Artificial Intelligence, Machine Learning and Deep Learning

In order to introduce reinforcement learning (RL), let's first figure out how it fits in the realm of ML (Figure 1.13) :

- Supervised learning is task driven (regression and classification).
- Unsupervised learning is data driven (dimensionality reduction and clustering).
- Reinforcement learning is goal-oriented (agents act in an environment and learn from mistakes through trial and error in order to reach a goal).

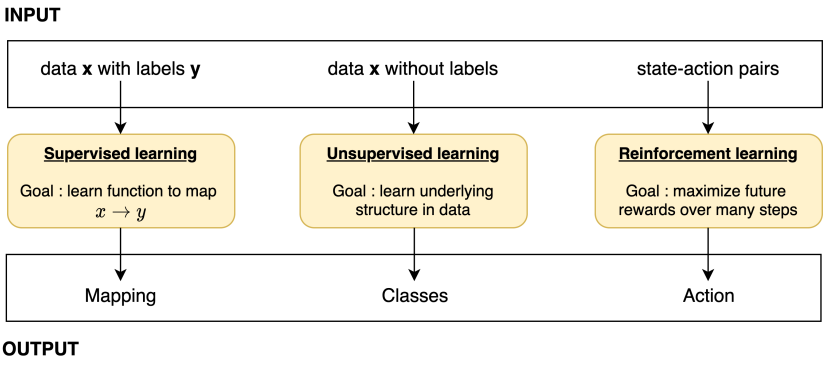

Figure 1.13: Types of machine learning

1.5.2 Key concepts and terminology

Agent-environment interface

In order to better understand the mathematical framework behind RL, let's first review key concepts and notations used throughout this work. As a computational framework, RL relies on discrete time steps. At each time step $t = 0, 1, 2, ...$, an agent receives observation O_t and reward R_t from the environment and based on this, the agent takes action A_t. Time is then incremented and the environment is transitioned to a new state S^e_{t+1}. The agent receives a new reward R_{t+1}, observes O_{t+1} and so on. The sequence of state-action-reward from an initial state to a terminal state is called an episode (trajectory if there is no terminal state).

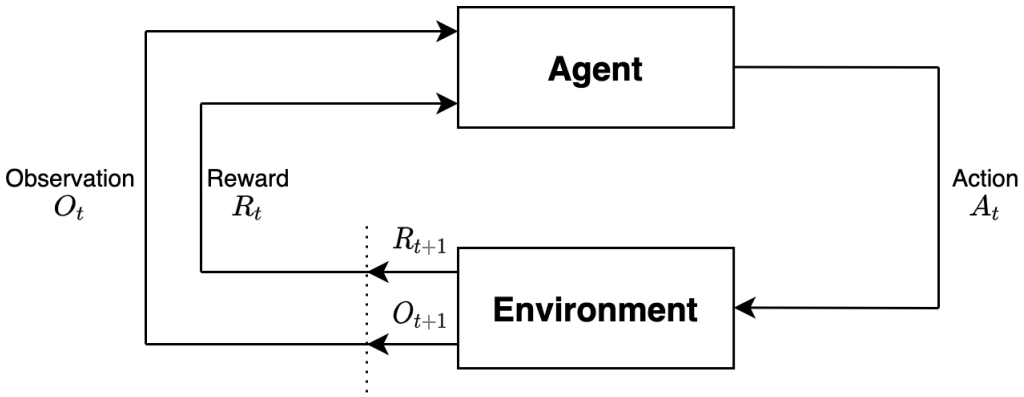

Figure 1.14: Agent-environment interface slightly modified from [12]

The Figure 1.14 represents the agent-environment interface and is constituted with the following elements :

- An environment, e.g., a maze, a chess board, etc., defined an agent's surroundings and is composed of (physical, biological, etc.) rules.
- An agent, e.g., traffic controller, autopilot, etc., is typically defined as a software component that acts in its environment.
- An environment state S_t^e is an information used within the environment and determines the next observation/reward from the environment's point of view. It is usually not visible to the agent [30].
- An agent state S_t^a is an information used within the agent and determines the next action from the agent's point of view [30].
- A reward R_t is a scalar feedback signal, indicating how well the agent is doing at the current time step.
- An action A_t, e.g., move up, move down, etc., allows interactions with the environment and belongs to the agent.

Fully and partial observable environment

The propriety of full observability allows an agent to observe the environment state directly. Formally, the problem becomes a Markov Decision Problem (MDP) [30] :

$$O_t = S_t^a = S_t^e \tag{1.9}$$

If the environment is not fully observable, the agent observes the environment indirectly. Formally, the problem becomes a Partially Observable Markov Decision Process ($POMDP$) and the agent must construct its own state representation S_t^a [30] :

$$O_t \neq S_t^a \neq S_t^e \tag{1.10}$$

Types of RL agent

There are different types of agents [30] :

- Policy-based : if the agent contains and works with a policy, a deterministic or stochastic function mapping a state to action. Therefore it allows the agent to know what action should be taken for each state. The goal of the agent is to find the optimal policy. More intuitively, one can think of the policy as the agent's strategy or behavior.

$$\begin{cases} a = \pi(s) \\ \pi(a|s) = \mathcal{P}\left[A_t = a | S_t = s\right] \end{cases} \tag{1.11}$$

- Value-based : if the agent contains and works with a value function, which is the expected discounted cumulative rewards given the current state. The goal of the agent is to maximize this quantity and is expressed as

$$v_\pi(s) = \mathbb{E}\left[G_t | S_t = s\right] \tag{1.12}$$

  where G_t is defined as the return :

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{1.13}$$

  Intuitively, equation 1.12 indicates to the agent how good it is to start from current state s and follow policy π. The discount factor (between 0 and 1) represents the importance of immediate reward. A γ factor closes to 0 means immediate reward is privileged over future reward and γ closes to 1 means future reward is almost as important as immediate reward.

- Actor-critic : if it contains and works with a value function and policy.

Besides policy and value functions, an agent containing a Model is called a Model-Based agent, otherwise it is a Model-Free agent. The Model is composed of a transition model

allowing the agent to predict the next state (dynamics) and a reward model enabling the agent to predict the next immediate reward :

$$\begin{cases} \mathcal{R} = \mathbb{E}\left[R_{t+1}|S_t = s, A_t = a\right] \\ \mathcal{P}^a_{ss'}\left[S_{t+1} = s'|S_t = s, A_t = a\right] \end{cases} \tag{1.14}$$

where s' is the next state and s is the current state.

1.5.3 Environment mathematical framework

Under the assumption of a fully observable environment, a *MDP* describes this environment. The goal of this section is to detail this formalism step by step with a set of definitions :

Markov property

A state S_t is Markov if and only if [30] :

$$\mathcal{P}\left[S_{t+1}|S_t\right] = \mathcal{P}\left[S_{t+1}|S_1, S_2, ..., S_t\right] \tag{1.15}$$

Intuitively, equation 1.15 means that it does not matter how an agent gets in the current state, the future depends upon where the agent currently is (memory-less property).

Markov process

A Markov process is a memory-less stochastic process, i.e., a sequence of random states $S_1, S_2, ...$ with the Markov property, as defined before.
Formally, a Markov process is a tuple $< \mathcal{S}, \mathcal{P} >$ with

- $\mathcal{S}$, a (finite) set of states.
- $\mathcal{P}$, a transition probability matrix (i.e dynamics), $\mathcal{P}\left[S_{t+1} = s'|S_t = s\right]$

Markov reward process

A Markov reward process is a Markov process with "judgment" values [30].

Formally, a Markov reward process is a tuple $< \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma >$ with

- $\mathcal{S}$, a (finite) set of states.
- $\mathcal{P}$, a transition probability matrix (i.e dynamics), $\mathcal{P}\left[S_{t+1} = s' | S_t = s\right]$
- $\mathcal{R}$, a reward function, $\mathcal{R} = \mathbb{E}\left[R_{t+1} | S_t = s\right]$
- $\gamma \in [0, 1]$, a discount factor.

Markov decision process

A Markov decision process is a Markov reward process with decisions [30].

Formally, a Markov decision process is a tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ with

- $\mathcal{S}$, a (finite) set of states.
- $\mathcal{A}$, a (finite) set of actions.
- $\mathcal{P}$, a transition probability matrix (i.e dynamics), $\mathcal{P}^a_{ss'}\left[S_{t+1} = s' | S_t = s, A_t = a\right]$
- $\mathcal{R}^a_s$, a reward function, $\mathcal{R}^a_s = \mathbb{E}\left[R_{t+1} | S_t = s, A_t = a\right]$
- $\gamma \in [0, 1]$, a discount factor.

Solving MDP

Another quantity of interest is defined as the Q function, also called action-value function, which is the expected return starting from state s, taking action a, and then following policy π [30] :

$$q_\pi(s,a) = \mathbb{E}\left[G_t | S_t = s, A_t = a\right] \tag{1.16}$$

Solving an MDP, i.e., learning process of the *RL* agent, is to find the maximum Q function over all policies :

$$q_*(s,a) = \max_\pi \; q_\pi(s,a) \tag{1.17}$$

The following section describes two algorithms commonly implemented by an agent to solve a *MDP*.

1.5.4 Agent mathematical framework

As seen in the previous section, the goal of an agent is to solve a Markov decision process, i.e., find the optimal policy π. It is optimal if for any policy π' [30] :

$$\pi \geq \pi' \text{ if and only if } v_\pi \geq v_{\pi'} \text{ for all } s \in \mathcal{S} \tag{1.18}$$

The algorithms used in this work, Deep Q-Networks (*DQN*) and Deep Deterministic Policy Gradient (*DDPG*), are deep reinforcement learning algorithms, i.e., based on deep neural networks and reinforcement learning techniques. The main advantage of *DDPG* over *DQN* is the ability to learn from high observations dimension, such as images [6].

Algorithm 1 : DQN

Q-Learning is a reinforcement learning technique used for learning the optimal policy by learning the optimal Q function $q_*(s, a)$. To learn this function, at every iteration, the agent updates the Q-values for each state-action pair using the Bellman optimality equation [30] until the Q function converges to the optimal Q function. This iterative approach is called value-iteration.

$$q_*(s, a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(s', a')\right] \tag{1.19}$$

The algorithm uses a Q-table, which is a table storing values called Q-values for each state-action pair. The dimension of this table is given by the size of the state space and action space. Initially, the agent knows nothing about the environment, and therefore the Q-values for each pair are set to zeros. To answer the question of how the agent should take actions based on this table, let's first introduce the exploration/exploitation dilemma. Exploration is the act of exploring the environment in order to find information and exploitation is the act of exploiting the information already known by the agent in order to maximize the return. The agent needs to balance those two objectives since only exploitation can lead the agent to miss better strategy (higher return) and with only exploration, the agent does not use known information to maximize the return. A common solution to address this problem is called $\epsilon - greedy$. It is an algorithm in which an exploration rate ϵ is set and defined as the probability that the agent will explore rather than exploit. Initially it is set to 1, meaning that agent will only explore its environment. As the agent learns more about the environment, at the start of each episode, the rate decreases so that the likelihood of exploration becomes less probable. In order to choose between exploration or exploitation at each time step, a random number $r \in [0, 1]$ is generated and if $r > \epsilon$, the agent will choose its next action to be an exploitation, i.e., choose the highest Q-value for its current state otherwise it will be an exploration one, i.e?, choose an action randomly. Therefore, the first action is an exploration one, since

at the beginning of the episode, the exploration rate is set to 1. At each time step, the agent updates the table by using a learning rate $\alpha \in [0, 1]$, which is a factor indicating how much previous information will be retained from the previous Q-value :

$$q^{new}(s, a) = (1 - \alpha)q^{old}(s, a) + \alpha \left(\gamma \max_{a'} q_*(s', a')\right) \tag{1.20}$$

It is a weighted sum of the old value and learned value. The learning happens with a finite number of time steps and episodes. Eventually, it converges to the optimal policy.

The problem with Q-Learning is that the size of the Q-table limits the performance. In a complex environment, the number of actions and states is high, meaning that the update time in the Q-table becomes very high. Instead of using value iteration to compute Q-values and find the optimal Q function directly, a function approximator can estimate the optimal Q function. In 2013, Mnih et al. [31] introduced *DQN* which is the idea of using deep neural networks with Q-Learning. Therefore *DQN* is a deep reinforcement learning algorithm. The use of neural network allows to estimate the Q function. The state is given at the input of the network and produces at the output of the network, Q-values for all possible actions for this state. At each time step for every episode during training, the agent experiences the tuple $e_t = (s_t, a_t, r_{t+1}, s_{t+1})$. The last N experiences are stored in a dataset called the replay memory. Random samples or batch from this memory is used to train the network, also called the policy network. This allows to break the correlation between consecutive samples (efficient learning). The steps of the algorithm are the following :

1. Initialize replay memory.
2. Initialize the policy network with random weights
3. For each episode, initialize a starting state and at each time step of the episode :
   (a) the agent chooses and executes an action (according to exploration/exploitation), observes reward and next state and store the experience e_t in replay memory.

(b) the agent samples a random batch from the replay memory

(c) the agent preprocess the batch

(d) the agent sends the batch at the input of the policy network

(e) the policy network computes the loss by comparing output Q-values and target Q-values (given by Equation 1.21)

(f) the policy network updates the weights by using classical gradient descent and backpropagation to minimize the loss

$$Loss = q_*(s,a) - q(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(s',a')\right] - \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right] \tag{1.21}$$

To compute the loss, the term $\max_{a'} q_*(s',a')$ needs to be computed. It is done by feeding the network the next state (from the experience tuple) and taking the maximum output of the network. The training can become unstable since the same network is used to compute the target and the prediction. In practice, the algorithm uses two networks. A second network (target network) is used to calculate the target Q-values, while the prediction Q-values are computed by the first network (policy network).

Algorithm 2 : DDPG

Some environments need continuous action space and are not suitable with *DQN* since it requires a discrete space of actions. A solution proposed by Lillicrap et al. [13] is to use *DDPG*, which is a type of actor-critic algorithm with four networks :

- The actor network is a policy network with parameters θ^{μ} whose input is the current state and output is a deterministic continuous action instead of a probability distribution over action given the state. This is the reason for 'deterministic' in the naming of the algorithm. To solve the exploration/exploitation dilemma, Gaussian noise is added to the output action [13]. Mathematically the network is defined as :

$$\mu(s|\theta^{\mu}) \tag{1.22}$$

- Target policy network with parameters $\theta^{\mu'}$ is a time-delayed copy of the original actor network, used for computing targets allowing better stability of the learning process. Mathematically the network is defined as :

$$\mu'(s|\theta^{\mu'}) \tag{1.23}$$

- The critic network is a Q network with parameters θ^{Q} whose input is the current state and action given by the actor network. The output is the Q-value of the state-action pair. Mathematically the network is defined as :

$$Q(s, a|\theta^{Q}) \tag{1.24}$$

- Target Q network with parameters $\theta^{Q'}$ which is a time-delayed copy of the original critic network, used for computing targets allowing better stability of the learning

process. Mathematically the network is defined as :

$$Q'(s, a|\theta^{Q'}) \tag{1.25}$$

The algorithm is given by the Figure 1.15 :

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Figure 1.15: DDPG algorithm as described in the original paper [13]

where index i is indexing the batch of size N, y_i are the targets calculated from the target networks, r_i are the immediate rewards, and τ is a parameter close to one. Figure 1.16 summarizes the relation between the actor and critic networks :

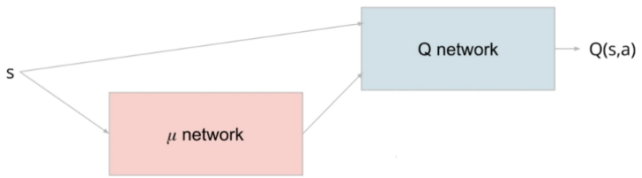

Figure 1.16: Actor-Critic [14]

Chapter 2

Materials and methods

The goal of this chapter is to use relevant literature detailed in *Chapter 1* to implement a computer simulation including :

- An implementation of an environment representing cells proliferation (cancer and healthy) on a 3D grid.
- An implementation of actions, such as irradiation of the 3D grid with a modulated beam of protons, (x,y) positioning of the beam, and variable rest time after the irradiation.
- a reinforcement learning agent whose goal will be to find a treatment plan based on rewards functions.

The simulation is highly based on the work of Moreau [6]. The main difference is the use of protons instead of radiotherapy. Moreover, the simulation is in 3D instead of 2D and considers the density for each voxel. The results will be then discussed in *Chapter 3*.

2.1 Basis of the simulation

The basis of the simulation is a grid in 3D with $int\left(\frac{width*length*depth}{voxelSize}\right)$ voxels (Figure 2.1). Each cubic voxel of volume $voxelSize^3$ in mm^3 contains :

- a quantity of glucose [mg], which is a nutrient for cells.
- a quantity of oxygen [ml], which is a nutrient for cells.
- a density $[g/cm^3]$, allowing a grid with different tissues (bones, water, etc.).
- a list of healthy and cancer cells.

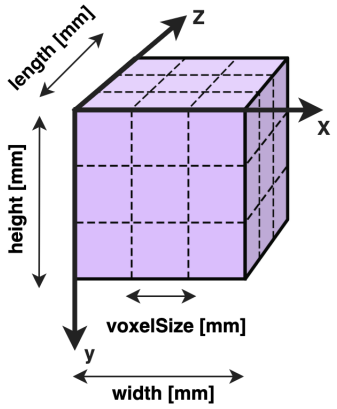

Figure 2.1: 3D Grid of the simulation

Initially, the simulation starts with a quantity of glucose and oxygen, one healthy cell per voxel, several sources (nutrients), and one cancer cell in the middle of the volume. The sources represent endothelial cells, which form blood vessels in the human body [6]. The positions of endothelial cells are entirely random (uniform distribution). The small quantity of initial glucose and oxygen allows early spread of healthy tissues throughout the grid and the formation of a tumor mass in the center of the volume. The parameters used are summarized in Table 2.1. Note that values described in tables are the physical ones but in the simulation they are scaled values [20].

Parameter	Description	Value	Unit
width	width of grid	$30, 100$	[mm]
height	height of grid	$30, 100$	[mm]
length	length of grid	$50, 400$	[mm]
voxelSize	size of voxel, i.e resolution	1	[mm]
voxelsNumber	number of voxels	$int\left(\frac{width*height*length}{voxelSize}\right)$	[/]
numSources	number of endothelial cells (sources)	$0.02 * voxelsNumber$	[/]
density	matrix representing density of the tissue per voxel	$density[i, j, k] = 1$ (water everywhere)	$[g/cm^3]$

Table 2.1: Grid parameters used for the simulation

2.2 Cells proliferation

2.2.1 Cell object

Each voxel contains a list of healthy and cancer cells. Those cells are objects defined by a set of properties. They are summarized in Table 2.2 and based on the work of Moreau [6]. Two types of cells are defined in the simulation :

- HealthyCell : representing healthy tissue in the model
- CancerCell : representing tumoral tissue in the model

They mainly differ in consumption of glucose, with cancer ones consuming more than healthy ones. Moreover, cancer cells can not enter in quiescent mode. The cell cycle used in the simulation is the one described in *Chapter 1*. Physically, one tick is equivalent to one hour of the cycle in the code and corresponds to one iteration. Cells in the mitosis stage stay in their voxels and new cells formed are placed to one of their neighbors with smallest density. Regarding cancer cells created via mitosis, they are placed randomly (uniform) in one of their neighbors, allowing the tumor to not spread too quickly. In quiescent mode, the cells consume less than in others modes (25% less).

2.2.2 Parameters

The parameters used are summarized in Table 2.2

Property	**Definition**	**Value**	Unit
stage	stage of the cell in the cycle	0 : Gap1 1 : Synthesis 2 : Gap 2 3 : Mitosis 4 : Quiescent	[/]
alive	killed or not by radiation	[True, False]	[/]
age	age of the cell, determines if it should go to next stage	[0,11]	[h]
efficiency	represents small variations in consumption between cells	~ N(1,1/3)	[/]
average_oxygen_ consumption	average oxygen consumption for a cell (cancer or healthy)	2.16×10^{-9}	[ml/h]
average_glucose_ consumption	average glucose consumption for a healthy cell	3.6×10^{-9}	[mg/h]
average_c_glucose_ consumption	average glucose consumption for a cancer cell	5.4×10^{-9}	[mg/h]
oxygen_ consumption	oxygen consumption for a cancer or healthy cell	average_oxygen_ consumption * efficiency	[ml/h]
glucose_ consumption	glucose consumption for a healthy cell	average_glucose_ consumption * efficiency	[mg/h]
glucose_c_ consumption	glucose consumption for a cancer cell	average_glucose_c_ consumption * efficiency	[mg/h]
critical_oxygen_ level	min amount of oxygen to keep cell alive	3.88×10^{-8}	[ml]
critical_glucose_ level	min amount of glucose to keep cell alive	6.48×10^{-8}	[mg]
critical_neighbors	maximum amount of neighbors	9	[cells]
quiescent_oxygen_ level	if oxygen in a voxel is less than this value, healthy cells enters quiescent	1.04×10^{-7}	[ml]
quiescent_glucose_ level	if glucose in a voxel is less than this value, healthy cells enters quiescent	1.73×10^{-7}	[mg]
repair	model repair time in hours	9	[h]

Table 2.2: Cell parameters and threshold values used for the simulation

2.3 Nutrients

2.3.1 Replenishment and diffusion process

At the start of the simulation, a number of sources called $numSources$ are placed randomly (uniform) across the grid. Each hour, representing one tick :

- each voxel containing cells diffuses a certain amount of their nutrients to its neighbors across all directions. It is done through a parameter called the $diffusionRate$. At the border of the grid, they are deleted from the simulation, allowing to alleviate a 'border effect' [6].
- the sources are replenished with $replenishmentGlucose$ and $replenishmentOxygen$ values because of the loss of nutrients at the borders.

2.3.2 Angiogenesis

Once a tumor reaches a stage where the environment can not provide sufficient nutrients, the tissue around the tumor can develop blood vessels to supply nutrients to the tumor. This phenomenon is called angiogenesis. The sources have a certain probability of moving each hour to model this phenomenon, given by $pMove = 1/24$. This allows the sources to move on average once a day. The next step is to determine if they move in the direction of the tumor with probability $pMoveToTumor$ or to a random (uniform) neighbor. As the number of cancer cells increases, the probability of moving to the tumor also increases.

2.3.3 Parameters

The parameters used are summarized in Table 2.3

Parameter	Description	Value	Unit
initialGlucose	initial quantity of glucose per voxel	10^{-6}	[mg]
initialOxygen	initial quantity of oxygen per voxel	10^{-5}	[ml]
replenishmentGlucose	replenishment of glucose sources	1.3×10^{-6}	[mg]
replenishmentOxygen	replenishment of oxygen sources	4.86×10^{-5}	[ml]
diffusionRate	diffusion rate of voxels	0.3	[/]
pMove	probability of sources moving once a day	1/24	[/]
pMoveToTumor	probability of sources moving to tumor	$\frac{\text{number of cancer cells}}{50000}$	[/]

Table 2.3: Nutrients parameters used for the simulation

2.4 Beam implementation

Particle traveling can interact with matter, and the effect can be various. Let's assume a Markov process, meaning that the next event is determined by the current state of the particle and does not depend on the entire history. The particle has a direction, energy, and position. Those three elements defined the state of a particle i for each time step s [mm] :

$$state_i = (energy, \overrightarrow{direction}, \overrightarrow{position}) \tag{2.1}$$

- energy is a scalar in [KeV]
- direction is a 3D unit vector
- position is a 3D vector

Note that this part is based on Fippel et al. [32] and [33].

2.4.1 Particle simulation - Electromagnetic interactions

The first step is to simulate electromagnetic interactions, i.e., ionization. Protons can undergo 10^6 EM interactions per cm. This requires too much calculation time to simulate all the interactions. The idea is too condensed interactions in one single step of defined length, i.e., the simulation step.

Continuous slowing down approximation (CSDA)

The particle starts with initial energy, position, and direction. As seen in *section 1.4.2*, protons interact with matter, and inelastic Coulomb interaction with atomic electrons happens, leading to energy loss. Therefore, for each step s $[mm]$, the particle loses energy ΔE $[KeV]$ and this energy loss depends on the mass density ρ $[g/cm^3]$, the mass stopping power SP(E) $[MeVcm^2/g]$, and the step size s $[cm]$ (Figure 2.2).

$$\Delta E = \rho * SP(E) * s \tag{2.2}$$

This process continues until there is no energy left in the particle and the simulation for this particle ends.

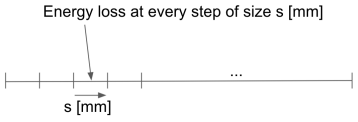

Figure 2.2: Energy loss at every step s [mm]

Sampling initial particle position

Practically, in a clinic for instance, one uses a modulated beam of protons instead of a unique particle. The profile of the beam can be modeled as a Gaussian [33]. For instance, a beam targeted on XY plane yields in

$$\begin{cases} p_x \sim \mathcal{N}(p_{startX}, \sigma_x^2) \\ p_y \sim \mathcal{N}(p_{startY}, \sigma_y^2) \\ p_z = p_{startZ} \end{cases} \tag{2.3}$$

where the voxel $(p_{startX}, p_{startY}, p_{startZ})$ represents the initial mean position of the particle. Particles will be spread around this mean according to the variances defined above. The spot size of the beam is therefore defined as the variances in x and y. Larger the spot size, the larger the variances.

Sampling initial particle energy

Practically, a beam is not a monochromatic but instead composed of multiples particles with various energies. This can also be modeled by a Gaussian distribution :

$$E_{start} \sim \mathcal{N}(E_{mean}, \sigma_E^2) \tag{2.4}$$

where $E_{start}.$ is the starting energy of the particle and E_{mean} the mean energy of the beam.

Energy straggling

In the realm of quantum mechanics, physics is described with probabilistic models. In practice, there is a small statistical perturbation that is applied over the stopping power. This variation of energy loss due to quantum mechanics is called energy straggling. This perturbation can be modeled by Bohr's theory (Gaussian distribution) for large steps.

The parameters influencing this perturbation are the rest mass of electrons m_e and protons m_p $[kg\frac{m^2}{s^2}]$, the radius of electrons r_e [cm], the density of electrons n_{el}, the atomic number Z, the ratio of proton velocity to the velocity of light β and the maximum energy transferred to electrons T_e^{max} [KeV], the total energy of the proton E_p [KeV]:

$$\begin{cases} \Delta E \sim \mathcal{N}(\rho * SP(E) * s, \Omega^2) \\ \Omega^2 = 2\pi r_e^2 * m_e c^2 * n_{el} * \frac{z^2}{\beta^2} * s * T_e^{max} * (1 - \frac{\beta^2}{2}) \\ T_e^{max} = \frac{2m_e\beta^2\gamma^2}{1+2\gamma m_e/m_p+(m_e/m_p)^2} \\ \gamma = \frac{E_p}{m_p} = \frac{T_p+m_p}{m_p} \\ \beta = \sqrt{1-\frac{1}{\gamma^2}} \end{cases} \tag{2.5}$$

Therefore, the energy loss deposited in a voxel is a random variable with Gaussian distribution.

Multiple Coulomb Scattering

As seen in *section 1.4.2*, protons interact with matter, and elastic Coulomb scattering with atomic nucleus happens. The particle's motion can be correctly simulated by modeling angular deflection caused by this interaction. This can be modeled by changing the direction of the particle at the end of the step, and this is changed according to Multiple Coulomb Scattering theory [32] [33].

- The azimuthal angle $\phi \sim U(0, 2\pi)$
- The polar angle $\theta \sim \mathcal{N}(0, (\frac{E_s}{\beta pc} z)^2 \frac{s}{X_0(\rho)})$

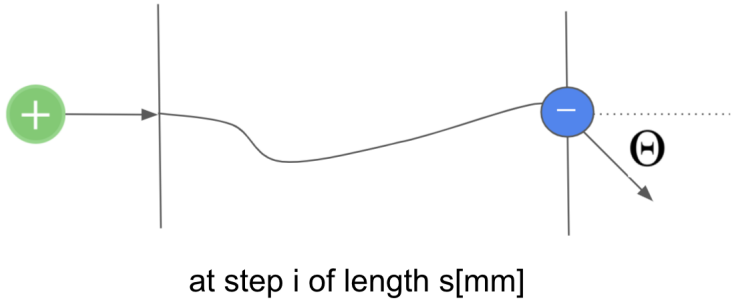

Figure 2.3: Sampling of deflections angles

where E_s is a constant factor computed by fitting differential dose distributions in water with nuclear reactions switched off [32], $pc = \sqrt{\sqrt{T_p + m_p} - \sqrt{m_p}}$ is the product of the proton's momentum and the speed of light and $X_0(\rho)$ is the radiation length [mm] of a material with density ρ

Parameters

The parameters and constants used are summarized in Table 2.4

Parameter	Definition	Value	Unit
spWater	table of stopping powers from PSTAR program	table of values in water	$[MeVcm^2/g]$
beamMeanEnergy	mean energy of the proton	depending on the agent	[MeV]
beamPosition	center position of the impact on XY plane	depending on the agent	[mm]
sigmaX, sigmaY	representing diameter of the beam	2, 2	[mm]
sigmaE	variance of initial particle energy	1	[MeV]
particlesNumber	number of protons in a beam	10^9	[/]
s	step length	1	[mm]
m_p	rest mass of proton	938.272	[MeV]
m_e	rest mass of electron	0.511	[MeV]
r_e	classical radius of electron	2.81×10^{-12}	[mm]
z	atomic number of proton	1	[/]
x_w	radiation length in water	360.86	[mm]
E_s	Multiple Coulomb Scattering constant factor	12.4	[/]

Table 2.4: Parameters and constant used for the simulation

Parameters highlighted in green in Table 2.4 are the ones that the agent will choose. One can also define other actions, but this will lead to a simulation that is too complex.

2.4.2 Particle simulation - Nuclear interactions

In contrast with *EM* interactions, the occurrences of nuclear interactions are pretty small. Again, the model is described in Fippel et Al [32]. The sum of cross sections gives the probability of nuclear interactions. The formulas depend on the proton's kinetic energy and were generated by fitting nuclear cross section databases. The nuclear interactions are not modeled for this work causing slow simulation due to secondary particles generations and not really suitable for deep reinforcement learning purpose, which needs a lot of training samples.

2.5 Integration of deep reinforcement learning

2.5.1 DeeR framework

DeeR is a python library mainly developed by François-Lavet [34] allowing the use of deep reinforcement learning. The framework is based on Keras and TensorFlow.

- A NeuralAgent can be defined by an environment, discount factor, neural network (e.g., Q network), replay memory, batch size, epochs, etc.
- Controllers can be attached to the NeuralAgent to set the learning rate, the exploration rate, etc.
- The environment must inherit Environment interface and implements functions observe(), act(), nActions(), inputDimensions(), inTerminalState(), and reset(). The names are self explanatory.
- Once the NeuralAgent is trained, it can be saved (dump the network) locally and used to get a treatment plan.

2.5.2 Training of the agent

Observations

To act in its environment, the agent first receives the states of the environment. This is done by feeding images in three dimensions to the neural network of the agent, defined as :

$$D(x,y,z) = \begin{cases} -numberCancerCells & \text{if there is a cancer cell} \\ numberHealthyCells & \text{if there are only healthy cells} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

This kind of observation is also investigated in [6].

Actions

The agent will have to choose actions based on the observations discussed in the previous section. The type of actions are the following :

- change (x,y) position around the center of target XY plane.
- change the mean energy of the beam.
- change the rest time between irradiation.

Two types of agents are investigated, an agent implementing *DQN* and another one implementing *DDPG* :

1. The action space of the *DQN* agent is discrete and uni-dimensional. As seen in the previous section, the *DQN* algorithm needs to be small. The action space is summarized in Table 2.5

Parameter	**Values**	**Unit**
XY plane position	(15,15)	[mm]
Beam mean energy	[50,55,60,65,70,75,80]	[MeV]
Rest time	12	[hours]

Table 2.5: DQN agent action space

2. The action space of the *DDPG* agent is continuous and multidimensional. The action space is summarized in Table 2.6

Parameter	**Values**	**Unit**
XY plane position	([14,16],[14,16])	[mm]
Beam mean energy	[50, 80]	[MeV]
Rest time	[12, 24]	[hours]

Table 2.6: DDPG agent action space

Terminal state

The sequence of state-action-reward from an initial random state to a terminal state is called an episode. An episode can end in three manners :

- A success, every cancer cells are killed ($cancerCells = 0$)
- A failure, the number of remaining healthy cells is low ($healthyCells < 10$), meaning that the tumor has invaded the entire grid.
- A time out, allowing the agent to get the treatment that is not too long ($ticks < 500$). Treatment that is too long can be harmful to the patient due to bystander effects not included in the simulation. Moreover, long treatment impacts the convenience of the patient [6].

Rewards

The agent's primary goal is dual, i.e., spare the surrounding tissues of the tumor (OAR) and kill the tumor (TV). To achieve the goal, the agent needs to maximize the return. A reward function based on biological criteria is investigated in [6]. At each time step, the agent receives a reward :

$$r = \frac{(cancerCellsKilled - \lambda * healthyCellsKilled)}{k_1} \tag{2.7}$$

where

$$\begin{cases} cancerCellsKilled = \#cancerCells|_{\text{before irradiation}} - \#cancerCells|_{\text{after irradiation \& rest}} \\ healthyCellsKilled = \#healthyCells|_{\text{before irradiation}} - \#healthyCells|_{\text{after irradiation \& rest}} \\ k_1 \text{ is a normalization factor to get reward values between -1 and 1} \\ \lambda \text{ is a regularization factor to get less aggressive treatment [6]} \end{cases} \tag{2.8}$$

The problem with this reward function is that the agent tends to let the tumor grows and then send high doses to get high reward values. To manage this problem, a special reward is given in a terminal state [6]. In the case of a time out or failure state, a reward of -1 is sent. In the case of success, the reward received is given by [6] :

$$\frac{Init_{hcells} - End_{hcells}}{k_2} \tag{2.9}$$

where

$$\begin{cases} Init_{hcells} = \#heatlhyCells \text{ at the start of an episode} \\ End_{hcells} = \#healthyCells \text{ at the end of an episode} \\ k_2 \text{ is a normalization factor to get reward values between 0 and 1} \end{cases} \tag{2.10}$$

The factors λ, k_1 and k_2 are different from the one used in [6] since the simulation is not exactly the same (geometry, use of protons, etc.). They are also determined experimentally and are given in Table 2.7

Parameter	**Values**
λ	3
k_1	10000
k_2	5000

Table 2.7: Experimental values for the parameters of rewards functions

DQN agent hyper-parameters

The parameters used for training the DQN agent are summarized in Table 2.8

Parameter	Definition	Values
epoch	number of complete passes through the dataset an epoch is made up of one or many episodes	30
epoch_length	is the maximum number of steps during an episode	100
batch_size	determines the number of iteration per epoch	32
df	discount factor	0.95
initial_e	initial exploration rate	0.8
min_e	minimum exploration rate	0.2
initial_learning_rate	initial learning rate	0.0001
learning_rate_decay	learning rate decay	25%

Table 2.8: DQN parameters based on [6]

DDPG agent hyper-parameters

The parameters used for training the DDPG agent are summarized in Table 2.9

Parameter	Definition	Values
epoch	number of complete passes through the dataset an epoch is made up of one or many episodes	30
epoch_length	is the maximum number of steps during an episode	100
batch_size	determines the number of iteration per epoch	32
df	discount factor	0.95
initial_std	initial standard deviation of Gaussian noise	0.5
final_std	final standard deviation of Gaussian noise	0.005
initial_e	initial exploration rate	0.8
min_e	minimum exploration rate	0.2
initial_learning_rate	initial learning rate	0.0001
learning_rate_decay	learning rate decay	25%

Table 2.9: DDPG parameters based on [6]

Chapter 3

Results and discussion

This chapter aims to display and discuss the results obtained from the implementation detailed in *chapter 2*. The simulation in 3D can not be efficiently executed on a personal PC since the computations are pretty intensive for deep reinforcement learning purposes and depend on specific packages such as DeeR. To alleviate this problem, the setup used is the following :

- The code was executed on CÉCI (Consortium des Équipements de Calcul Intensif) which is a consortium of high-performance computing (HPC) centers of UCLouvain, ULB, ULiège, UMons, and UNamur. It was executed on the Dragon2 system of the HPC centers accessed through SSH. It is a system with accelerator enabled composed of 4x Volta V100 GPUs. The code was optimized to run on GPU with CUDA V11 pre-installed on the system.

- It is necessary to install the packages from the file requirements.txt with pip, which can be found in the git repo along with the code. Moreover, if needed, the app can be pulled from the hub in a container via the following commands for Singularity (pre-installed on HPC) and Docker :

Listing 3.1: Pull with Singularity or Docker

```
singularity pull docker://amsingh05/deep-rl-pt:latest
# or with Docker :
docker pull amsingh05/deep-rl-pt:latest
```

- Let's note that on an HPC, one needs to ask the system the resources (CPU, GPU, memory, etc.) required to run the code. It is done through the software Slurm, which is a workload manager. The following script and configuration is the one used to run the code :

Listing 3.2: run.sh

```
#!/bin/bash -l

#SBATCH --job-name=aman_masterthesis
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=8
#SBATCH --ntasks-per-node=1
#SBATCH --time=20:00:00
#SBATCH --mem-per-cpu=500
#SBATCH --partition=gpu
#SBATCH --gres=gpu:1
#SBATCH --mail-type=begin        # send email when job begins
#SBATCH --mail-type=end          # send email when job ends
#SBATCH --mail-user=amanpreet.singh@student.uclouvain.be

module load cuda/11.0.2
module load TensorFlow/2.3.1-fosscuda-2019b-Python-3.7.4
module load OpenCV/4.2.0-foss-2019b-Python-3.7.4

# COMMENT/UNCOMMENT one of the following command :

# train DQN agent :
#python3 DeepRL-PT/trainAgent.py -n 'DQN'

# train DDPG agent :
#python3 DeepRL-PT/trainAgent.py -n 'DDPG'

# run DQN agent :
#python3 DeepRL-PT/runAgent.py -n 'DQN'

# run DDPG agent :
python3 DeepRL-PT/runAgent.py -n 'DDPG'

# run naive treatment :
#python3 DeepRL-PT/runAgent.py -n 'DQN' --naive 'True'
```

Also, note that it is necessary to load the correct version of CUDA and TensorFlow to use the GPU, which can be cumbersome. The code for running on HPC can be executed in the directory containing the file 'run.sh' with the following commands :

Listing 3.3: Running the code on HPC dragon2

```
pip3 install --user -r requirements.txt  #for the first execution
sbatch run.sh
```

3.1 Cells proliferation

The dimensions of the simulated phantom is 40x40x200 mm^3. From Figure 3.1 to Figure 3.6, the number of ticks simulated are 375. One can observe that the initial cancer cell in the middle of the volume spreads in surrounding tissues forming a big tumor mass. For instance at Figure 3.3, the mass is at least 4mm of depth. The tumor mass grows in volume until eventually, it invades the whole phantom. Figure 3.7 to Figure 3.9 show the development of this volume with a different perspective (3D). The color, red for cancer cells and green for healthy cells, is not constant. As time goes, some voxels become darker, representing higher cells density than brighter voxels. The cells density in the center of the phantom is very high. Moreover, some voxels become completely dark at the borders, meaning that cells in those voxels are dead. This can be explained by the fact that sources migrate to the tumor as it grows (angiogenesis process), leading to a small amount of nutrients around the tumor.

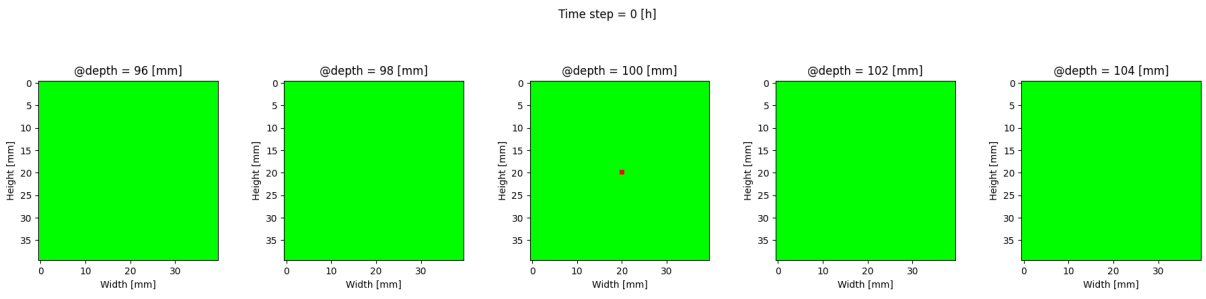

Figure 3.1: XY planes for depth = [96, 98, 100, 102, 104] mm with 100mm being the center of the tumor. Time step = 0 h

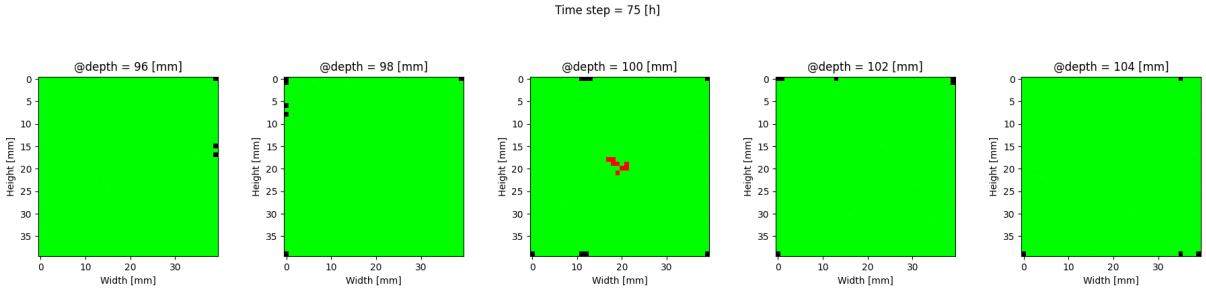

Figure 3.2: XY planes for depth = [96, 98, 100, 102, 104] mm with 100mm being the center of the tumor. Time step = 75 h

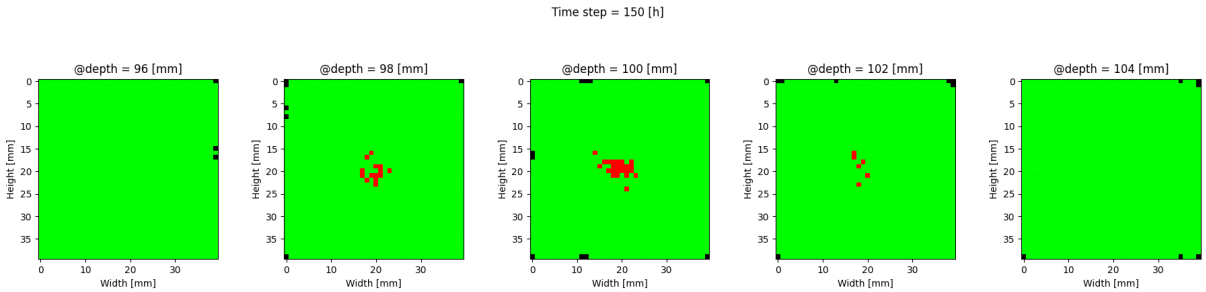

Figure 3.3: XY planes for depth = [96, 98, 100, 102, 104] mm with 100mm being the center of the tumor. Time step = 150 h

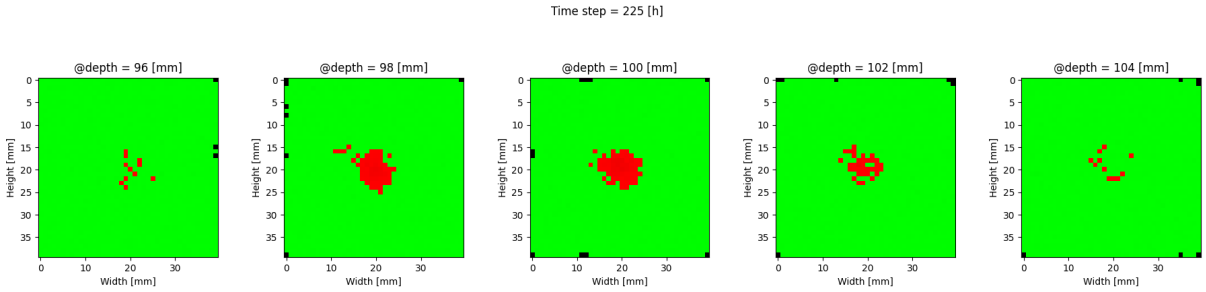

Figure 3.4: XY planes for depth = [96, 98, 100, 102, 104] mm with 100mm being the center of the tumor. Time step = 225 h

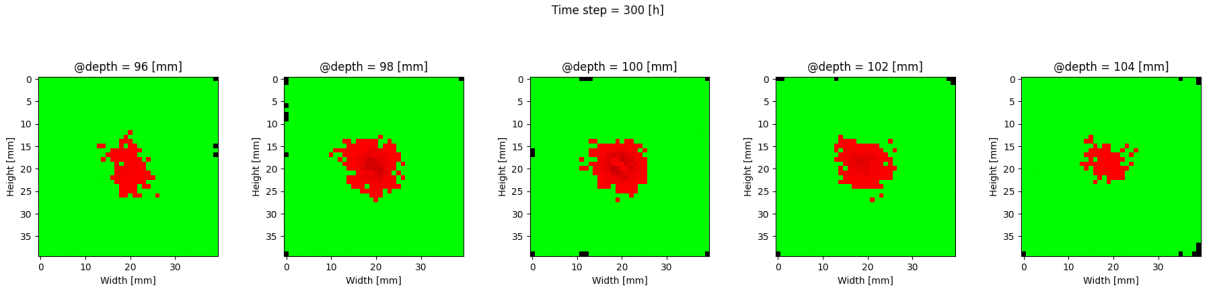

Figure 3.5: XY planes for depth = [96, 98, 100, 102, 104] mm with 100mm being the center of the tumor. Time step = 300 h

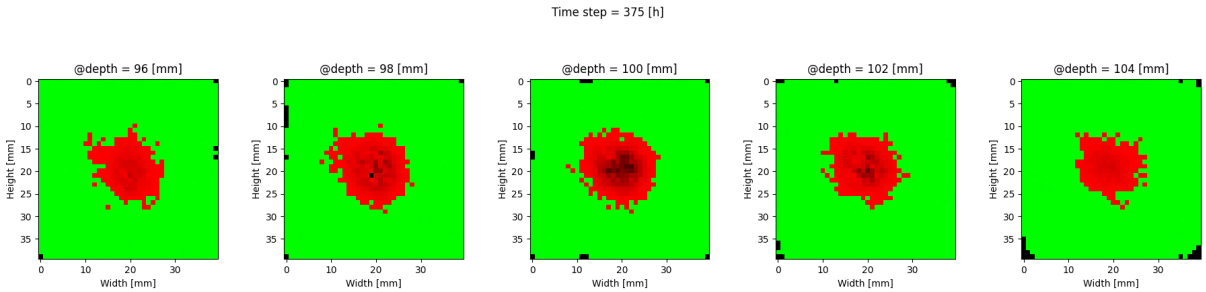

Figure 3.6: XY planes for depth = [96, 98, 100, 102, 104] mm with 100mm being the center of the tumor. Time step = 375 h

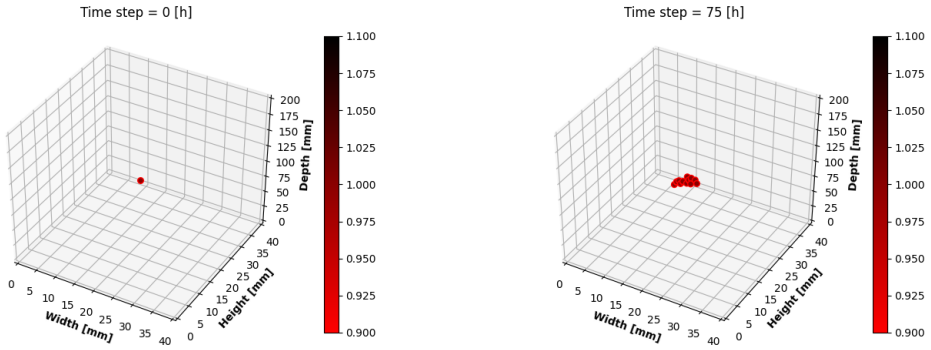

Figure 3.7: Development of tumor mass in 3D. Time step = 0 h (left) and time step = 75 h (right)

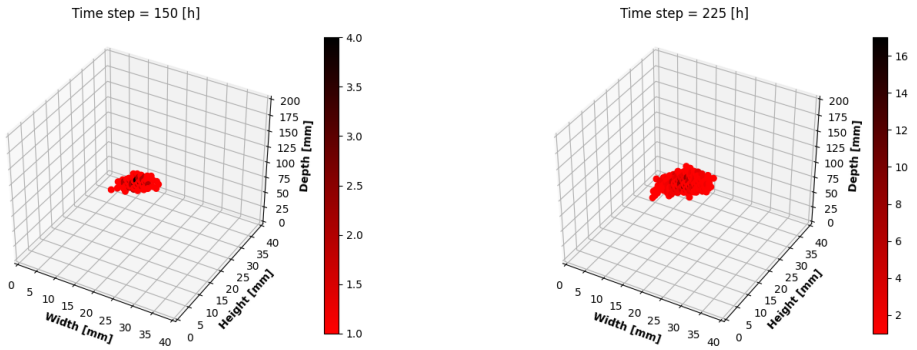

Figure 3.8: Development of tumor mass in 3D. Time step = 150 h (left) and time step = 225 h (right)

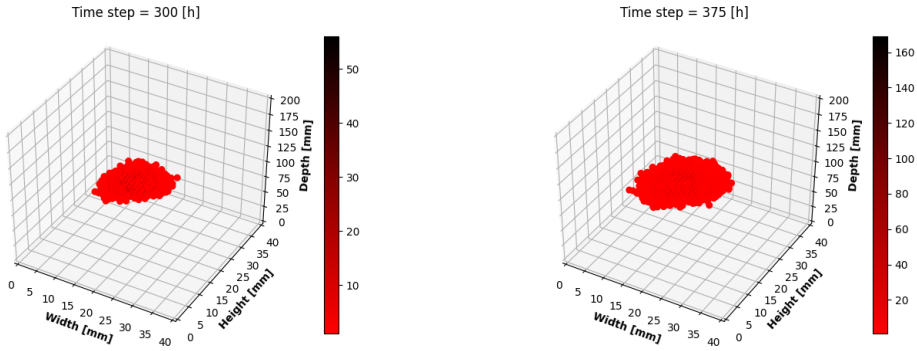

Figure 3.9: Development of tumor mass in 3D. Time step = 300 h (left) and time step = 375 h (right)

Figure 3.10 and Figure 3.11 show the evolution in time of glucose and oxygen in the center of the tumor (XY plane). The apparition of bright points on the plane, representing high density nutrients, is due to the migration of sources to the tumor's center of mass (angiogenesis), which is also a synonym of an increasing number of cancer cells.

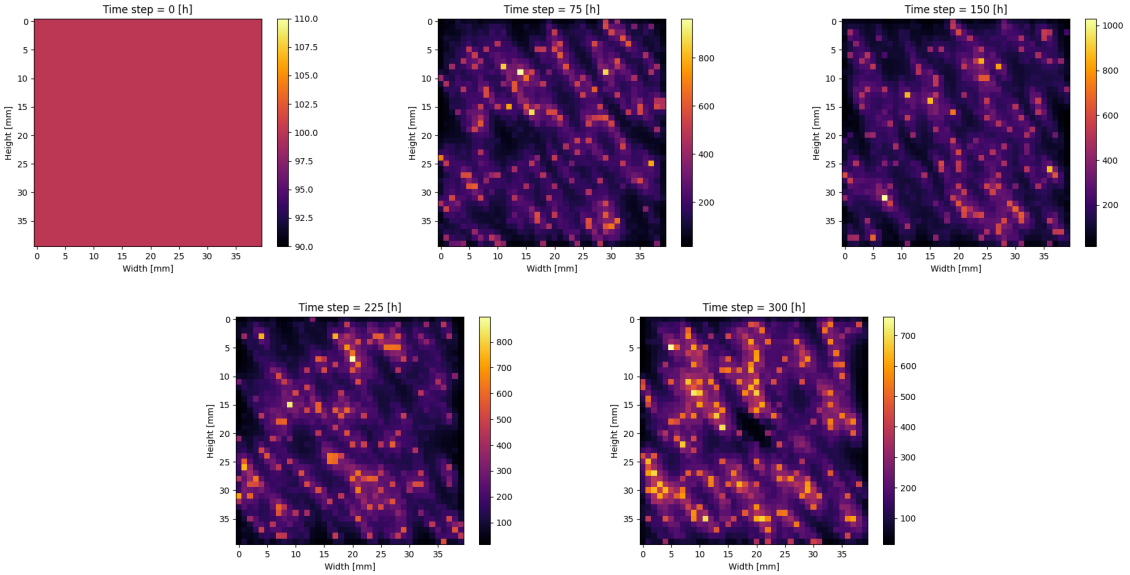

Figure 3.10: XY planes of oxygen at the center of tumor for various time steps

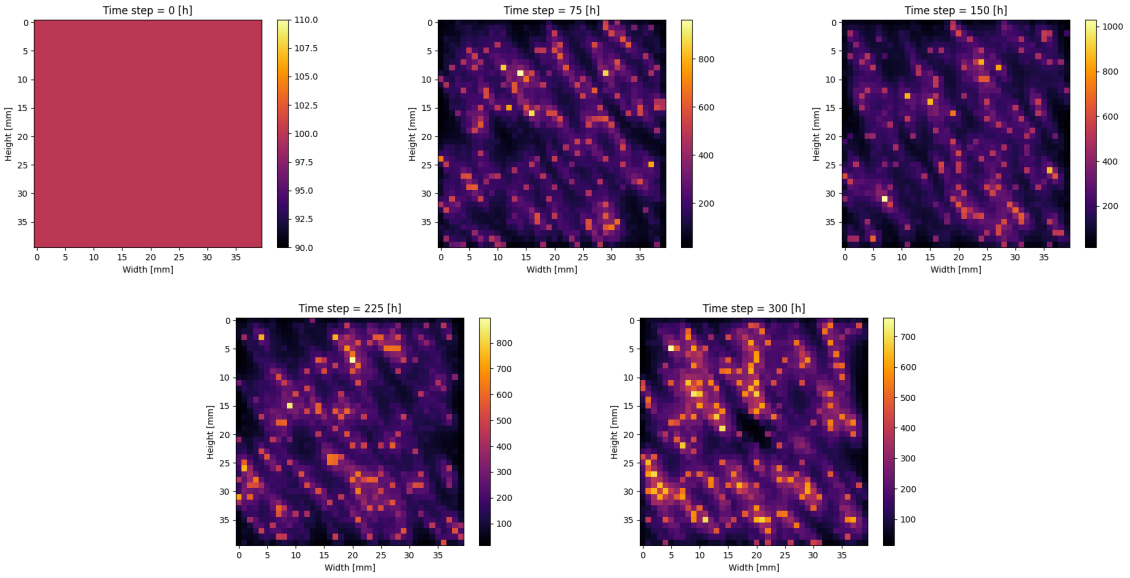

Figure 3.11: XY planes of glucose at the center of tumor for various time steps

3.2 Beam implementation

The simulated phantom is water with dimensions 100x100x400 mm^3 and a density of $1g/cm^3$ for each voxel (water). One can observe the formation of a Bragg peak in Figure 3.12 and 3.14 due to the transport of protons. As seen in section 1.4.2, the speed of proton is reducing due to interactions with matter, and as the speed reduces, the energy loss increases forming the Bragg peak. Moreover, the beam is not a straight beam because of MCS effect, i.e., the protons of the beam are deflected.

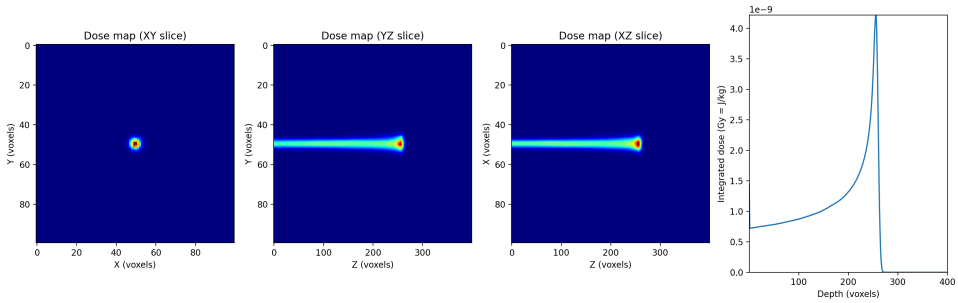

Figure 3.12: Sagittal, frontal and transversal dose planes per proton targeted in the center of XY plane with mean energy 200MeV, spot size $\sigma_x = \sigma_y = 2$, variance of energy $\sigma_E = 1$

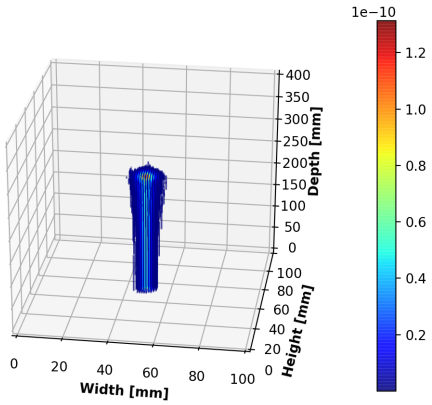

Figure 3.13: 3D view of the beam in Figure 3.12

Figure 3.14 displays the same experience as in Figure 3.12 but with mean energy of 180MeV. The only difference is the location of the Bragg p peak, which is less deep because

of reduced mean energy.

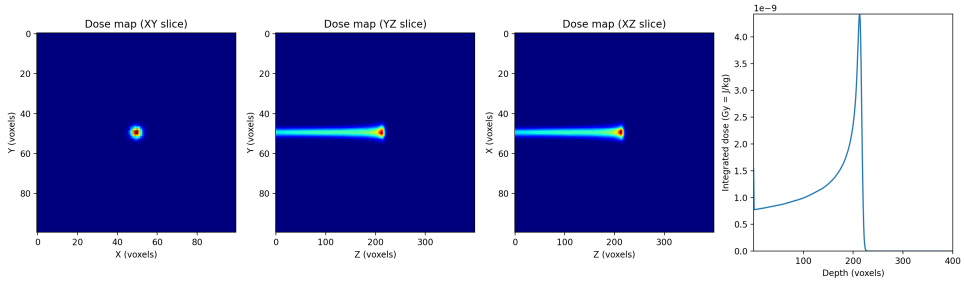

Figure 3.14: Sagittal, frontal and transversal dose planes per proton targeted in the center of XY plane with mean energy 180MeV, spot size $\sigma_x = \sigma_y = 2$, variance of energy $\sigma_E = 1$

3.3 Treatment plan

3.3.1 Naive treatment

Figure 3.15 displays an episode :

- First, the cells proliferate, producing a starting state.
- The treatment starts after time t = 180 hours (vertical bar in orange)
- In this example, actions are taken naively with fixed values (meanBeamEnergy = 60MeV, rest = 24 hours, target (x,y) position at the center) to kill the cancer cells.

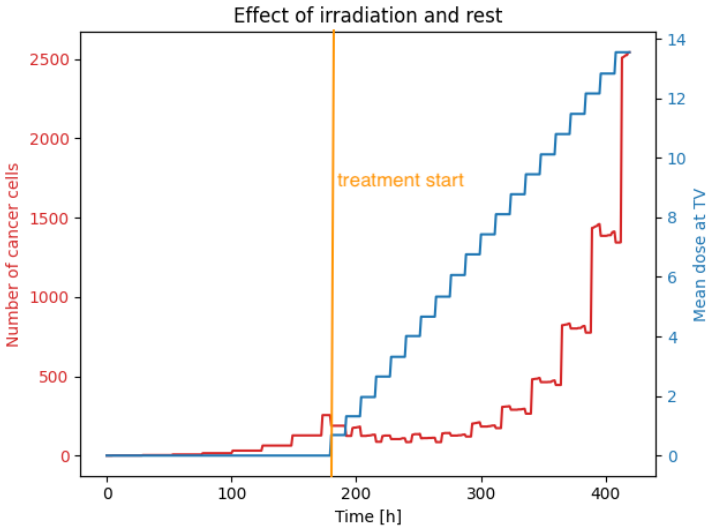

Figure 3.15: Failed treatment plan with fixed values

At the start of the treatment, cancer cells are dying, but after some time t, the number of cancer cells increases again. The reason for this rebound is due to fixed values used for the treatment. Since the environment is in 3D, cancer cells outside of the range of beam proliferate, forming another mass (Figure 3.16)

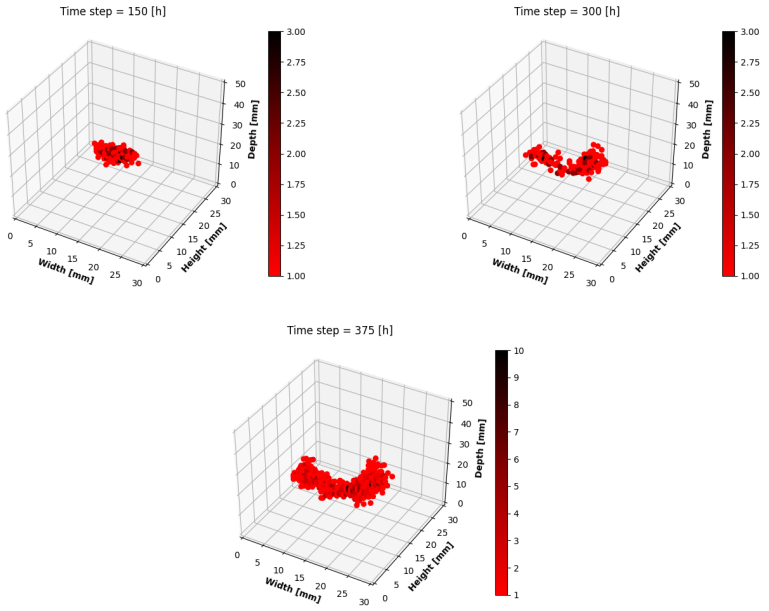

Figure 3.16: Tumor evolution before, during and after the treatment

3.3.2 DQN agent treatment

The action space of *DQN* agent is summarized in Table 2.5. In contrast to naive treatment presented in section 3.3.1, the mean beam energy is variable, impacting the tumor depth. Figure 3.17 shows the treatment plan proposed by the *DQN* agent. The strategy chosen by the agent is to increase the mean energy if the number of cancer cells is high, and decrease if the number of cancer cell decreases (Figure 3.18). The mean dose at TV is higher for the *DQN* treatment than the naive treatment, allowing to have much lower cancer cells count after 420 hours simulated. Unfortunately, it was not able to control the tumor. Figure 3.19 displays the state of the tumor in 3D. The tumor is split into three parts, and since no (x,y) displacement has been allowed to the *DQN* agent, it can not control the tumor growing away from the center.

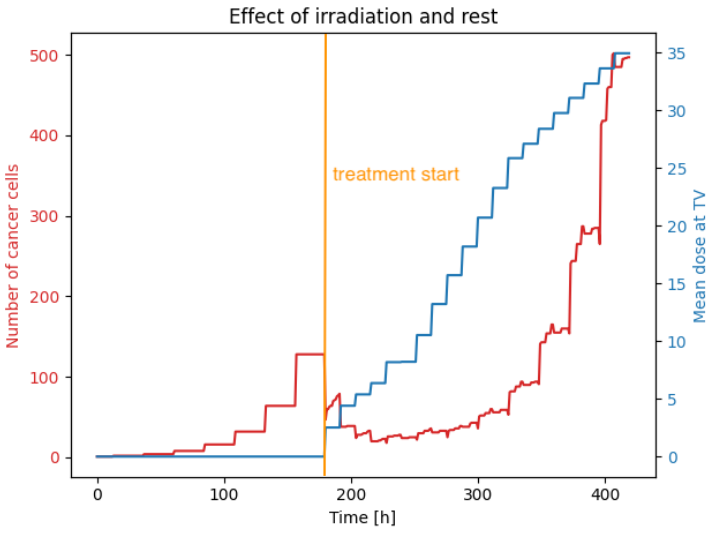

Figure 3.17: DQN treatment plan

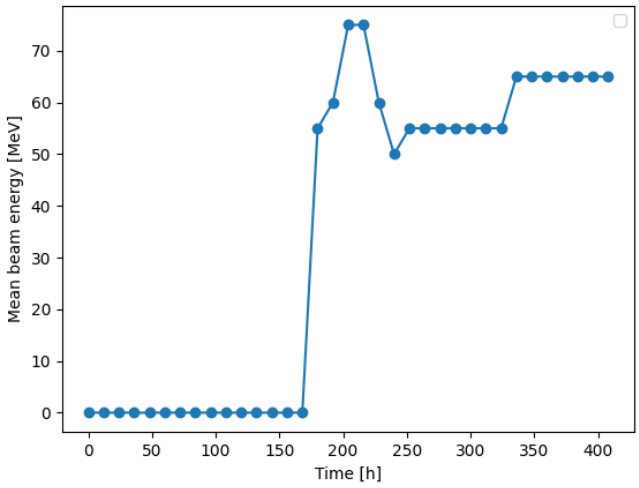

Figure 3.18: Mean beam energy during treatment

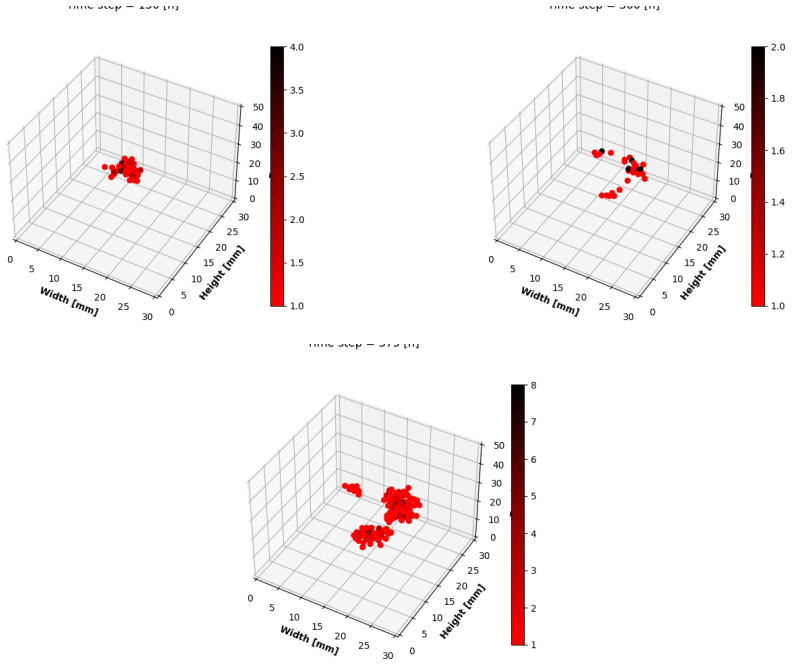

Figure 3.19: Tumor evolution before (left plot), during (right plot) and after the treatment (bottom plot)

3.3.3 DDPG agent treatment

The action space of *DDPG* agent is summarized in Table 2.6. In contrast to naive treatment presented in section 3.3.1, the mean beam energy is variable, impacting the tumor depth. Moreover, the beam (x,y) position is not fixed and is part of the agent's action space. Figure 3.20 shows the treatment plan proposed by the *DDPG* agent. The strategy chosen by the agent is similar to the *DQN* agent, i.e., increase the mean beam energy if the number of cancer cells is high and decrease if the number of cancer cells decreases (Figure 3.20). The mean dose at TV is slightly higher than the naive and *DQN* treatments since the beam is allowed to move in (x,y). Therefore, there are much lower cancer cells after 420 hours simulated. Again here, the tumor is not controlled at 100%. The agent chooses first to target the center of the tumor with high energy, killing most of the cancer cells. Some cancer cells survived to the first irradiation (Figure 3.22), allowing them to spread on the side. Then, the agent targets right side of the tumor (Figure 3.21) as it continues to spread on the side (Figure 3.22) but is unsuccessful due to limited interval of positions (Table 2.6).

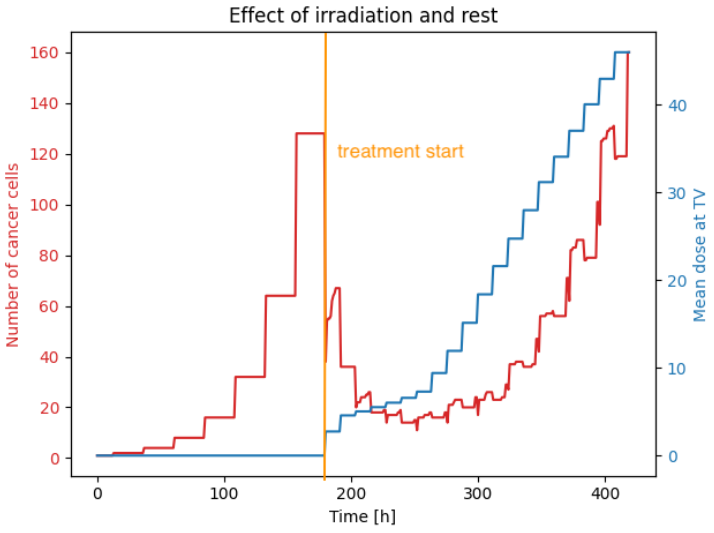

Figure 3.20: DDPG treatment plan

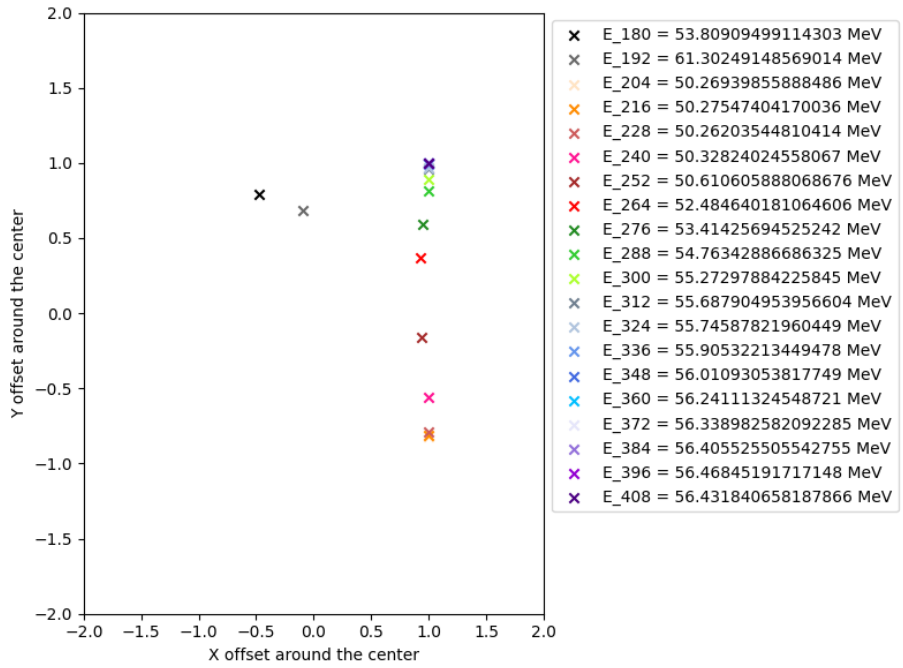

Figure 3.21: Position of the beam during treatment where E_XXX represents the mean beam energy at tick XXX

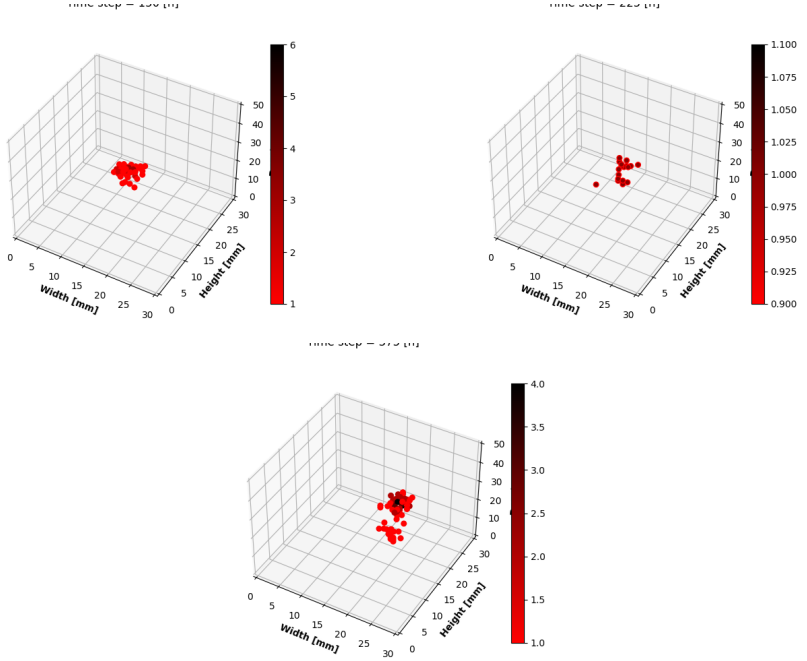

Figure 3.22: Tumor evolution before (left plot), during (right plot) and after the treatment (bottom plot)

Conclusion

General overview

This master thesis aimed to use deep reinforcement learning agents to propose a cancer treatment plan using biological cell proliferation and proton therapy simulation based on related work. Two agents were proposed, a DQN and a DDPG agent. Based on the state of the tumoral environment, the agents act using a beam of protons on this environment and receive rewards in return. Their goal was to maximize the sum of the cumulative reward, which is based on biological criteria. The results are averaged over 100 episodes and are summarized in Table 3.1.

Treatment	**Number of killed healthy cells**	**Number of cancer cells**
No treatment	0	101979
Naive	4869	2304
DQN	7418	500
DDPG	6711	160

Table 3.1: Post treatment results (420 h)

The agents were able to control the tumor at a certain point. Indeed, DQN and DDPG agents performed better than the naive approach at killing the cancer cells but killed more healthy cells than the naive methods. It can be concluded that more improvements are needed for the simulation and potentially used in clinics.

Improvements and further works

A lot of other physical, biological and, chemical reactions occur while the whole process is simulated. Some improvements are needed to make the simulation closer to reality. Here are some areas of investigation for further works :

- Implement the simulation in a language such as C++ that is more performant than Python. Indeed, the Python language is a limiting factor for the implementation of other functionality. Moreover, the number of epoch was limited due to the performance achieved by Python. It can be increased and give the best-trained network in terms of average loss.
- Even if the probability of nuclear reactions while proton is traversing material is relatively small, they still happen and need to be added to the simulation. Monte Carlo approaches can be used to simulate those reactions accurately.
- Actions such as a rotation of the beam. Practically in clinics, the beam is rotated to irradiate the patients.
- Use a phantom that is not uniquely composed of water but also air and bones to represent human body. In fact, this functionality is already implemented but not tested.
- Implement a vascular network which is more in phase with the reality than random sources.
- Adding OAR voxels into the simulation. They are already implemented but not added into the simulation. One can for instance investigate a reward function based on OAR.
- Bystander effects are not included in the simulation, for instance, mutation of healthy cells due to irradiation.

Bibliography

[1] S. Edmond, "Introduction to radiation therapy and proton therapy." Feb. 2021.

[2] National Institutes of Health, "Non ionizing radiation."

[3] T. O'Shea and M. Foley, *Monte Carlo Simulation of Medical Accelerator Electron Treatment Heads*. PhD thesis, Feb. 2012.

[4] O. Desouky and G. Zhou, "Biophysical and radiobiological aspects of heavy charged particles," *Journal of Taibah University for Science*, vol. 10, 04 2015.

[5] J. Metzcar, Y. Wang, R. Heiland, and P. Macklin, "A Review of Cell-Based Computational Modeling in Cancer Biology," *JCO clinical cancer informatics*, vol. 3, pp. 1–13, Feb. 2019.

[6] G. Moreau, "Optimizing radiation therapy dose planification using deep reinforcement learning," Master's thesis, Ecole polytechnique de Louvain, Université catholique de Louvain, 2020. Prom. : Macq, Benoît.

[7] The science hive, "The Cell Cycle, Mitosis & Meiosis (A Level)."

[8] M. Joiner and A. J. v. d. Kogel, *Basic clinical radiobiology*. CRC Press/Taylor Francis Group, 2009.

[9] W. D. Newhauser and R. Zhang, "The physics of proton therapy," *Physics in Medicine and Biology*, vol. 60, pp. R155–R209, Mar. 2015. Publisher: IOP Publishing.

[10] E. Vitti and J. Parsons, "The radiobiological effects of proton beam therapy: Impact on dna damage and repair," *Cancers*, vol. 11, p. 946, 07 2019.

[11] Massachusetts General Hospital Radiation Oncology, "Bragg peak."

[12] R. S. Sutton, F. Bach, and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press Ltd, 2018.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2019.

[14] G. Sunny, "Deep Deterministic Policy Gradient (DDPG): Theory and Implementation."

[15] CDC, "Cancer Clusters - Occupational Cancer | NIOSH | CDC," June 2020.

[16] H. Sung, J. Ferlay, R. L. Siegel, M. Laversanne, I. Soerjomataram, A. Jemal, and F. Bray, "Global Cancer Statistics 2020: GLOBOCAN Estimates of Incidence and Mortality Worldwide for 36 Cancers in 185 Countries," *CA: a cancer journal for clinicians*, vol. 71, pp. 209–249, May 2021.

[17] Cancer Research UK, "Lung cancer survival statistics," May 2015.

[18] L. Gong, Y. Zhang, C. Liu, M. Zhang, and S. Han, "Application of radiosensitizers in cancer radiotherapy," Feb 2021.

[19] H. Murshed, *Fundamentals of radiation oncology: physical, biological, and clinical aspects*. Academic Press, 2019.

[20] N. O'Neil, "An agent based model of tumor growth and response to radiotherapy," 2012.

[21] A. Jalalimanesh, H. S. Haghighi, A. Ahmadi, and M. Soltani, "Simulation-based optimization of radiotherapy: Agent-based modeling and reinforcement learning," *Math. Comput. Simul.*, vol. 133, pp. 235–248, 2017.

[22] G. M. Cooper, "The eukaryotic cell cycle," Jan 1970.

[23] R. Weissleder, J. Wittenberg, M. G. Harisinghani, and J. W. Chen, *Primer of diagnostic imaging*. Elsevier Health Sciences, 2011.

[24] C. M. van Leeuwen, A. L. Oei, J. Crezee, A. Bel, N. a. P. Franken, L. J. A. Stalpers, and H. P. Kok, "The alfa and beta of tumours: a review of parameters of the linear-quadratic model, derived from clinical radiotherapy studies," *Radiation Oncology (London, England)*, vol. 13, p. 96, May 2018.

[25] G. G. Powathil, D. J. A. Adamson, and M. A. J. Chaplain, "Towards predicting the response of a solid tumour to chemotherapy and radiotherapy treatments: clinical insights from a computational model," *PLoS computational biology*, vol. 9, no. 7, p. e1003120, 2013.

[26] S. Christiane, "Introduction to radiobiology," tech. rep., Brigham Women's Hospital, 2020.

[27] S. Gulliford and K. Prise, "Relative Biological Effect/Linear Energy Transfer in Proton Beam Therapy: A Primer," *Clinical Oncology*, vol. 31, pp. 809–812, Dec. 2019.

[28] L. H. Praestegaard, "Interactions of the proton with matter," November 2020. Aarhus University Hospital.

[29] T. Borgi, N. Zoghlami, M. Abed, and N. Mohamed Saber, "Big data for operational efficiency of transport and logistics: A review," pp. 113–120, 07 2017.

[30] D. Silver, "Lecture 1: Introduction to reinforcement learning." DeepMind.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[32] M. Fippel and M. Soukup, "A monte carlo dose calculation algorithm for proton therapy," *Medical physics*, vol. 31, pp. 2263–73, 09 2004.

[33] S. Kevin, S. Wuyckens, and E. Sterpin, "Monte carlo lab." Feb. 2021.

[34] V. François-Lavet *et al.*, "Deer." `https://deer.readthedocs.io/`, 2016.

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