Master Thesis

Optimizing treatment pathways of cancer patients using process mining and reinforcement learning

Author: Roza Hoek Spaans (2783463)

First supervisor:	Svetlana Dubinkina
Second reader:	Ger Koole
Company supervisor:	Louis Juan

August 29, 2025

Abstract

Hospitals often face challenges with managing patient care efficiently. Healthcare is a domain where many decisions need to be made daily, and data can be helpful here. This is also true for oncology, where there are different treatment options and patient responses to treatment can vary. The aim of this study is to optimize treatment pathways for cancer patients in the gynecology department of a Dutch hospital by using process mining and reinforcement learning.

The initial plan was to combine process mining with reinforcement learning, but due to some limitations, the methods were applied separately. Using process mining, treatment pathways were visualized in process models at the department level and specifically for the radiotherapy department. The process model on department level provides insight into how patients flow through the hospital departments during their treatment period, including which departments are central points during treatment and which departments are closely connected. This gives information about how the departments are involved in the treatment process. For the radiotherapy department, the main treatment trajectory became clear, where patients receive teletherapy, brachytherapy and hyperthermia or a combination of these therapies.

Reinforcement learning was then applied to optimize chemotherapy dosing for cancer patients, where different reward functions and algorithms were compared. For simulating the data, a mathematical model based on ordinary differential equations was used. The results showed that the reward function plays an important role in the reinforcement learning algorithm. A reward function that also includes information about the patient's well-being succeeds in reducing the tumor while providing a more careful treatment approach. In contrast, a reward function that only focuses on reducing tumor cells also succeeds in reducing the tumor but can be too aggressive. Additionally, Deep Q-Networks outperformed Q-learning by succeeding in reducing the tumor with less drugs.

Contents

List of Figures

List of Tables

1

Introduction

Hospitals face numerous complex challenges, such as managing high patient volumes, allocating resources, and providing timely care. Decisions need to be made quickly and effectively on a daily basis, often with limited information. Data becomes valuable as it can provide insights into patient flows, resource usage, and potential inefficiencies. This data can be analyzed resulting in giving the ability for hospitals to make informed decisions, improving operations, enhance patient care, and reduce costs. In a high-pressure healthcare environment, where decisions must be made quickly and accurately, data-driven insights can play a crucial role in improving efficiency and care.

An important domain in this context is oncology, which is the medical branch dedicated to the diagnosis, treatment, and prevention of cancer [3]. Cancer remains a major cause of mortality. Among people aged between 30 and 69 years old, cancer is one of the three leading causes of death in 177 of 183 countries [4]. Oncologists must remain informed about newly approved therapeutics, at the same time they are required to evaluate the advantages and disadvantages of various treatment regimens to determine the most suitable option for their patients [5]. Costs of cancer treatment are increasing due to factors such as rising drug prices, increased use of advanced therapies, and population changes, leading to substantial financial burdens on patients and healthcare systems [6, 7]. The diversity of cancer types and the variable patient responses to treatments, coupled with the dynamic nature of cancer progression, makes this a challenging problem [3].

That is why data is becoming increasingly valuable in this sector. Artificial intelligence (AI) has the potential to help doctors in improving medical decisions or even substituting human judgement in specific areas of healthcare. AI techniques have the capability to uncover information hidden in the large amount of data, thereby assisting in the clinical decision-making processes. Moreover, AI can support physicians in making a diagnosis, predicting the spread of diseases and customizing treatment paths [8, 9].

One approach to improving cancer treatment is the analysis of observational data to optimize treatment pathways. A treatment pathway represents the sequence of diagnoses and treatments a patient undergoes throughout their care. Given the complexity of oncology treatments, structuring and refining these pathways can enhance both patient outcomes and hospital efficiency. By analyzing past treatment decisions and their outcomes, effective strategies can be identified that may reduce costs while maintaining high-quality care [10].

1.1 Problem description

The goal of this study is to optimize the treatment pathways for cancer patients in the gynecology department of a Dutch hospital. A treatment pathway is the process a patient goes through and it consists of diagnoses and treatments. Efficient management of clinical pathways is crucial to ensure timely and effective care. In this study, this problem is addressed using process mining and reinforcement learning

(RL). Applying process mining in healthcare can help physicians understand the actual execution of processes, discovering new processes, and finding improvement opportunities [11]. RL offers a solution for optimal decision-making in healthcare, addressing challenges like noisy, multi-dimensional, incomplete data, complex dynamics, and sequential decisions with delayed feedback [12].

In this study, process mining will be applied to analyze the flow of patients through the departments, where useful insights of the treatment process will be identified. Then with the knowledge of that, reinforcement learning is applied to support the dynamic decision making of these treatment pathways, making them more efficient and reducing unnecessary procedures. In particular, reinforcement learning is used to optimize drug dosing of chemotherapy, with the aim of improving dosing precision, reducing side effects, and enhancing overall treatment efficacy.

The purpose of this study is to investigate how process mining and reinforcement learning can be leveraged to enhance the efficiency of treatment pathways for cancer patients and improve the quality of patient care. The research question addressed in this study is:

> *How can process mining and reinforcement learning be applied to optimize the treatment pathways of cancer patients in the gynecology department of a Dutch hospital, with a focus on improving efficiency and patient outcomes?*

By addressing this question, this study will contribute to the development of data-driven decision support systems in oncology, improving personalized treatment strategies.

1.2 Relevance Accenture

Accenture is a global consultancy firm working across different industries, including healthcare and public services. This study aligns with Accenture's focus on innovation and social impact. The company helps many healthcare clients improve their processes using data-driven solutions. By applying advanced AI techniques, this study supports the company's goal of using technology to improve healthcare.

The development of innovative solutions for optimizing cancer treatment pathways using process mining and reinforcement learning will provide valuable insights into how these technologies can be applied in the healthcare sector to improve efficiency, reduce costs, and enhance patient outcomes. Additionally, the findings could strengthen Accenture's position in healthcare consulting by offering data-driven solutions that evolve with new developments in AI, thus supporting Accenture's focus on innovation. Optimizing cancer treatment pathways not only makes healthcare more efficient but also helps provide better care for patients, strengthening Accenture's role in driving positive change in the health sector.

1.3 Thesis outline

Chapter 2 provides an introduction to process mining and reinforcement learning, with related work about the application of these methodologies in healthcare. Chapter 3 describes the data used in this study, including a real-world dataset and a mathematical model to simulate data. Chapter 4 outlines the methodology, detailing the use of the heuristic miner algorithm, Q-learning, and Deep Q-Networks. Chapter 5 presents the experimental setup. Chapter 6 discusses the results of both process mining and reinforcement learning models. Finally, Chapters 7 and 8 conclude the thesis with findings, limitations, and future research directions.

2

Literature review

This chapter provides an introduction to process mining and reinforcement learning and reviews the relevant literature for this study. Section 2.1 will focus on process mining, and Section 2.2 on reinforcement learning.

2.1 Process mining in healthcare

This section will give an introduction to process mining and related work where process mining is applied in the healthcare sector will be discussed.

2.1.1 Introduction to process mining

Process mining is a technique used to analyze and improve business processes by extracting insights from data generated by information systems during process execution. The data used in process mining is stored in an event log. An event log consist of traces, a trace is a sequence of activities. These activities are the events. Each event represents a step in the process. A unique identifier for the case, the specific activity, and a reference for the execution time are documented for every event. Table 2.1 illustrates a simple representation of an event log. Additional information can also be included such as costs, resources or the type of the event [13, 14].

Case id	**Activity**	**Timestamp**
...	...	...
Case 1	Activity A	04-01-2025 11:30:40
Case 1	Activity B	04-01-2025 12:10:30
Case 1	Activity C	04-01-2025 12:34:50
Case 2	Activity A	07-01-2025 09:11:43
Case 2	Activity C	07-01-2025 10:51:45
...	...	...

Table 2.1: Example of a simple event log data.

There are three basic types of process mining: discovery, conformance checking, and enhancement. In process discovery, the goal is to develop a process model from the event log, without relying on any a priori model. The aim is to create a model that accurately reflects the observed behavior in the data while avoiding overfitting or underfitting, which can either misrepresent or simplify the actual process dynamics. In contrast, both conformance checking and enhancement start with an existing model. Conformance checking involves comparing the recorded behavior in the event log with the expected behavior defined in the model. This comparison can highlight deviations and bottlenecks. Enhancement, on the other hand, seeks to improve or extend the existing model based on insights derived from the event data [13, 15].

2.1.2 Related work about process mining in healthcare

Process mining is a valuable tool in healthcare, offering insights into complex clinical workflows and enabling data-driven improvements in patient care. The event log represents the real-life process, and analyzing this and making it interpretable for clinicians or doctors can support them in decision making [14].

There are various studies who applied process mining within the healthcare domain. Rojas et al.[11] provided a comprehensive review of how process mining has been applied in healthcare, demonstrating different approaches for process mining. It reviewed current work in this field, and gives insights in the most used algorithms, tools, and the medical fields of healthcare where process mining is applied. For example, many of the reviewed studies employed control flow discovery techniques, such as the heuristics miner algorithm, to derive the actual flow of activities of the process. Oncology is one of the medical fields where process mining has been applied to the most.

Muñoz-Gama et al. [14] also conducted a literature review, focusing on key characteristics and challenges in the field. Healthcare continues to evolve, yet it remains bounded by numerous protocols and guidelines that must be strictly followed. Additionally, the healthcare domain typically adopts a patient-centered view, whereas process mining generally aims to identify and analyze recurring process patterns. Moreover, since healthcare involves highly sensitive data, issues related to data access and quality can significantly limit the effectiveness and reliability of process mining outcomes.

Rabbi et al. [16] applied different process mining algorithms to a healthcare dataset. They evaluated four distinct algorithms that can be implemented in Python: the alpha miner, directly-follows graph, heuristic miner, and inductive miner. The alpha miner discovers process models from event logs but struggles with noise, incomplete data, and loops. The directly-follows graph represents activities as they occur in sequence, though it can lead to complex models that are difficult to interpret. The heuristic miner improves on this by handling incomplete data and mining more complex behavior, although it may introduce many dependencies. The inductive miner aims to build structured and stable models, but it depends on identifying recurring patterns, which can be a limiting factor. The researchers of this study investigated the process of the installation of Central Venous Catheter procedures of students. Through process mining, insights were gained into potential issues of the process, offering assistance to medical students. For example, many students encountered difficulties in all the steps from *tool preparation* to *probe location*, as this phase consistently required more time than others in the procedure.

Applying process mining in healthcare comes with several important challenges. Clinical processes are often complex and show high variability, as each patient may follow a different path. This makes it difficult to create clear and structured process models. In addition, hospital data comes from different sources, which can make data preparation and integration difficult. The knowledge of domain experts is essential for applying process mining correctly, as they help choose the right methods and interpret the results. The models that are discovered are often complex and less structured, which makes them harder to visualize and understand, especially in the healthcare domain. Another challenge is that healthcare processes change over time, so models need to be updated regularly. Finally, patient data is sensitive and must be handled carefully to ensure privacy and security [11, 14].

2.2 Reinforcement learning in healthcare

This section provides an introduction to reinforcement learning and reviews existing research on its application in chemotherapy drug dosing.

2.2.1 Introduction to reinforcement learning

Reinforcement learning (RL) is a branch of machine learning where an agent learns to make decisions by interacting with an environment to maximize the reward. RL stands apart from supervised learning, where during the training process of the model knowledge of the correct output is available. In RL this is not the case, which introduces a fundamental challenge, the trade-off between exploration and exploitation. The agent has to exploit its current knowledge to gain rewards, but also explore new possibilities to discover better actions for the future [17, 18].

In RL problems, an agent and environment interact with each other. The agent can be seen as the *decision-maker*, and the environment is everything the agent can interact with. The interaction between the agent and the environment is visualized in Figure 2.1. In this loop, the agent interacts with the environment by selecting actions according to a policy. These actions move the system from one state to another, while the agent simultaneously receives observations and reward signals that guide its learning process [17].

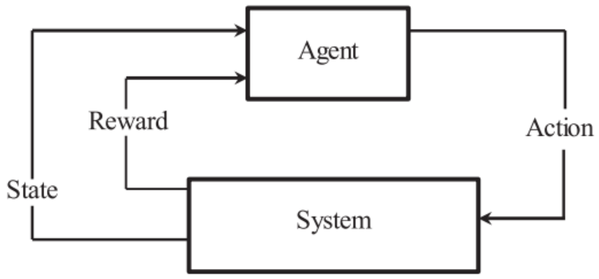

Figure 2.1: Interaction between the agent and the environment in reinforcement learning [1].

A reinforcement learning problem can be modeled as a Markov decision process (MDP). A MDP captures the dynamics of the problem and consist of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ between states, i.e. the probability of going from state $s \in \mathcal{S}$ to state $s' \in \mathcal{S}$ after choosing action $a \in \mathcal{A}$, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ [18].

At each discrete time step t, the agent observes the current state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$. Subsequently, based on the selected action it transitions to the following state, and based on the next state the agent receives a reward $r_t \in \mathbb{R}$. The reward quantifies how good the selected action is. This allows the agent to adjust the decision making process to learn the best policy π [17].

Reinforcement learning methods can be broadly categorized into model-based and model-free approaches. In model-based reinforcement learning, the agent learns or is provided with a model of the environment's dynamics, allowing it to plan and predict future states and rewards. In contrast, model-free reinforcement learning does not use an explicit model of the environment's transition dynamics. Instead it learns optimal behavior directly from interactions, typically through trial and error. This makes model-free methods computationally efficient but less flexible, while model-based methods offer greater flexibility at the cost of higher computational complexity. [19, 20].

Furthermore, RL algorithms can be divided into value-based and policy-based methods. Value-based methods focus on estimating a state-value function $V^\pi(s)$ or action-value function $Q^\pi(s,a)$ that represents the expected return of states or state-action pairs when starting in s and following policy π. That means how good it is to be in a given state or how good a given action is after being in a given state.

Discounting is often used to calculate the return in RL. Discounting reduces the contribution of future rewards compared to immediate rewards. In Equation 2.1 the expected discounted return is expressed.

$$R_t = r_{t+1} + \gamma \cdot r_{t+2} + \gamma^2 \cdot r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.1}$$

where r_t is the reward at time t and $\gamma \in [0,1]$ is the discount rate, which lowers the value of future rewards compared to immediate ones.

The expected discounted return R_t at time step t is the sum of all future rewards multiplied by powers of a discount factor $\gamma \in [0,1]$. This ensures that the total return remains finite even for infinite-horizon problems and allows the agent to prioritize immediate rewards while still considering long-term outcomes [17, 18]. The state-value function $V^\pi(s)$ then uses the expected discounted return R_t to quantify how favorable it is for an agent to be in state s when following policy π. For MDPs, the state-value function is defined as:

$$V^\pi(s) = E_\pi \{R_t \mid s_t = s\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s \right\}, \tag{2.2}$$

where E_π is the expected value when the agent follows policy π, R_t is the expected discounted return at time t, $\gamma \in [0,1]$ is the discount rate and s_t is the state at time t.

Similarly, the expected return of taking action a starting in state s under policy π, which is the action-value function $Q^\pi(s,a)$ can be defined as:

$$Q^\pi(s,a) = \mathbb{E}_\pi \{R_t \mid s_t = s, a_t = a\} = \mathbb{E}_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,\middle|\, s_t = s, a_t = a \right\}, \tag{2.3}$$

here, E_π, R_t, γ and s_t are the same as in the value function $V^\pi(s)$, and a_t is the action at time t [17].

Policy-based methods, on the other hand, directly learn a policy that maps states to actions without relying on value function estimation. These methods are especially useful in high-dimensional or continuous action spaces [21]. A lot of policy-based methods use a policy gradient technique to optimize a performance objective. The agent makes decisions based on policy $\pi(s,a,\theta)$, where $s \in \mathcal{S}$ is the state, $a \in \mathcal{A}$ is the action and θ is the parameter vector. In policy gradient techniques, the parameterized policy is adjusted according to the gradient of the performance. The performance objective is defined as the expected return under policy π and can be formulated as:

$$J(\pi) = E_\pi\{R_0\} \tag{2.4}$$

where R_0 is the expected discounted reward with start state s_0. This objective cares about the long-term reward that is obtained. Using the gradient $\nabla_\theta J(\pi)$, the parameter vector θ is computed, by:

$$\nabla_\theta J(\pi) = \sum_s d^\pi(s) \sum_a Q^\pi(s,a) \nabla_\theta \pi(s,a), \tag{2.5}$$

here $d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s \mid \pi)$ is defined as the discounted weighting of states encountered starting at s_0 and $Q^\pi(s,a)$ is the action-value function under policy π [22].

Additionally, actor-critic methods combine the strengths of both approaches by using a value function (critic) to evaluate the policy (actor), achieving greater stability and efficiency in learning [23].

2.2.2 Related work about RL in healthcare

Reinforcement learning has become increasingly valuable in healthcare, addressing a wide range of application domains such as automated medical diagnosis, health management, resource allocation, and scheduling. Among these, dynamic treatment regimes (DTRs) have emerged as a promising approach for adjusting treatment decisions over time based on a patient's evolving health status and history. In most cases, treatment isn't just a single decision but involves making choices at several points as the patient's condition develops. RL is well-suited for this problem, as it can learn policies that optimize long-term outcomes by taking into account both immediate and future consequences of each treatment decision.

DTRs are particularly relevant for chronic or complex diseases where treatment needs to be continuously adapted. Examples include cancer, diabetes, anemia, and HIV, where the progression of the disease and patient response to therapy vary significantly over time [12].

Numerous academic studies have explored the use of RL in optimizing medical treatments. One area of particular focus is chemotherapy dosing, where treatment decisions must be adapted over time to balance effectiveness and side effects. These studies highlight how RL can support personalized, sequential decision-making in real-world clinical settings, demonstrating its value in developing more effective and individualized care strategies.

Yang et al. [24] provided a literature review about the strategies of RL for chemotherapy treatments. It gives insights about how reinforcement learning methods can be applied to optimize DTRs, and about existing mathematical models used to simulate patient responses following chemotherapy. They emphasize the relevance of adaptive control in treatment personalization, and examine how offline and supervised RL approaches have been leveraged to improve the understanding of dose adjustment strategies.

There are different studies who have employed mathematical models to develop reinforcement learning frameworks for chemotherapy dosing. Zhao et al. [25] used a mathematical model where only the tumor size and toxicity of the tumor are included. They applied Q-learning, where the Q-function is approximated by using support vector regression and extremely randomized trees. With taking into account the toxicity of cancer, their method is able to reduce the burden of the tumor.

Q-learning is also utilized in a study by Padmanabhan et al. [1], where they employed a mathematical model to simulate three different patient scenarios and trained the RL model accordingly. Their proposed algorithm does not require prior knowledge of the system's dynamics. However, the model must be retrained for each new scenario.

A study by Mashayekhi et al. [26] proposed a deep reinforcement learning approach and compared its performance with the results of Padmanabhan et al. [1]. Their method utilizes an algorithm capable of handling continuous state and action spaces, leading to a reduction in both drug intake and treatment duration, while also increasing the rate of tumor cell reduction.

While the results are encouraging, applying RL to cancer treatment presents several challenges. As noted by Mashayekhi et al. [26], accurately modeling tumor dynamics and treatment response in continuous state and action spaces is complex. Another difficulty is designing reward functions that effectively balance treatment effectiveness with potential side effects. Padmanabhan et al. [1] also highlight the importance of ensuring that the learned treatment strategies are robust and clinically meaningful, which can be difficult when working with limited or noisy data. Furthermore, translating these models from simulation to real clinical settings requires careful validation and attention to ethical and safety considerations [24].

3

Data

Section 3.1 of this chapter describes the data used for process mining. In Section 3.2 the tumor growth model utilized for optimizing chemotherapy dosages is explained.

3.1 Description of the hospital data

The dataset used in this study is publicly available and originates from an initial BPI challenge [27]. This data is a real-life event log obtained from a Dutch academic hospital. The log captures detailed information on when specific activities occurred and relevant attributes related to the treatment process. While some anonymization has been applied, the data remains largely as it was recorded in the hospital's systems. Certain attributes appear multiple times for the same patient, indicating that they went through different, and sometimes overlapping, phases of care.

The dataset is an event log and consist of 1143 traces, each corresponding to a single patient. Each trace represents the treatment pathway of a patient, expressed as a sequence of events. For every event, both the activity and the corresponding department in which it occurred are recorded. In total, there are 43 departments represented. The event log covers a time period from January 2005 to March 2008.

Each trace includes various attributes per event, such as the activity, department, timestamp, number of executions, specialism code, hospital section, and activity code. Additionally, general information about the patient's overall treatment process is included, such as diagnosis, start date, age, and treatment code.

Patients can be diagnosed with different diagnoses over time. In the data, the diagnosis is stored both as text and as a diagnosis code. There are 11 unique diagnosis codes in total, but the description of the diagnosis can differ when the code remains the same. Table 3.1 provides an overview of the diagnosis codes, the corresponding regions of diagnosis, and the frequency of each code in the dataset. While the dataset also includes a treatment code, it does not provide any description explaining what the treatment entails.

Diagnosis code	Region	Count
M11	vulva	176
M12	vagina	22
M13	cervix uteri	368
M14	corpus uteri	145
M15	corpus uteri	17
M16	ovary	235
106	vulva, vagina, corpus uteri, cervix uteri	301
821	ovary	49
822	cervix uteri	132
823	ovary, corpus uteri	16
839	ovary, vulva	21

Table 3.1: Diagnoses in the event log [2].

3.2 Data simulation

To simulate a patient's response to chemotherapy, a mathematical model is employed. There are several models available that describe tumor growth and shrinkage, many of them are based on ordinary differential equations (ODEs). The model used in this work was originally proposed by de Pillis and Radunskaya [28]. It is a four-state model that tracks the number of tumor cells T, immune cells I, normal cells N, and the concentration of the drug in the bloodstream C. It should be mentioned that this tumor growth model is not specific for one cancer type. This four-state model is described by Equations (3.1)–(3.4), where $N(t) = x_1(t)$, $T(t) = x_2(t)$, $I(t) = x_3(t)$ and $C(t) = x_4(t)$.

$$\dot{x}_1(t) = r_2 x_1(t)\left(1 - b_2 x_1(t)\right) - c_4 x_1(t) x_2(t) - a_3 x_1(t)\left(1 - e^{-x_4(t)}\right) \tag{3.1}$$

$$\dot{x}_2(t) = r_1 x_2(t)\left(1 - b_1 x_2(t)\right) - c_2 x_3(t) x_2(t) - c_3 x_1(t) x_2(t) - a_2 x_2(t)\left(1 - e^{-x_4(t)}\right) \tag{3.2}$$

$$\dot{x}_3(t) = s + \frac{\rho x_3(t) x_2(t)}{\beta + x_2(t)} - c_1 x_3(t) x_2(t) - d_1 x_3(t) - a_1 x_3(t)\left(1 - e^{-x_4(t)}\right) \tag{3.3}$$

$$\dot{x}_4(t) = -d_2 x_4(t) + u(t) \tag{3.4}$$

Here, $u(t)$ represents the drug infusion rate at time t. In this model, normal and tumor cells grow logistically, while tumor cells are also reduced by immune attack and interaction with healthy tissue. Immune cells are produced at a constant rate and are activated by the tumor, but they are also weakened by the tumor itself.

The treatment affects all three cell types through similar terms involving $-(1 - e^{-x_4(t)})$, which reflect the drug's toxic effects that increase with concentration. This term reduces normal cells, kills tumor cells, and suppresses immune cells. The drug concentration $x_4(t)$ decreases naturally over time and increases with the input $u(t)$ [23].

The maximum value of drug dosage u_{max} is 1. Table 3.2 gives a description of the parameters of the model, and the values used in this study, original from [28].

Parameter	Parameter description	Value	Unit
a_1	Fractional immune cell kill rate	0.2	mg^{-1} day^{-1}
a_2	Fractional tumor cell kill rate	0.3	mg^{-1} day^{-1}
a_3	Fractional normal cell kill rate	0.1	mg^{-1} day^{-1}
b_1	Reciprocal carrying capacity of tumor cells	1	cell^{-1}
b_2	Reciprocal carrying capacity of normal cells	1	cell^{-1}
c_1	Immune cell competition term (competition between tumor cells and immune cells)	1	cell^{-1}day^{-1}
c_2	Tumor cell competition term (competition between tumor cells and immune cells)	0.5	cell^{-1}day^{-1}
c_3	Tumor cell competition term (competition between normal cells and tumor cells)	1	cell^{-1}day^{-1}
c_4	Normal cell competition term (competition between normal cells and tumor cells)	1	cell^{-1}day^{-1}
d_1	Immune cell death rate	0.2	day^{-1}
d_2	Decay rate of injected drug	1	day^{-1}
r_1	Per unit growth rate of tumor cells	1.5	day^{-1}
r_2	Per unit growth rate of normal cells	1	day^{-1}
s	Immune cell influx rate	0.33	cell day^{-1}
α	Immune threshold rate	0.3	cell
ρ	Immune response rate	0.01	day^{-1}

Table 3.2: Model parameters, their descriptions, values, and units [1].

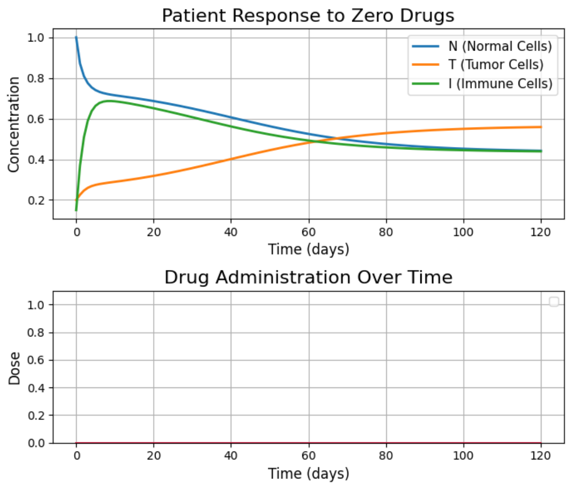

Figure 3.1: Response of the patient to zero drugs over time.

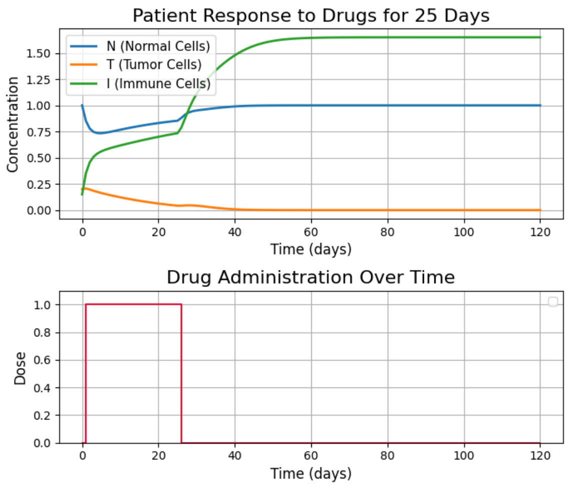

Figure 3.2: Response of the patient to maximum dose of drugs for 25 days.

To provide a deeper understanding of the tumor growth model, Figures 3.1 and 3.2 illustrate the dynamics of normal, tumor, and immune cells in absence of drugs and when full doses of drugs are administered for 25 days, respectively.

Figure 3.1 shows that without drugs, the number of normal cells gradually decreases, while tumor cells keep growing until they reach a stable level. The immune cells initially react and grow in number, but over time their effect fades, and they begin to decline. This underscores the inability of the immune system to completely suppress tumor development by itself.

In contrast, Figure 3.2 illustrates that with 25 days of drug treatment, tumor cells shrink rapidly and almost disappear. Normal cells recover and stabilize, while immune cells continue to rise, indicating that the therapy reduces the tumor burden and strengthens the immune system of the patient.

4

Methodology

In this chapter the methods are discussed. In Section 4.1 an algorithm for process mining will be discussed, the heuristic miner algorithm. Section 4.2 focuses on the reinforcement learning methods.

4.1 Process mining

With process mining, the goal is to analyze the treatment pathways of the patients. Data is stored as an event log, where from different patients the sequence of activities in the hospital are recorded. Various algorithms exist to discover process models from such data. This section focuses on the heuristic miner algorithm. This algorithm is one of the most widely used techniques for extracting process models from real-life, often noisy, event logs.

4.1.1 Heuristic miner algorithm

The heuristic miner is a process discovery algorithm in the field of process mining, designed to construct process models from event logs. It is particularly effective in handling real-life event logs, which often contain noise, incomplete traces, and complex behaviors. The heuristic miner takes frequent patterns into account and focuses on dominant behavior [29].

The algorithm follows a step-by-step approach to derive a process model from the event log. It starts by creating a dependency graph which forms a foundation for the model. For each pair of activities a and b, it calculates a dependency measure $a \Rightarrow b$, which expresses the likelihood that b follows a in the process. This is formalized in Equation 4.1.

$$a \Rightarrow b = \frac{|a > b| - |b > a|}{|a > b| + |b > a| + 1} \tag{4.1}$$

Here, $|a > b|$ denotes the number of times activity b directly follows activity a in the event log. The resulting value ranges between -1 and 1, where values closer to 1 indicate a strong dependency from a to b, while values closer to 0 or negative indicate weak relationships [29].

To filter out noise and weak patterns, the algorithm uses several thresholds. These thresholds help retain only the most relevant relations:

- *Dependency threshold:* Minimum value for the dependency measure $a \Rightarrow b$ required for a relation to be included in the model.
- *Positive observations threshold:* Minimum number of times a direct relation must occur to be considered.
- *Relative-to-best threshold:* Ensures that only relations whose dependency measures are within a certain percentage of the strongest observed dependency are included.

- *Length-one loops threshold:* Determines the minimum frequency with which an activity must directly follow itself ($a \to a$) in order to be considered relevant.
- *Length-two loops threshold:* Specifies the minimum frequency for detecting alternating loop patterns of the form $a \to b \to a$ [29, 30].

Once the significant relations have been identified, the algorithm proceeds to construct the control-flow structure of the model. This involves determining whether activities occur in sequence, in parallel, or as part of a decision point. For this, additional measures are computed to detect split and join patterns.

One important measure used at this stage is shown in Equation 4.2, which helps assess whether two activities b and c can occur in parallel after an activity a.

$$a \Rightarrow (b \wedge c) = \frac{|b > c| + |c > b|}{|a > b| + |a > c| + 1} \tag{4.2}$$

A higher value suggests that activities b and c can occur concurrently after a, rather than one strictly following the other. Based on this, the algorithm determines whether an activity has a parallel, sequential, or exclusive relation to its successor.

Finally, the result is translated into a process model, often represented as a directed graph or a Petri net. This model captures the dominant paths through the process as observed in the event log, while filtering out infrequent or noisy behaviors [29].

In summary, the heuristic miner algorithm builds a process model by first evaluating the relations, followed by calculating dependency measures. It then filters out weak or noisy connections using predefined thresholds. After that, it identifies the control-flow structure of the process, such as sequences, loops, and splits. Finally, the most significant and frequently observed behaviors are assembled into the final process model.

4.2 Reinforcement learning

This study explores the use of reinforcement learning to optimize chemotherapy drug dosage where different reward functions are investigated to guide the agent's behavior effectively. As a starting point, Q-learning is implemented due to its simplicity and clarity. The algorithm estimates the value of different actions in each state using a lookup table, which makes it easy to interpret and analyze. This makes Q-learning well-suited for understanding the dynamics of the environment and evaluating initial agent performance. Next, another method is implemented, Deep Q-Networks.

4.2.1 Q-learning

Q-learning is a model-free, value-based reinforcement learning algorithm that operates in environments with discrete state and action spaces. The goal of the Q-learning agent is to learn the optimal policy through direct interaction with the environment, without initial knowledge about the policy [31]. The learning process is centered around maintaining a Q-table, where each entry $Q(s, a)$ estimates the expected cumulative reward for taking action a in state s and thereafter following the optimal policy. The agent interacts with the environment over discrete time steps, observing a sequence of state-action-reward transitions. After each transition, the Q-value is updated according to the rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t \left[r_t + \gamma \max_{a' \in \mathcal{A}} Q_t(s_{t+1}, a') - Q_t(s_t, a_t) \right] \tag{4.3}$$

Here, $\gamma \in [0, 1]$ is the discount factor and α_t is the learning rate at time t, determining how strongly new information overrides existing estimates. It is assumed that all initial Q-values are given. Over time, the agent learns by updating the Q-table based on experience. Under standard assumptions, including a decaying learning rate and sufficient exploration, Q-learning is proven to converge to the optimal action-value function with probability one [31, 32].

At the end of the learning process, the final Q-table represents the optimal action-value function. The agent can derive the optimal policy π^* by selecting the action that maximizes the Q-value in each state:

$$\pi^*(s) = \arg\max_{a \in \mathcal{A}} Q(s, a) \tag{4.4}$$

This policy is guaranteed to be optimal in the limit, provided the learning process has converged [17, 31].

4.2.2 Deep Q-Networks

The Q-learning algorithm has several drawbacks. While it is simple and easy to interpret, it may not be suitable for more complex problems. First of all, Q-learning is not optimal for large spaces. It can be very time-consuming for Q-learning to fill the Q-table [26]. Another known drawback of the Q-learning algorithm is that it overestimates the action-values. Because of the maximization step in the algorithm it prefers overestimation over underestimation. As a result, the algorithm ends up learning action values that are unrealistically high [33].

Q-learning can also suffer from slow convergence because it learns in an environment without initial knowledge [34]. It is shown that Q-learning converges to optimal action-values, but that requires that all actions are sampled multiple times during training in all states. But this requires a lot of exploration and time, especially in environments with large state and action spaces [31]. Therefore, in this study, a second methodology is explored to address these issues, Deep Q-Networks (DQN) in particular.

DQN extends Q-learning by approximating the action-value function $Q(s, a; \theta)$ using a deep neural network, whose behavior is determined by its weights θ, rather than storing Q-values in a table. This makes DQN more suitable for environments with large or continuous state spaces, where maintaining an explicit Q-table is impractical [26, 35].

The DQN trains a neural network to approximate the optimal action-value function $Q^*(s, a)$. The agent's experiences at each time step t are stored as tuples $e_t = (s_t, a_t, r_t, s_{t+1})$ in a replay memory $D_t = \{e_1, \ldots, e_t\}$. During training, Q-learning updates are applied on minibatches sampled uniformly at random from this memory. At each iteration i, the network parameters θ_i are updated by minimizing the loss function:

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]$$

Here, s and s' denote the current and next states, a and a' are the current and next actions, r is the reward, γ is the discount factor, θ_i represents the current network parameters, and θ_i^- denotes the target network parameters used to compute the target Q-values. The target network parameters θ_i^- are updated with the Q-network parameters θ_i every C steps and remain fixed between updates. This approach stabilizes training and improves learning efficiency [35].

5

Experimental implementation

This section presents the details of the experimental implementation of this study. First, the reasons for the change in scope during the research are explained. Next, the reinforcement learning setup is introduced, including the different reward functions. Finally, the parameter optimization process is discussed.

5.1 Change of the scope

Initially, the aim of this study was to combine process mining with reinforcement learning. However, after exploring the dataset and applying process mining, several challenges emerged. First, the dataset described in Section 3.1 is highly complex, with almost every patient following a unique treatment path. In addition, some important information needed for reinforcement learning was missing.

Second, medical knowledge is necessary to interpret treatment decisions. Reinforcement learning can suggest actions based on patterns in the data, but it cannot judge whether a treatment is actually good or harmful for a patient. Understanding what is beneficial depends on factors such as the patient's condition, tumor size, and possible side effects. Without medical guidance, it is difficult to know if the recommended actions are safe or meaningful.

Therefore, it was decided to apply reinforcement learning to optimize cancer treatment independently, without using the results from process mining. Nevertheless, combining process mining with reinforcement learning remains a promising approach. Process mining provides a detailed view of real patient pathways, identifies common pathways and bottlenecks, and can reveal which sequences of treatments lead to better outcomes. If more specific information about the patients were available, integrating these insights with reinforcement learning could help the algorithm make decisions that are not only statistically optimal but also clinically meaningful, improving both safety and effectiveness.

5.2 Chemotherapy dosing optimization

RL is applied to the chemotherapy dosing problem. Figure 5.1 illustrates how reinforcement learning can guide dosing decisions for the four-state tumor model. At each time step t, a dose D_t is administered based on the current state (N_t, T_t, I_t, C_t). The patient's response is captured in the outcome state Q_t, and the agent receives a reward r_t based on treatment effectiveness. This process repeats over time, allowing the agent to learn an optimal dosing policy [24].

The state space is defined by all possible combinations of the four variables in the tumor growth model. For every simulated patient, the initial state is set to (1.0, 0.2, 0.15, 0.0). The action space corresponds to the drug dosage. Dosages range from 0 (no drug) to 1 (maximum dosage). To keep the problem discrete, 20 values within this range are considered.

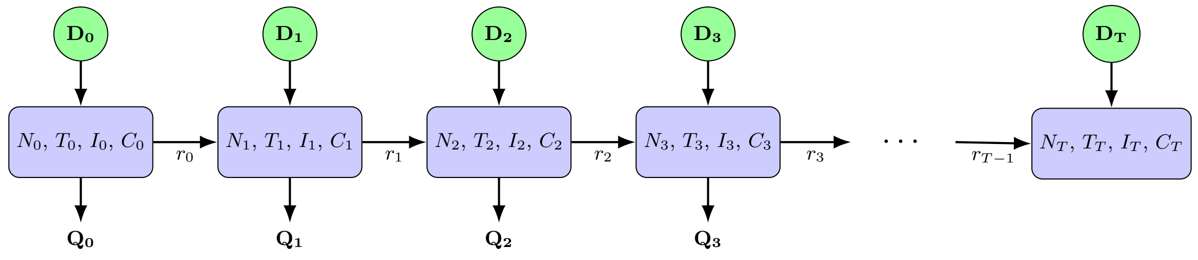

Figure 5.1: Reinforcement learning treatment plan for chemotherapy dosing with the four-state mathematical model.

The resulting finite action space is $\mathcal{A} \in \{0.0, 0.05, 0.11, 0.16, 0.21, 0.26, 0.32, 0.37, 0.42, 0.47, 0.53, 0.58, 0.63, 0.68, 0.74, 0.79, 0.84, 0.90, 0.95, 1.0\}$.

In this study, different reward functions are used to examine their effect on the agent's behavior. The first reward function is expressed in Equation 5.1, originally from [1]. The reward uses an error term $e(t) = x_2(t)$ which represents the number of tumor cells at time t. This reward function evaluates the change in tumor cell count between two consecutive time steps.

$$r_{k+1} = \begin{cases} \frac{e(kT)-e((k+1)T)}{e(kT)}, & e((k+1)T) < e(kT), \\ 0, & e((k+1)T) \geq e(kT), \end{cases} \tag{5.1}$$

The reward is only positive when the number of tumor cells decreases between time steps kT and $(k+1)T$, and is zero otherwise. This encourages actions that lead to a reduction in tumor cells.

The second reward function is given in Equation 5.2, based on [23]. This reward function takes all four states of the mathematical model into account. It encourages the normal and immune cells to increase, and the tumor cells and drug doses to decrease.

$$r_k = N(k) - T(k) + I(k) - w \cdot u(k) \tag{5.2}$$

In this reward function, a parameter w is introduced as a multiplier for the drug infusion rate. This parameter influences the amount of drug the agent administers. A higher value of w increases the penalty for prescribing drugs, thereby discouraging excessive administration. The purpose of including this parameter is to achieve a balance between drug usage and the patient's well-being.

The goal of the treatment is to reduce the number of tumor cells. For the implementation, a goal state of 10^{-4} tumor cells is defined. When the number of tumor cells falls below this threshold, the patient is considered recovered.

To reach this goal, the agent must learn to select appropriate actions during training. For this purpose, an ϵ-greedy policy is applied for action selection. With probability ϵ, a random action is taken, while with probability $1 - \epsilon$, the action suggested by the agent is chosen. This approach ensures a balance between exploration and exploitation during learning [26].

5.2.1 Parameter selection

To select appropriate values for the parameters, different values are analyzed and the best are selected in a grid search. Because the goal is to reduce the number of tumor cells completely, the number of days it takes to reduce the number of tumor cells below the goal state ($T < 10^{-4}$) is used for selecting parameters. This is based on a study by Mashayekhi et al. [26], where they took the average number of steps it took to reach the goal state in their actor-critic approach.

The grid search is performed using reward function 2 for training the agent. This choice is made because the action value w can be included, and initial exploration showed that parameter variations have a stronger impact on agents trained with reward function 2.

Q-learning grid search A grid search for Q-learning is conducted to optimize the reinforcement learning agent's parameters. The search space included four learning rates ($\alpha \in \{0.1, 0.2, 0.3, 0.5\}$), three discount factors ($\gamma \in \{0.7, 0.8, 0.95\}$), one exploration rate ($\epsilon = 0.2$), and three action value multipliers ($w \in \{1.3, 1.5, 1.7\}$), resulting in 36 unique combinations. Lower values for the learning rate and higher values for the discount factor are selected, as that focuses on the long-term results [32].

For each combination the recovery time, total drug use, and last day of drug administered is stored. The combinations are evaluated on the recovery time. Table 5.1 presents the best four performing combinations based on the recovery time.

α	γ	w	Recovery Time	Total Drugs	Last Drug Day
0.2	0.95	1.7	**78**	16.105	43
0.3	0.95	1.7	82	13.053	43
0.5	0.95	1.7	83	15.053	43
0.1	0.95	1.7	90	10.474	42

Table 5.1: Top 4 parameter combinations for Q-learning (using reward function 2) with the lowest recovery time.

The best-performing combination of parameters is found at $\alpha = 0.2$, $\gamma = 0.95$, $\epsilon = 0.2$, and $w = 1.7$, achieving a minimum recovery time of 78 days. In this setting the drug use is also low. Generally, lower values of α such as 0.1 lead to slower or unstable learning, while $\alpha = 0.2$ and 0.3 offer better and more consistent results. Higher discount factors, particularly $\gamma = 0.95$, significantly improve performance, indicating that long-term reward planning is crucial. An action value of $w = 1.7$ consistently leads to faster recovery and reduced drug administration compared to lower values. All results of the grid search for Q-learning can be found in the appendix, specifically in Table 8.1.

DQN grid search A grid search for the DQN algorithm is also conducted to obtain the best parameters and minimize recovery time. This grid search includes 96 parameter combinations. The DQN algorithm introduces additional parameters compared to Q-learning. In particular, it uses a batch size and a neural network learning rate. Also an epsilon decay rate is included. While in Q-learning ϵ was fixed at 0.2, in DQN it is set to 1 at the start of training and is gradually decreased each episode according to ϵ_{decay}. The memory size is also a parameter and is set to 10,000, and after every 10 episodes the Q-network parameters are updated. The parameters explored in the grid search are: discount factor $\gamma \in \{0.95, 0.99\}$, epsilon decay $\epsilon_{decay} \in \{0.99, 0.995\}$, batch size $\in \{32, 64\}$, learning rate $\in \{0.001, 0.0005\}$, episodes $\in \{200, 500, 1000\}$, and action value $w \in \{1.0, 1.3\}$.

γ	ϵ_{decay}	Batch Size	Learning Rate	Episodes	w	Recovery Time	Total Drugs	Last Day
0.95	0.995	32	0.0005	500	1.0	**70**	8.42	70
0.95	0.995	64	0.0005	200	1.0	76	13.47	75
0.99	0.990	32	0.001	200	1.0	71	25.26	71
0.99	0.990	32	0.0005	200	1.0	71	32.42	71

Table 5.2: Top 4 parameter combinations for DQN (using reward function 2) with the lowest recovery time.

The experiments show that the best results are achieved with $\gamma = 0.95$, $\epsilon_{\text{decay}} = 0.995$, a batch size of 32, a learning rate of 0.0005, 500 training episodes, and an action value of $w = 1.0$. This combination leads to a recovery time of 70 days. Overall, a lower learning rate and smaller batch sizes improve performance. In addition, a slower epsilon decay of 0.995 gives better exploration during training. Finally, lower action values consistently outperformed higher values for w. In Tables 8.2–8.4 in the appendix, all the result of the grid search for DQN can be found.

6

Results

In this chapter, the results are presented and discussed. The analysis begins with the outcomes of process mining, followed by the results of the reinforcement learning approaches. Q-learning and DQN are applied to optimize chemotherapy dosing, with a focus on the effect of the different reward functions.

6.1 Process mining

The heuristic miner algorithm is applied to the event log dataset from the hospital. Process mining is applied on department level and for one specific department, the radiotherapy department. The application of process mining to the radiotherapy department provides insights into the treatment journey of patients in this department.

Department level. Applying process mining at the department level gave a general view of patient movements between departments. There is filtered on one diagnosis to make the structure less complex. Even though filtering, the process in the hospital is very complex. The heuristic miner algorithm was used with adjusted threshold values (shown in Table 6.1) to reduce noise and focus on the main structure of the process.

The resulting process model, as visualized in Figure 6.1, reveals a complex but structured flow of interactions between various hospital departments. Important departments like the *Nursing ward*, *General Lab Clinical Chemistry*, and the *Obstetrics & Gynaecology clinic* stand out as central points in the process. The Nursing ward appears to be the most central and highly connected node, suggesting it plays a crucial role in the patient flow. Similarly, the Obstetrics & Gynaecology clinic and General Lab Clinical Chemistry are very active and closely connected to other departments, indicating their significant involvement in clinical processes. There are also some cycles visible in the process. These cycles may indicate that patients undergo multiple rounds of diagnostics or treatment.

Parameter	**Default value**	**Value used at department level**	**Value used in radiotherapy department**
Dependency threshold	0.5	0.6	0.5
Positive Observations threshold	1	5	5
Length-two loops threshold	0.5	0.7	0.5

Table 6.1: Parameters of the heuristic miner algorithm.

Radiotherapy. In addition to the department level analysis, process mining was also applied specifically to the radiotherapy department. Table 6.1 also provides the parameter values used in this model. This made it easier to focus on the treatment steps within one department, resulting in a less complex process model.

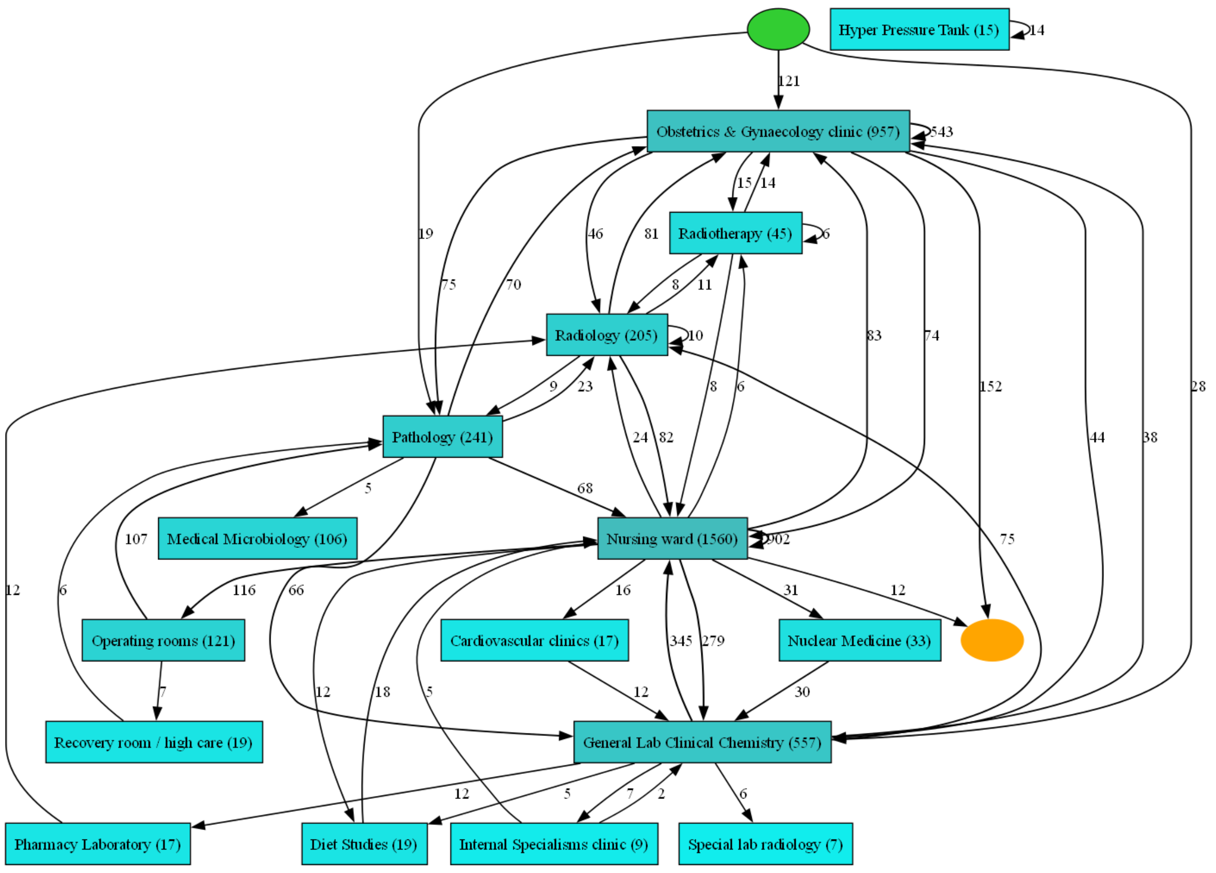

Figure 6.1: Process model at department level.

The process model in Figure 6.2 illustrates the flow of patients through various stages of consultation and treatment. The bottom part of the graph focuses on the main treatment trajectory, beginning with *Behandeltijd* (treatment time), which acts as a central point. From there, patients typically proceed to *teletherapy* the most common treatment pathway, or alternatively to *brachytherapy* or the less frequent *hyperthermia*. Teletherapy also serves as a transitional step, directing some patients further to brachytherapy or hyperthermia, while others complete their treatment journey at this stage. Both hyperthermia and brachytherapy eventually lead to the terminal node, representing the end of treatment.

The process mining analysis shows that the hospital workflow is complex, especially at the department level. Even when filtering for a single diagnosis, many different patient paths are visible. Some departments, like the *Nursing ward* and *General Lab Clinical Chemistry*, play a critical role and are involved in many patient journeys. In contrast, the radiotherapy department shows a clearer and more structured treatment pathway. Most patients follow a main route, with options for teletherapy, brachytherapy, or hyperthermia. This provides a clearer view of the treatment flow and helps identify areas that may benefit from further analysis or optimization.

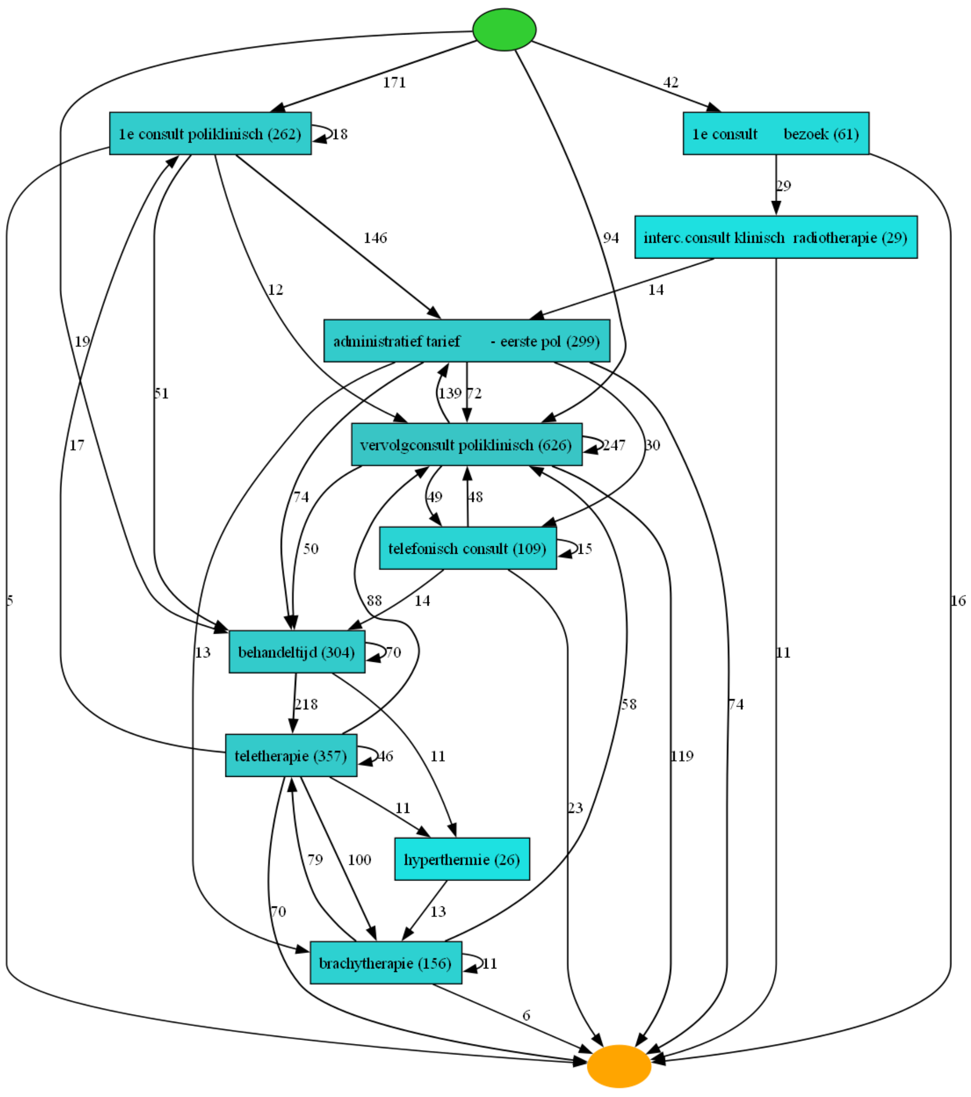

Figure 6.2: Process model of the radiotherapy department.

6.2 Reinforcement learning

In this section, the results obtained using reinforcement learning methods are presented. The performance of the Q-learning algorithm is reported first, followed by the results from DQN. A comparison is then provided between the two methods as well as the two reward functions used to train the agent.

6.2.1 Q-learning results

Q-learning is applied to investigate how chemotherapy drug dosing strategies can be optimized, with two agents trained for 50,000 episodes using different reward functions. The parameters are obtained using the grid search. In Figures 6.3 and 6.4 the response of a patient to the drug schedule trained by reward function 1 and 2, respectively, is visualized. It illustrates how the normal cells (N), tumor cells (T), and immune cells (I) respond to the drug doses over time. In the upper plots the dynamics of the cells can be seen, and in the lower plot the recommended drug dosage by the agent (the policy) for every time step t.

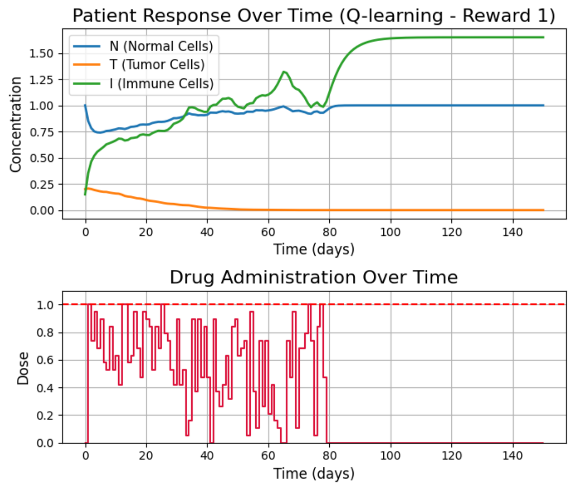

Figure 6.3: Simulated patient response to drug administration learned by a Q-learning agent using reward function 1.

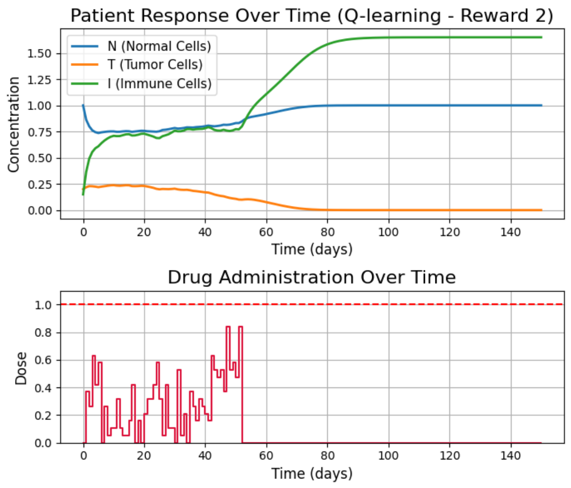

Figure 6.4: Simulated patient response to drug administration learned by a Q-learning agent using reward function 2.

The policy learned with reward function 1 uses a fluctuating dosing schedule with generally high drug doses. In contrast, reward function 2 uses fewer drugs and stops treatment earlier, even though the number of tumor cells has not reached the goal state (10^{-4}). However, because of the dynamics of the tumor growth model, the immune cells are at a level where they can still reduce the tumor cells completely on their own.

Table 6.2 summarizes the numerical results. Reward function 1 leads to recovery in 78 days but requires a substantially higher drug dosage. Reward function 2 extends recovery time to 89 days, but with a much lower total dosage.

Metric	**Reward Function 1**	**Reward Function 2**
Total drugs administered	46.63	15.21
Recovery time (days)	78	89
Last drug administration day	78	51

Table 6.2: Performance of Q-learning agents trained with two different reward functions.

Overall, the Q-learning results show a trade-off between treatment intensity and recovery time. Reward function 1 focuses on fast tumor reduction but uses more drugs, while reward function 2 gives a more balanced approach, using fewer drugs while still achieving recovery.

6.2.2 DQN results

Two DQN agents are trained with the two different reward functions using the parameters obtained from the grid search. Figures 6.5 and 6.6 show the patient responses to the drug schedule trained by the two reward functions. Similar to Q-learning, reward function 1 leads to fast tumor reduction, while reward function 2 preserves immune cells better. Unlike Q-learning, DQN policies are more stable and produce consistent outcomes without large fluctuations in drug dosing.

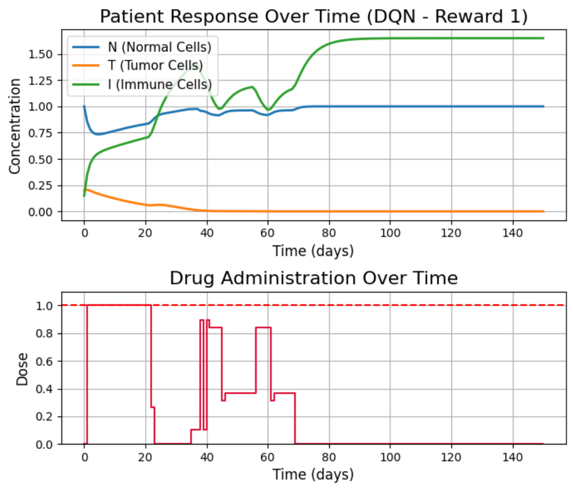

Figure 6.5: Simulated patient response to drug administration learned by a DQN agent using reward function 1.

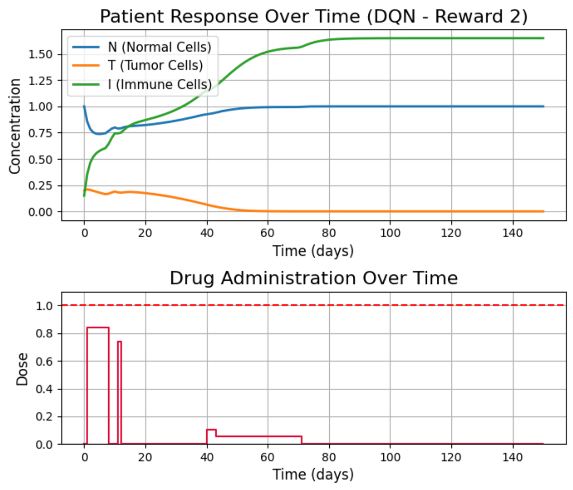

Figure 6.6: Simulated patient response to drug administration learned by a DQN agent using reward function 2.

Table 6.3 presents the numerical results. Reward function 1 achieves recovery in 68 days but requires a higher total drug dosage. Reward function 2 takes 70 days to reach recovery, using less drugs overall.

Metric	**Reward Function 1**	**Reward Function 2**
Total drugs administered	37.95	8.42
Recovery time (days)	68	70
Last drug administration day	68	70

Table 6.3: Performance of DQN agents trained with two different reward functions.

In summary, the DQN results confirm the same trade-off seen in Q-learning. Faster recovery requires more drugs given to the patient, while a slower approach can also succeed in reducing the tumor.

6.2.3 Comparison

Comparing the results of the different reward functions shows that the choice of reward strongly affects how the RL agent behaves. Reward function 1 gives more drugs and focuses on reducing the tumor quickly. Reward function 2 considers other aspects of the tumor growth model and encourages normal and immune cells to stay healthy, which leads to less drug use. As a result, reward function 1 achieves faster tumor reduction but at the cost of higher drug exposure, while reward function 2 takes longer to reach recovery but may reduce side effects. Lower drug doses may be preferable for patients because they reduce toxicity and harmful side effects, while still being able to reduce the tumor effectively.

For Q-learning and DQN, reward function 2 uses different action values: $w = 1.7$ for Q-learning and $w = 1.0$ for DQN. Even though Q-learning is penalized more for giving drugs, the Q-learning agent still administers more medication than the DQN agent. In addition, Q-learning policies tend to fluctuate more in their drug dosing, while DQN policies produce smoother and more stable treatment schedules.

The results from the trained Q-learning and DQN agents also differ. This is is due to the way each method represents the learned policy and how it learns. In Q-learning, the policy is stored as a Q-table, where for every state–action pair encountered during training the corresponding Q-value is recorded. The best action for any given state is then determined by selecting the action with the highest Q-value in the table. DQN, on the other hand, represents the policy using a neural network that approximates the Q-function. Instead of storing individual Q-values for each state–action pair, the network generalizes across states, allowing it to estimate Q-values for previously unseen states. This makes the DQN approach more robust, particularly in environments with large or continuous state spaces, because it can generalize and make informed decisions even for states that were not explicitly visited during training.

7 Conclusion

In this study, the original idea was to combine process mining with reinforcement learning for optimizing treatment pathways. Along the way it became clear that this approach wasn't feasible within the current scope due some limitations, so both methods were applied separately. Even though, the methods still gave valuable insights.

Process mining revealed how treatment pathways actually work in practice for the gynecology department of a Dutch hospital. It showed how patients flow through various departments during their treatment period, and for the radiotherapy department the flow of specific treatments and their sequence are identified. These insights are valuable because they show the complexity of workflows in hospitals.

At the department level, the process model showed central departments like the *Nursing ward*, *General Lab Clinical Chemistry*, and the *Obstetrics & Gynecology clinic*, highlighting their important roles in patient flow. The analysis also showed cycles indicating repeated rounds of diagnostics or treatments. For the radiotherapy department, the analysis provided a clearer view of patient care, with a structured treatment trajectory centered around *Behandeltijd*, and typical pathways through *teletherapy*, *brachytherapy*, or *hyperthermia*.

Reinforcement learning was then used to optimize chemotherapy dosing for cancer patients. Two different reward functions were tested to see how they affect the agent's behavior. The first reward function focused only on reducing tumor cells as fast as possible. Only the number of tumor cells is included in this reward function. The higher the reduction in tumor cells, the higher the reward. The second reward function also considered the immune cell and normal cell levels of the patient and the drug infusion rate. Which makes this function more focused on the patient's overall well-being, not just tumor elimination.

The choice of reward function had a big impact on the treatment strategies. Reward function 1 used high drug doses in the policy, achieving fast tumor reduction but potentially causing more toxicity for patients. Reward function 2 produced more balanced treatment schedules that used fewer drugs and preserved immune cells better. This approach took longer to reduce the tumor but is for patients a safer option, as is it reduces side effects, while still being able to eliminate the tumor completely.

Comparing Q-learning and Deep Q-Networks showed that the algorithm also matters. Q-learning uses a Q-table which limits its ability to handle complex situations. Deep Q-Networks can generalize better and consistently outperformed Q-learning in the experiments. The Deep Q-Networks agent with reward function 2 achieved the best results. It successfully reduced the tumor while using the least amount of drugs, making it the most suitable approach for patient care.

These findings show that both the reward function design and the choice of algorithm are important in medical RL problems. Even small changes can lead to very different treatment recommendations, and consequently, different patient outcomes. In practice, this study suggests that reinforcement learning could be a valuable tool to support oncologists in personalizing treatment strategies. If this approach were combined with process mining, efficiency and patient safety in cancer care could be improved, as real-world treatment data would then be incorporated into the decision-making process.

8 Discussion

While the results of this study provide useful insights, several limitations and considerations should be highlighted. First, reinforcement learning models are naturally sensitive to randomness. Factors like weight initialization, exploration strategies, and random changes in the environment can all affect training outcomes. This means the agents' performance may vary between runs, and reported results should be seen as indicative rather than definitive.

Another limitation is the absence of medical input. The models here are purely data-driven and have not been validated by medical experts. While the simulated environments approximate patient responses, real-world treatment decisions require careful judgment, patient preferences, and other factors. Including medical expertise would make the models more realistic and safer to use in practice.

The initial idea of combining process mining with reinforcement learning is still promising, even though it was not possible in this study. Process mining could provide a clear picture of actual patient flows, which could guide the design of RL environments, limit action choices, or suggest policies. Combining these methods in future work could improve both understanding and applicability of RL treatment strategies.

Parameter tuning is another area to consider. Here, a grid search was used to select DQN parameters, but this method may not find the best possible settings. More advanced techniques could produce stronger and more reliable results.

The mathematical model for simulating patients for RL is a simplified version of the real-world. It cannot fully represent the biological differences between patients, possible side effects of treatment, or toxicity. Future work could include more realistic patient models to better reflect these complexities.

The choice of reward functions also limits the results. Only two reward functions were tested, focusing mainly on tumor reduction and a simple measure of patient well-being. Real clinical priorities are more complex, including toxicity, quality of life, and long-term outcomes. Expanding the reward functions with multiple objectives, for example by adding or adjusting weights in reward function 2, could give more realistic and clinically useful strategies.

Looking forward, several directions could improve this research. Developing more complex, clinically-informed reward functions, exploring other RL algorithms, modeling patient responses more realistically, and including medical experts in design and validation are all promising. Together with process mining, these improvements could lead to safer, more effective, and personalized treatment strategies.

References

[1] Regina Padmanabhan, Nader Meskin, and Wassim M Haddad. Reinforcement learning-based control of drug dosing for cancer chemotherapy treatment. *Mathematical biosciences*, 293:11–20, 2017. iii, iv, 5, 7, 9, 15

[2] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Analysis of patient treatment procedures. In *Business Process Management Workshops (1)*, volume 99, pages 165–166, 2011. iv, 8

[3] Foluke Ekundayo. Reinforcement learning in treatment pathway optimization: A case study in oncology. *International Journal of Science and Research Archive*, 13(02):2187–2205, 2024. 1

[4] Freddie Bray, Mathieu Laversanne, Hyuna Sung, Jacques Ferlay, Rebecca L Siegel, Isabelle Soerjomataram, and Ahmedin Jemal. Global cancer statistics 2022: Globocan estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: a cancer journal for clinicians*, 74(3): 229–263, 2024. 1

[5] Jennifer L Ersek, Eric Nadler, Janet Freeman-Daily, Samir Mazharuddin, and Edward S Kim. Clinical pathways and the patient perspective in the pursuit of value-based oncology care. *American Society of Clinical Oncology Educational Book*, 37:597–606, 2017. 1

[6] A. Mariotto, K. Yabroff, Yongwu Shao, E. Feuer, and Martin L. Brown. Projections of the cost of cancer care in the united states: 2010-2020. *Journal of the National Cancer Institute*, 103 2:117–28, 2011. doi: 10.1093/jnci/djq495. 1

[7] N. Meropol, D. Schrag, Thomas J. Smith, T. Mulvey, R. Langdon, D. Blum, P. Ubel, and L. Schnipper. American society of clinical oncology guidance statement: the cost of cancer care. *Journal of clinical oncology : official journal of the American Society of Clinical Oncology*, 27 23:3868–74, 2009. doi: 10.1200/JCO.2009.23.1183. 1

[8] Fei Jiang, Yong Jiang, Hui Zhi, Yi Dong, Hao Li, Sufeng Ma, Yilong Wang, Qiang Dong, Haipeng Shen, and Yongjun Wang. Artificial intelligence in healthcare: past, present and future. *Stroke and vascular neurology*, 2(4), 2017. 1

[9] Silvana Secinaro, Davide Calandra, Aurelio Secinaro, Vivek Muthurangu, and Paolo Biancone. The role of artificial intelligence in healthcare: a structured literature review. *BMC medical informatics and decision making*, 21:1–23, 2021. 1

[10] Robert K Mahar, Myra B McGuinness, Bibhas Chakraborty, John B Carlin, Maarten J IJzerman, and Julie A Simpson. A scoping review of studies using observational data to optimise dynamic treatment regimens. *BMC medical research methodology*, 21:1–13, 2021. 1

[11] Eric Rojas, Jorge Munoz-Gama, Marcos Sepúlveda, and Daniel Capurro. Process mining in healthcare: A literature review. *Journal of biomedical informatics*, 61:224–236, 2016. 2, 4

[12] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021. 2, 7

[13] Wil Van Der Aalst. Process mining. *Communications of the ACM*, 55(8):76–83, 2012. 3

[14] Jorge Munoz-Gama, Niels Martin, Carlos Fernandez-Llatas, Owen A Johnson, Marcos Sepúlveda, Emmanuel Helm, Victor Galvez-Yanjari, Eric Rojas, Antonio Martinez-Millana, Davide Aloini, et al. Process mining for healthcare: Characteristics and challenges. *Journal of Biomedical Informatics*, 127:103994, 2022. 3, 4

[15] Wil Van Der Aalst. Process mining: Overview and opportunities. *ACM Transactions on Management Information Systems (TMIS)*, 3(2):1–17, 2012. 3

[16] Fazla Rabbi, Debapriya Banik, Niamat Ullah Ibne Hossain, and Alexandr Sokolov. Using process mining algorithms for process improvement in healthcare. *Healthcare Analytics*, 5:100305, 2024. 4

[17] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, Cambridge, 1998. URL `https://muse.jhu.edu/book/60836`. 5, 6, 13

[18] Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012. 5, 6

[19] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 5

[20] Evan M Russek, Ida Momennejad, Matthew M Botvinick, Samuel J Gershman, and Nathaniel D Daw. Predictive representations can link model-based reinforcement learning to model-free mechanisms. *PLoS computational biology*, 13(9):e1005768, 2017. 5

[21] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999. 6

[22] Richard S Sutton, Satinder Singh, and David McAllester. Comparing policy-gradient algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 30(4):467–477, 2000. 6

[23] Inkyung Ahn and Jooyoung Park. Drug scheduling of cancer chemotherapy based on natural actor-critic approach. *BioSystems*, 106(2-3):121–129, 2011. 6, 9, 15

[24] Chan-Yun Yang, Chamani Shiranthika, Chung-Yih Wang, Kuo-Wei Chen, and Sagara Sumathipala. Reinforcement learning strategies in cancer chemotherapy treatments: A review. *Computer Methods and Programs in Biomedicine*, 229:107280, 2023. 7, 14

[25] Yufan Zhao, Michael R Kosorok, and Donglin Zeng. Reinforcement learning design for cancer clinical trials. *Statistics in medicine*, 28(26):3294–3315, 2009. 7

[26] Hoda Mashayekhi, Mostafa Nazari, Fatemeh Jafarinejad, and Nader Meskin. Deep reinforcement learning-based control of chemo-drug dose in cancer treatment. *Computer Methods and Programs in Biomedicine*, 243:107884, 2024. 7, 13, 15

[27] Boudewijn van Dongen. Real-life event logs - hospital log, 2011. URL `https://data.4tu.nl/articles/dataset/Real-life_event_logs_-_Hospital_log/12716513/1`. 8

[28] L.G De Pillis and A Radunskaya. The dynamics of an optimally controlled tumor model: A case study. *Mathematical and Computer Modelling*, 37(11):1221–1244, 2003. ISSN 0895-7177. doi: https://doi.org/10.1016/S0895-7177(03)00133-X. 9

[29] Anton JMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristicsminer algorithm. 2006. 11, 12

[30] AJMM Weijters and Joel Tiago S Ribeiro. Flexible heuristics miner (fhm). In *2011 IEEE symposium on computational intelligence and data mining (CIDM)*, pages 310–317. IEEE, 2011. 12

[31] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992. 12, 13

[32] Regina Padmanabhan, Nader Meskin, and Wassim M Haddad. Closed-loop control of anesthesia and mean arterial pressure using reinforcement learning. *Biomedical Signal Processing and Control*, 22:54–64, 2015. 13, 16

[33] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. 13

[34] Qian Zhou, Yang Lian, Jiayang Wu, Mengyue Zhu, Haiyong Wang, and Jinli Cao. An optimized q-learning algorithm for mobile robot local path planning. *Knowledge-Based Systems*, 286:111400, 2024. 13

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. 13

Appendix

α	γ	ε	w	Recover Time	Last Drug Day	Total Drugs
0.1	0.7	0.2	1.3	107	110	45.947
0.1	0.7	0.2	1.5	107	36	8.947
0.1	0.7	0.2	1.7	89	47	15.158
0.1	0.8	0.2	1.3	99	100	46.000
0.1	0.8	0.2	1.5	111	111	49.105
0.1	0.8	0.2	1.7	∞	22	4.211
0.1	0.95	0.2	1.3	95	99	43.790
0.1	0.95	0.2	1.5	99	99	40.263
0.1	0.95	0.2	1.7	90	42	10.474
0.2	0.7	0.2	1.3	109	109	46.368
0.2	0.7	0.2	1.5	103	33	8.579
0.2	0.7	0.2	1.7	∞	38	7.579
0.2	0.8	0.2	1.3	110	110	45.842
0.2	0.8	0.2	1.5	92	46	14.474
0.2	0.8	0.2	1.7	∞	39	8.789
0.2	0.95	0.2	1.3	99	101	44.579
0.2	0.95	0.2	1.5	98	98	49.789
0.2	0.95	0.2	1.7	**78**	43	16.105
0.3	0.7	0.2	1.3	107	114	45.316
0.3	0.7	0.2	1.5	∞	32	7.158
0.3	0.7	0.2	1.7	120	43	10.105
0.3	0.8	0.2	1.3	106	105	43.947
0.3	0.8	0.2	1.5	101	34	9.474
0.3	0.8	0.2	1.7	99	41	11.105
0.3	0.95	0.2	1.3	92	92	38.842
0.3	0.95	0.2	1.5	113	107	45.737
0.3	0.95	0.2	1.7	82	43	13.053
0.5	0.7	0.2	1.3	109	112	48.263
0.5	0.7	0.2	1.5	107	52	13.474
0.5	0.7	0.2	1.7	117	45	10.000
0.5	0.8	0.2	1.3	107	96	40.211
0.5	0.8	0.2	1.5	94	55	17.368
0.5	0.8	0.2	1.7	∞	37	7.421
0.5	0.95	0.2	1.3	96	96	40.579
0.5	0.95	0.2	1.5	105	96	43.211
0.5	0.95	0.2	1.7	83	43	15.053

Table 8.1: Grid search results of the Q-learning algorithm trained with reward function 2.

γ	ϵ_{decay}	Batch Size	Learning Rate	Episodes	w	Recovery Time	Total Drugs	Last Drug Day
0.95	0.99	32	0.001	200	1.0	∞	25.4211	148
0.95	0.99	32	0.001	200	1.3	∞	24.9474	149
0.95	0.99	32	0.0005	200	1.0	∞	28.6316	149
0.95	0.99	32	0.0005	200	1.3	∞	21.0526	150
0.95	0.99	64	0.001	200	1.0	∞	27.6842	149
0.95	0.99	64	0.001	200	1.3	85	11.7895	85
0.95	0.99	64	0.0005	200	1.0	76	13.4737	75
0.95	0.99	64	0.0005	200	1.3	∞	24.7368	150
0.95	0.995	32	0.001	200	1.0	∞	27.3158	148
0.95	0.995	32	0.001	200	1.3	∞	12.4737	150
0.95	0.995	32	0.0005	200	1.0	∞	19.5789	124
0.95	0.995	32	0.0005	200	1.3	122	21.4737	122
0.95	0.995	64	0.001	200	1.0	∞	28.6842	149
0.95	0.995	64	0.001	200	1.3	∞	9.6316	150
0.95	0.995	64	0.0005	200	1.0	89	46.2105	89
0.95	0.995	64	0.0005	200	1.3	∞	24.7368	150
0.99	0.99	32	0.001	200	1.0	71	25.2632	71
0.99	0.99	32	0.001	200	1.3	∞	23.5263	149
0.99	0.99	32	0.0005	200	1.0	71	32.4211	71
0.99	0.99	32	0.0005	200	1.3	94	29.8947	94
0.99	0.99	64	0.001	200	1.0	87	53.2105	87
0.99	0.99	64	0.001	200	1.3	∞	26.0526	150
0.99	0.99	64	0.0005	200	1.0	∞	16.6842	150
0.99	0.99	64	0.0005	200	1.3	87	32.5263	87
0.99	0.995	32	0.001	200	1.0	88	47.0000	88
0.99	0.995	32	0.001	200	1.3	112	26.4737	112
0.99	0.995	32	0.0005	200	1.0	107	18.7368	107
0.99	0.995	32	0.0005	200	1.3	74	31.5789	74
0.99	0.995	64	0.001	200	1.0	82	27.1053	82
0.99	0.995	64	0.001	200	1.3	∞	27.6316	150
0.99	0.995	64	0.0005	200	1.0	75	26.8947	75
0.99	0.995	64	0.0005	200	1.3	100	26.7895	100

Table 8.2: Grid search results of the DQN algorithm for 200 episodes trained with reward function 2.

γ	ϵ_{decay}	Batch Size	Learning Rate	Episodes	w	Recovery Time	Total Drugs	Last Drug Day
0.95	0.99	32	0.001	500	1.0	90	36.9474	90
0.95	0.99	32	0.001	500	1.3	93	9.1579	93
0.95	0.99	32	0.0005	500	1.0	∞	19.7368	150
0.95	0.99	32	0.0005	500	1.3	75	32.4737	75
0.95	0.99	64	0.001	500	1.0	87	23.2632	87
0.95	0.99	64	0.001	500	1.3	100	26.3158	100
0.95	0.99	64	0.0005	500	1.0	80	33.4737	80
0.95	0.99	64	0.0005	500	1.3	84	20.7368	84
0.95	0.995	32	0.001	500	1.0	94	7.6316	93
0.95	0.995	32	0.001	500	1.3	93	15.6316	93
0.95	0.995	32	0.0005	500	1.0	70	8.4211	70
0.95	0.995	32	0.0005	500	1.3	101	39.2105	101
0.95	0.995	64	0.001	500	1.0	84	15.4737	84
0.95	0.995	64	0.001	500	1.3	83	17.5263	83
0.95	0.995	64	0.0005	500	1.0	118	23.5263	118
0.95	0.995	64	0.0005	500	1.3	92	16.1579	92
0.99	0.99	32	0.001	500	1.0	93	30.8421	93
0.99	0.99	32	0.001	500	1.3	85	43.8421	85
0.99	0.99	32	0.0005	500	1.0	79	35.3684	79
0.99	0.99	32	0.0005	500	1.3	80	38.3158	80
0.99	0.99	64	0.001	500	1.0	111	33.6316	111
0.99	0.99	64	0.001	500	1.3	85	19.1053	85
0.99	0.99	64	0.0005	500	1.0	∞	16.6842	150
0.99	0.995	32	0.001	500	1.0	90	45.2631	90
0.99	0.995	32	0.001	500	1.3	85	42.2632	85
0.99	0.995	32	0.0005	500	1.0	89	56.5263	89
0.99	0.995	32	0.0005	500	1.3	80	19.8421	80
0.99	0.995	64	0.001	500	1.0	96	39.7895	96
0.99	0.995	64	0.001	500	1.3	87	22.4211	87
0.99	0.995	64	0.0005	500	1.0	84	39.7895	84
0.99	0.995	64	0.0005	500	1.3	108	54.0000	108

Table 8.3: Grid search results of the DQN algorithm for 500 episodes trained with reward function 2.

γ	ϵ_{decay}	Batch Size	Learning Rate	Episodes	w	Recovery Time	Last Drug Day	Total Drugs
0.99	0.99	32	0.0010	1000	1.0	∞	149	15.105
0.99	0.99	32	0.0010	1000	1.3	∞	149	10.421
0.99	0.99	32	0.0005	1000	1.0	∞	150	15.158
0.99	0.99	32	0.0005	1000	1.3	∞	150	18.263
0.99	0.99	64	0.0010	1000	1.0	142	142	45.684
0.99	0.99	64	0.0010	1000	1.3	95	95	27.579
0.99	0.99	64	0.0005	1000	1.0	∞	150	11.053
0.99	0.99	64	0.0005	1000	1.3	∞	150	13.579
0.99	0.995	32	0.0010	1000	1.0	123	123	63.158
0.99	0.995	32	0.0010	1000	1.3	135	135	51.895
0.99	0.995	32	0.0005	1000	1.0	∞	150	23.421
0.99	0.995	32	0.0005	1000	1.3	∞	150	23.684
0.99	0.995	64	0.0010	1000	1.0	127	127	38.737
0.99	0.995	64	0.0010	1000	1.3	100	100	13.895
0.99	0.995	64	0.0005	1000	1.0	∞	147	11.000
0.99	0.995	64	0.0005	1000	1.3	∞	150	13.316
0.95	0.99	32	0.0010	1000	1.0	86	51	6.053
0.95	0.99	32	0.0010	1000	1.3	83	83	7.579
0.95	0.99	32	0.0005	1000	1.0	82	82	20.579
0.95	0.99	32	0.0005	1000	1.3	104	104	7.789
0.95	0.99	64	0.0010	1000	1.0	**73**	73	9.684
0.95	0.99	64	0.0010	1000	1.3	87	87	5.316
0.95	0.99	64	0.0005	1000	1.0	77	77	24.632
0.95	0.99	64	0.0005	1000	1.3	∞	150	22.421
0.95	0.995	32	0.0010	1000	1.0	94	94	9.526
0.95	0.995	32	0.0010	1000	1.3	78	78	20.421
0.95	0.995	32	0.0005	1000	1.0	∞	150	23.053
0.95	0.995	32	0.0005	1000	1.3	89	89	10.632
0.95	0.995	64	0.0010	1000	1.0	76	76	9.737
0.95	0.995	64	0.0010	1000	1.3	79	79	12.368
0.95	0.995	64	0.0005	1000	1.0	105	105	10.579
0.95	0.995	64	0.0005	1000	1.3	108	108	11.947

Table 8.4: Grid search results of the DQN algorithm for 1000 episodes trained with reward function 2.