Lukas Bostelmann-Arp

Multi-objective optimization of cancer therapy using a multi-agent simulation at cellular level

FAKULTÄT FÜR
INFORMATIK

Intelligent Cooperative Systems
Computational Intelligence

Multi-objective optimization of cancer therapy using a multi-agent simulation at cellular level

Master Thesis

Lukas Bostelmann-Arp

September 27, 2021

Supervisor: Prof. Dr. Sanaz Mostaghim

Advisor: Prof. Dr. med. Thomas Tüting

Advisor: Dr. med. Andreas Braun

Lukas Bostelmann-Arp: *Multi-objective optimization of cancer therapy using a multi-agent simulation at cellular level*
Otto-von-Guericke Universität
Intelligent Cooperative Systems
Computational Intelligence
Magdeburg, 2021.

Abstract

Melanoma is a dangerous type of skin cancer whose incidence has increased significantly in recent years. Even though several therapy approaches exist, it is unclear when to use which, as they are based on distinct mechanisms and have different effects on the tumor state. One of those effects is the development of resistance. As every case is different, a method that predicts the tumor progression based on the current state and the administered medications would be beneficial. Individual therapy plans can be constructed which actively steer the tumor progression for each patient using this knowledge. One way of implementing such a tool is by using an in-silico model. Therefore, this work creates an agent-based skin cancer simulation and tests it using an evolutionary algorithm to find optimal therapies. As it is supposed to be a proof of concept, the most straightforward approach is used in most cases. While the medical model needs to be revised, the overall work turned out to be successful. The simulation produces realistic results to some extent, and the optimization algorithm provides plausible therapy plans. Nonetheless, some critical aspects need to be tackled in the future to make the work medically relevant. On the one hand, the implementation needs to be more efficient to allow for more extensive tests and optimization scenarios. On the other hand, a reliable approach to tune the parameters of the simulation is required. Such an approach must feature some search or optimization strategy and must most likely use actual medical data.

Contents

List of Figures

1 Introduction

1.1 Motivation

In medicine, models help to understand intricate interrelationships and thus advance the research of new medications. They do this primarily by reducing the complexity of the individual facts. On the level beneath the original human organism are animal models. Here, primarily rats and mice are used for basic research, finding cures for diseases, research on toxicology, and other tasks. Cell cultures represent even simpler models. Although gained knowledge is not easily transferable back to the human organism, concepts are easier to grasp due to the minimal complexity. However, complicated scenarios are not possible with cell cultures. One solution to this problem may be organs-on-chips [9, 11]. These are microsystems featuring living human cells capable of representing organ physiology. Another alternative, however, may be in-silico models. With the starting point of cell cultures, a model can be implemented in-silico that defines the behavior of individual cells. A simulation then can bring insights into the multicellular interactions and effects of specific influences such as induced drugs. These multi-agent systems feature the advantage that adding or subtracting different simulation elements can create an arbitrary complexity with reasonable costs. Advances in computational power further facilitate the use of computer-aided medicine. Nonetheless, computers have already been in use for some time. In the 1960s, computers were used first for administrative purposes in health care. Additionally, an effort was made to incorporate them into the decision-making process [13]. Since then, computers have been used in a growing number of medical areas. A recent

field is the application of artificial intelligence to medical issues, both virtual and physical [8]. The optimization algorithm used in this work can also be assigned to this topic. Regarding simulations, some multi-agent systems were also already tested and utilized in the medical field: One work used an agent-based computer simulation at a cellular level for a better understanding of the inflammatory processes [1], while another takes a look at different models simulating epithelial tissues [18]. However, an agent does not have to be a single cell or a cluster of cells. One team used a spatiotemporal simulation to model the outbreak of the coronavirus [12]. In their case, an agent refers to a human that interacts with other humans.

When it comes to skin cancer, multiple therapies exist that can be practiced. However, it is currently unclear when it makes sense to use which therapy and in which form. The evolution of the tumor itself can be seen as a Darwinian process, where the tumor evolves based on external influences [21]. Actively steering the tumor progression may be part of future therapies. Some works already incorporate this idea and try to construct models which include the surrounding microenvironments into the equations. One work reviews different systems to characterize tumor development [23]. Another work makes use of dynamic programming to optimize policies in cancer treatment by incorporating evolutionary game theory [7]. Ideally, each patient receives a customized therapy that is optimal regarding the current and future state of the tumor. An in-silico solution to find such therapy is required, as a trial and error approach is not feasible using the patient. It is a promising research field, as tumors cannot anticipate, while humans can [6]. This difference brings an advantage that hopefully leads to the sustainable handling of cancer.

1.2 Research goals

The aim of this work is not to produce any medically relevant results yet. It strives to present a first feasibility study that examines the possibilities in this research area. More precisely, it addresses two questions that build upon each

other. First, is it possible to develop a cell-based simulation that provides meaningful data despite featuring a significantly reduced complexity? Second, can this simulation be used to optimize therapies or help decision-making, for example, when weighing costs and benefits? Not all relevant aspects and approaches can be covered in this work. The most straightforward approach is used in most cases while naming different options that can be examined in follow-up experiments. Furthermore, the computer science aspects are in focus, while the medical ones are presented and used in a simplified manner.

1.3 Thesis structure

Following the introduction, chapter 2 encompasses the background knowledge required for this work. This theoretical part can be split into two sections. The first of which explains the computer science-related fundamentals. Following, the second one introduces the simplified but required medical knowledge to understand the general concepts of the simulation. The main part of this work consists of two stages that try to answer the two questions introduced in the previous section. The first involves modeling and implementing a simplified cancer model. For this, an agent-based system is used, where each agent represents a single cell. The details are shown in chapter 3. In a subsequent chapter, chapter 4, this model will then be validated by using real-world knowledge. Afterward, the second stage involves creating a therapy for the previously implemented simulation and optimizing its parameters. An evolutionary algorithm is used for this that finds the optimal solutions in a multi-objective search space. It does not require any domain-related knowledge besides the modeling details of the therapy. This part is covered in chapter 5. The experimental settings and the results of the optimizations are discussed in chapter 6, together with the question to what extent the tumor development can be directed using different therapies. The final chapter 7 concludes this work and expounds problems regarding the employed approach, as well as presents adaptation and guidelines for future research.

2 Theory

2.1 Evolutionary game theory

Before starting with the evolutionary game theory (EGT) itself, it makes sense to introduce first the general game theory (GT). Game theory studies mathematical models that deal with cooperative and non-cooperative interactions among rational decision-making entities. Such an entity is an agent or player who takes part in some game. GT is often used in social science to understand behavior or economics to help with the decision-making process. The primary aspect of game theory is that the payoff, the win or loss at the end of the game, is influenced by both one's own and others' decisions. As a result, the selected strategy also depends on the strategies pursued by others. The players have the goal to maximize their payoff. The case in which no player can improve the payoff by switching to another strategy is called Nash equilibrium and is often part of the model analysis. The whole issue can be better understood by studying an example, like the prisoner's dilemma. This scenario is a simple interaction often used to introduce the subject, and one can find much literature about it.

In EGT, the game theory is applied in an evolutionary context that represents a sequence of interactions among the participants, with fitness being the fundamental aspect of survival. The focus lies now on populations instead of individual players. In contrast to general GT, a population consists of individuals that do not actively reason about their own and others' strategies. In other words, there is no overall consciousness anymore. As a result, the

player is not the object of interest but the strategies adopted by the individuals. Furthermore, the payoff is now fitness in an evolutionary sense, and the individuals do not choose their strategy but inherit or mutate them. A famous example is the hawk-dove game [19]. Analogous to the Nash equilibrium in GT, evolutionarily stable strategies (ESS) are of great interest. Once adopted by a population in a specific environment, a set of ESS cannot be substituted by a novel set of strategies. However, since EGT plays only a minor role in this work, it will not be discussed in depth. Furthermore, there is already much literature on this topic, like these books [20, 16].

2.2 Evolutionary algorithms

Evolutionary algorithms (EAs) are metaheuristics designed to solve computational optimization problems. A metaheuristic can provide a sufficiently good solution to any optimization problem while making only a few assumptions about the domain. That way, they can easily be applied to a wide variety of problems. Evolutionary algorithms are inspired by the theory of evolution and the concept of survival of the fittest. They can be characterized by three core concepts [22]:

- **Population-based:** A group of *individuals*, each representing a single solution, is called a population. Together, they learn the problem in a parallel manner.
- **Fitness-oriented:** Each *individual* consists of two parts, the gene representation and the resulting *fitness value*. Fitter individuals are favored during the iterative updates of the algorithm leading to convergence.
- **Variation-driven:** *Individuals* are altered using different nature-inspired variation operators like crossover and mutation to explore the solution space.

While the base algorithms are always very similar, different flavors are preferred for different tasks. If only one objective function needs to be optimized,

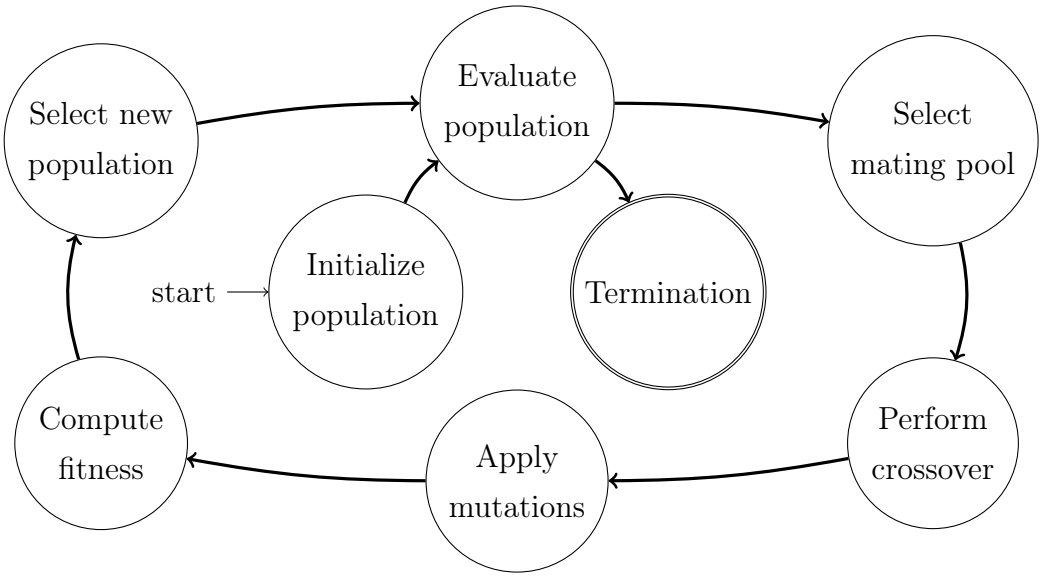


Figure 2.1: Basic EA algorithm

it is called a single-objective optimization problem. A typical example is finding a global minimum in a one-dimensional function. However, depending on the context, multiple objective functions need to be considered. Such a multi-objective optimization scenario is also part of this work and, therefore, will be explained further in subsection 2.2.2. However, first, the general process and the contained parts of an evolutionary algorithm will be introduced in the following part.

2.2.1 Basic algorithm

Evolutionary algorithms take an iterative approach to finding adequate solutions to an optimization problem. Figure 2.1 shows an overview of the general scheme. The algorithm starts by sampling an initial population within the search space. Having a diverse first set of individuals is favorable as it helps to explore the search space faster. In the next step, the population will be evaluated regarding the termination criterion. That can be a predefined number of generations or a check whether it has converged using a performance metric. In that case, the algorithm outputs the locally optimal solution. If the search

continues, a subset of the whole population gets chosen, which is responsible for generating new solutions. Different selection operators can be employed that often focus on the individuals' fitness values. From this mating pool, various offsprings will be created by combining the genes from the respective parents. This process is called crossover, and again different operators exist that depend on the type of genes. Afterward, the second variation operator mutates some of the created offsprings. Next, the newly created individuals' fitness values will be computed. Using these values, the new generation of solutions gets selected. One can pick from the offspring population alone or the union of both sets, the parent generation and the created offspring. With this, the cycle is complete, and it starts all over again.

2.2.2 Multi-objective optimization

A multi-objective optimization scenario features multiple conflicting objective functions. In this context, the objective functions are conflicting when optimizing only one function leads to worse values for the other objectives and the other way around. In other words, there usually is no solution that is the best regarding all objectives, especially not in the final set of solutions. The overall problem can be defined as follows:

Definition 1. *(Multi-objective optimization problem)*

$$\begin{aligned} &\textit{Minimize } \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})\} \\ &\textit{s.t. } \vec{x} \in S \end{aligned}$$

m conflicting objective functions $f_i : S \to R$ need to be minimized by finding solutions $\vec{x}$ from the set of feasible solutions S.

As a consequence, there is no single optimal solution to the problem. Instead, there are various solutions with different advantages but also disadvantages. Nevertheless, a metric is required to distinguish valuable points from those that do not contribute to a helpful compromise regarding the objectives. For this, the so-called Pareto optimality is used:

Definition 2. *(Pareto Optimality)*

$\vec{x^*} \in S$ is (globally) Pareto optimal (PO) if there does not exist another $\vec{x} \in S$ such that $f_i(\vec{x}) \leq f_i(\vec{x^*})$ for all $i = 1, \ldots, m$ and $f_j(\vec{x}) < f_j(\vec{x^*})$ for at least one j.

That means that a solution is not PO if another solution exists that is at least as good as the first one in every objective and strictly better in at least one. In that case, the second solution dominates the first one. The subset of a population containing only Pareto optimal solutions is called Pareto front and should survive to the next generation. There are many algorithms with different methods to select the new population. The algorithm used in this work gets explained in more detail in chapter 5.

Once the multi-objective optimization algorithm has terminated, it returns the Pareto front. However, it cannot say which of these returned solutions is the best one. A decision-making process is still required to analyze the solutions and select the final one. For this task, expert knowledge of the corresponding domain is necessary.

2.3 Medical background

Melanoma is a dangerous kind of skin cancer. The primary cause of an outbreak is exposure to ultraviolet light. Fair-skinned people, in particular, have an increased risk of developing skin cancer in their lifetime. As a result, it most frequently occurs in North America, northern Europe, Australia, and New Zealand. Melanomas originate from melanocytes, which are pigment-producing cells that occur most frequently in the basal layer of the skin. However, the exact stages of development, including the involved biological processes, will not be further elaborated here, as they are not required for understanding this work. While not everything is known in this research field, the basics are well represented in literature [15].

Before looking at the therapy options, it makes sense to explicate the tumor environment used in the simulation. It is not sufficient to examine the tumor cells in isolation, as the surrounding microenvironment significantly affects the tumor development and spread. Apart from tumor cells, immune cells play a significant role. Since the whole immune system is too complex to explain, this section focuses on the two types of cells implemented in the simulation. The first type of immune cell represented in this model, the T cell, is a subgroup of lymphocytes, also called white blood cells. T cells play a significant role in the adaptive immune response. There are various types of T cells, and their birth, maturing, and activation process is complex. Therefore, this work heavily simplifies those aspects. Only one type of T cell is used, which can directly detect and kill cancer cells. The second type of immune cells covered are neutrophils, which are a subgroup of leukocytes. However, they are part of the innate immune system and often first responders in inflammatory environments, including tumor tissues. Even though they are supposed to help the human, it is a current research topic to what extend neutrophils help tumor cells elude other immune cells [17, 5]. That is also the reason why they are included in the simulation.

Apart from tumor and immune cells, the third category represents the surrounding tissue in which the tumor grows. Blood vessels are part of this category, as they are essential due to the delivery of oxygen and nutrients. The second and final type of cell, called fiber in this work, represents the connective tissue. It is again a simplification of the original fibroblasts in the human body. Fibers are implemented as a static mash that can be destroyed by the individual tumor cells to proceed in the environment. Other aspects of tumor fibroblast interactions are not used in this work.

Back to the general topic, this work is focused on tumor therapy and not on tumorigenesis or metastasis. Superficial melanomas that have not yet infiltrated deeply into the tissue can often be excised with a high success rate [3]. However, melanomas tend to spread fast. Under these circumstances, a surgical operation is not sufficient anymore. Different therapy types can be employed

to at least slow down tumor progression. Here, the focus lies on two distinct therapy types, signal transduction therapy and immunotherapy. Signal transduction therapy is a combination of multiple drugs that block different biological pathways which would otherwise promote tumor growth. However, this procedure may fail due to alternative pathways found by the tumor leading to resistance. On the other hand, immunotherapy inhibits mechanisms that help the tumor elude the immune system's attack. As a result, the immune cells can better identify and attack tumor cells. Again, the emerging of resistance is a problem that limits the sustainability of such a therapy.

3 Modeling the simulation

3.1 General concept

The simulation can be divided into two parts: The generic simulation framework and the problem-specific logic. First of which initializes the simulation based on a configuration file and supervises the primary update cycle. Furthermore, it is responsible for in- and output, such as user input, visualization, or saving entire simulations. The framework does not contain any problem-specific knowledge and thus can be used for various simulations from different domains. However, the base entities of such simulations must be agents, which, in this case, represent individual cells. Each of these agents contains several components. These components define the behavioral traits of different agent types. The configuration of each component stack is specified in a user-written configuration file used by the framework to construct a simulation. The end-user can either use predefined components which provide basic behaviors or program and import problem-specific ones. Thereby, each component is derived from an abstract component class. The inheritance allows for modular and reusable code, even across domains. The individual components within a cell communicate via so-called signals, stored in the cell object and accessible for all others. Figure 3.1 depicts a rough overview of the program, which will be further explained in the next section.

As the rest of the general framework is not critical for understanding this work and mainly consists of conventional code, this chapter focuses on the medical model and the concrete implementation of those rules.

Class: Simulation	Class: Cell
MEMBER	**MEMBER**
cells: list[Cell]	**components**: list[Component] **signals**: list[Signal]
METHODS	**METHODS**

update():
  for $cell_i$ in cells:
  for $cell_j$ in cells:
  $cell_i$.interact($cell_j$)
 	
  for cell in cells:
  cell.update()
	interact(cell):
	  for component in components:
	  component.interact(cell)
	
	update():
	  for component in components:
	  component.update()

Figure 3.1: General object oriented concept of the simulation

3.1.1 Interactions

As introduced before, the simulation is written so that the cells and their components contain all problem-specific logic. Therefore, there is no general code that computes interactions and updates the simulation from the outside. This pure agent-based approach is not the most efficient computation-time-wise as it contains a deep hierarchy with many function calls. However, it is easy to expand and adapt due to its modularity. Since the interactions also happen locally within each component, only the affected parts need to be modified instead of an overlooking code when changing, adding, or removing features from the simulation. The job of the simulation framework is only to present each cell to every other possible cell with which it might interact. During this interaction step in the update cycle, neighborhoods can be formed by the components that are helpful for the following update procedure. They are inspired by the neighborhood structure used in swarm intelligence [2]. The

focus primarily lies on the k-closest neighborhood. With this topology, a cell considers the k closest cells as its neighbors. Closeness is thereby meant in a spatial way using the euclidean distance. This approach is justified since cells do not view their surroundings but interact directly with physical contact or perceive messenger substances. Implementation-wise, neighborhoods store references of the corresponding cells and the distances. As a result, they can be used by iterating over them and extracting those two values, as seen in later presented algorithms.

3.1.2 Movement system

The basic movement system is the same one as used in many swarm intelligence applications. The standard motion equations can be seen in Equation 3.1 and Equation 3.2.

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{3.1}$$

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + \vec{f_i} \tag{3.2}$$

In a new update cycle, the velocity gets calculated first. It consists of a momentum term and an acting force. The momentum term is the scaled velocity vector from the previous update cycle. The factor $\vec{f_i}$ can be any force acting on the agent and thus changing its velocity and direction. Given the new velocity vector and the old position, one can calculate the resulting position within the environment. This movement system is straightforward and does not reflect the true nature correctly, as cells usually crawl within the tissue and do not move freely within the space as it is implemented here. Nonetheless, it is sufficient as a base framework for the first test. Further, two main interaction principles are also copied from swarm intelligence methodology. Agents either attract or repel each other. Special functions are applied that precisely determine the behavior by outputting a resulting force given the distance of two cells. The total force for a single agent is then the sum of all individual ones. The simulation in this work makes use of two of those functions that are explained in the following:

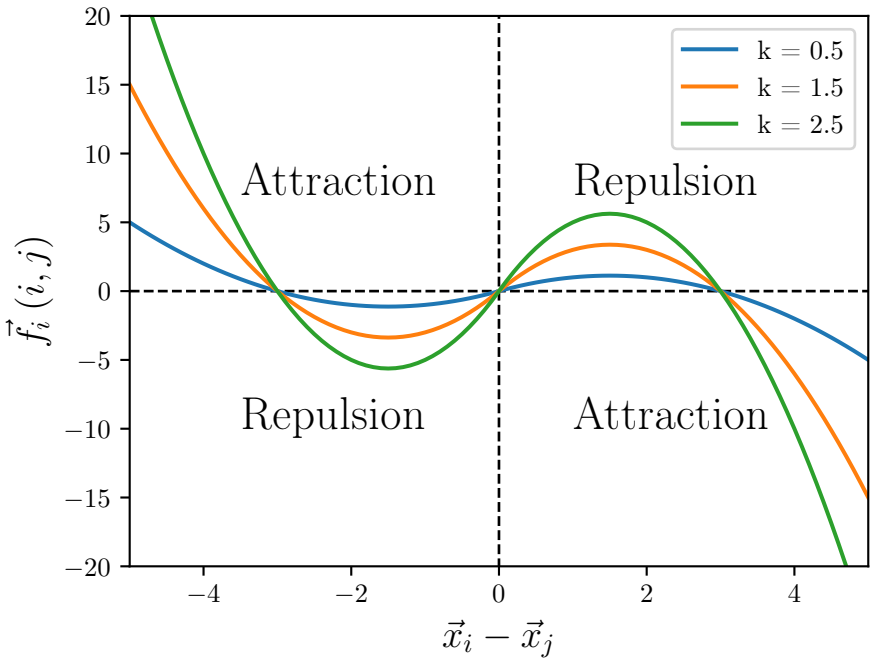


Figure 3.2: Repulsion-I for agent i given agent j with $d = 3.0$

- **Repulsion-I:**

$$\vec{f}_i(i,j) = [-k(\|\vec{x}_i - \vec{x}_j\| - d)](\vec{x}_i - \vec{x}_j), \quad k > 0 \tag{3.3}$$

Two agents with this function try to maintain the distance d between each other. If the current distance is smaller than that, they repel each other. On the other hand, if the current distance is greater than d, they attract each other. One should note that both agents experience the same force. The factor k determines how strongly they attract and repel each other. If k is chosen too large, the force may be too strong, which results in oscillations around the desired distance. On the other hand, if the factor is too small, the effect may vanish, given the own movement of the agents. The user must configure the final factor k from the movement component assigned to each cell type. This function, in general, is frequently used for tracking other agents, which is also the case for this simulation, where T cells can follow tumor cells. Figure 3.2 shows an explanatory plot of the function.

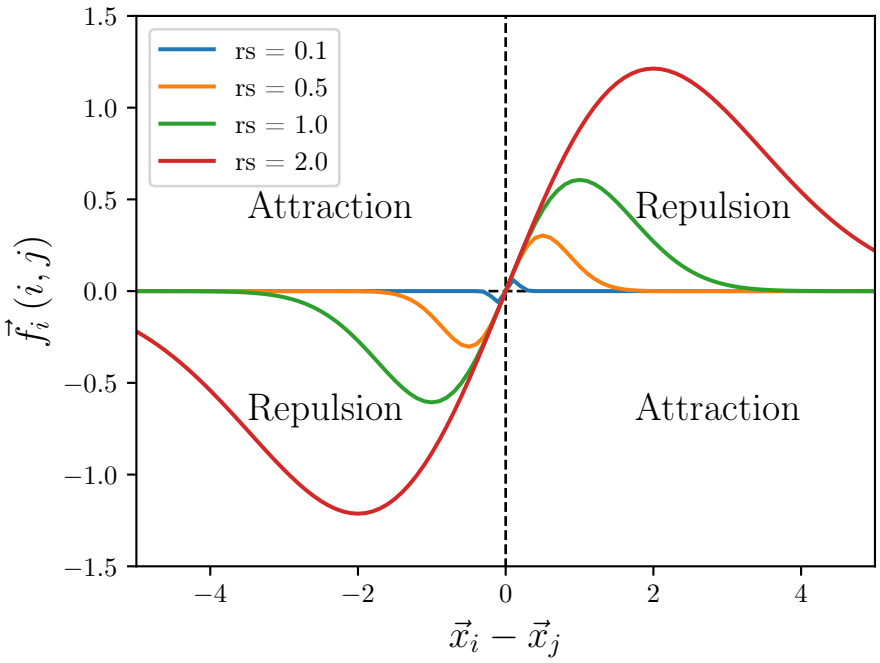


Figure 3.3: Repulsion-II for agent i given agent j with $k_r = 1.0$

- **Repulsion-II:**

$$\vec{f}_i(i,j) = k_r \ \exp\left(\frac{\frac{-1}{2}\|\vec{x}_i - \vec{x}_j\|^2}{r_s^2}\right)(\vec{x}_i - \vec{x}_j), \quad k_r, r_s > 0 \tag{3.4}$$

The second function is, in contrast to the previous one, purely repelling. Any two agents that are too close repel each other. However, there is no possibility to set a fixed minimum distance that two agents should always maintain. How far the repulsion force extends depends on r_s, while k_r acts as an additional magnitude modifier. As the repulsion theoretically extends infinitely, it is hard to quantify the parameters. This work uses a trial and error approach with appropriate visualizations to find suitable values for the two factors. Further, this repulsion function is the base for most interactions within the simulation. It prevents two cells from staying in the exact location, but on the other hand, it is not strict and therefore mimics the deformability of cells. An explanatory plot with $k_r = 1.0$ can be seen in Figure 3.3.

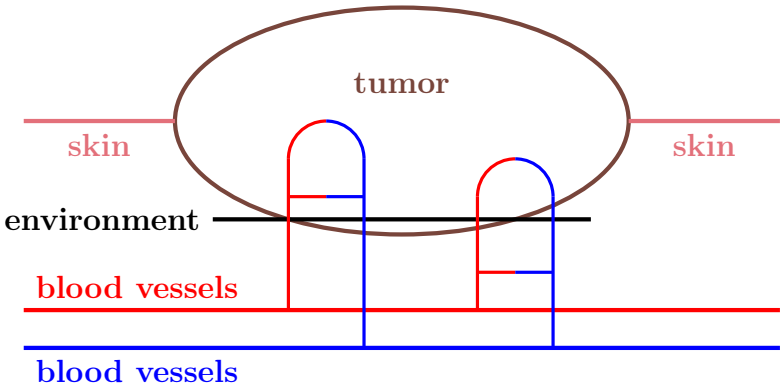


Figure 3.4: Side view of the spatial relations

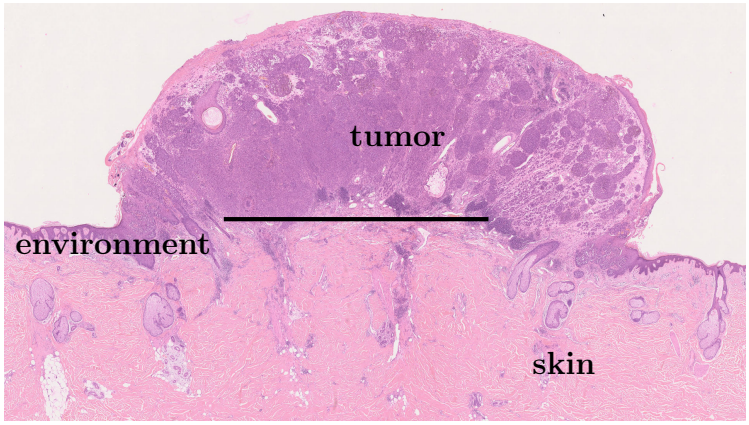


Figure 3.5: Annotated view of a real tumor

3.1.3 Medical viewpoint

The simulation computes the tumor growth in a later stage of its progression but before it has metastasized. Therefore, the emergence is not covered, as this work primarily focuses on the therapy and not prevention. Since it often takes a while for the tumor to be diagnosed, this time frame is suitable for further examination. Regarding the spatial dimension, the visible environment exposes a cross-section of the tissue at the border of the tumor. Therefore, one can see how the tumor invades the healthy tissue and expands in the visible plane. A schematic of the spatial relations is shown in Figure 3.4. Below that, Figure 3.5 shows a real tumor section. Both figures share the same point of view.

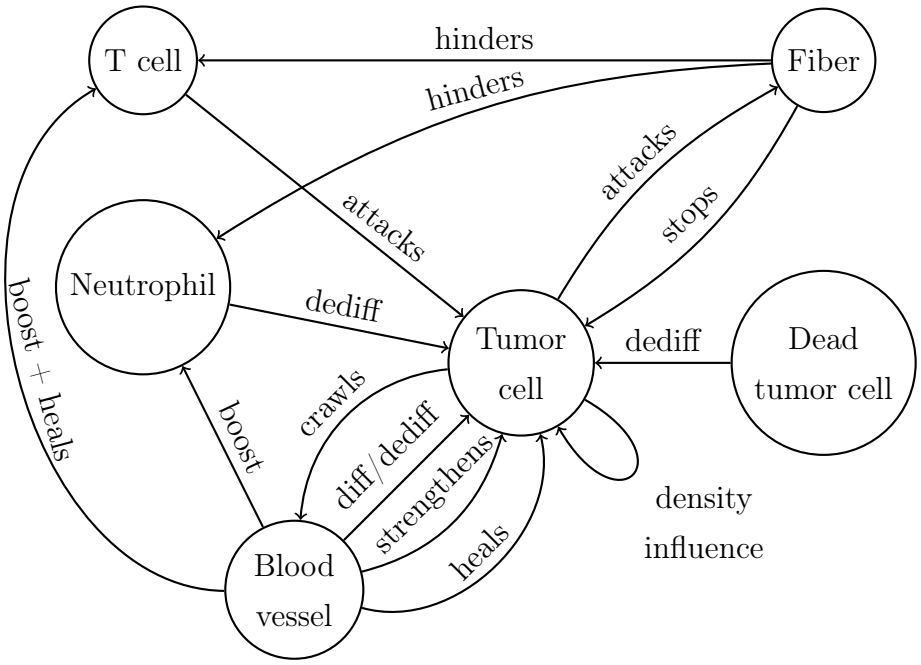


Figure 3.6: Simple interaction diagram

3.2 Cells

This section presents the different cells and their modeling as well as implementation details. Each cell type is uniquely defined by its component stack, which will be illustrated in the following. Apart from the components, all cells share some fundamental traits. They have a precise location and orientation within the environment, as well as a health value. However, the base cell object only stores those values. How these change throughout the simulation depends entirely on the components.

For a better understanding, Figure 3.6 shows a diagram of the general interaction between the different cell types. What the individual interactions mean will be explained in the following. Additionally, Figure 3.7 shows a table with all components broken down by the cell type. One should note that the Attack, Health, Death, and Movement components are different for each cell type. They are all derived classes adapted to the unique individual behavior.

T Cell	Neutrophil	Tumor	Blood Vessel	Fiber
		basic		
Blood Boost	Blood Boost	Blood Boost		
		Blood Neighbor		
		Density Estimator		
Attack		Attack		
Health	Health	Health		
Death	Death	Death		Death
		Differentiation		
Movement	Movement	Movement		

Figure 3.7: Component stacks of all cell types

3.2.1 T cell

In this simulation, the main antagonists of tumor cells are the T cells. They are a simplification of the actual T cells in the human body. All T cells are assumed to be activated and can detect and attack all tumor cells regardless of the differentiation state. Their component stack is described in the following:

Blood boost: This component computes the effect of blood vessels in the vicinity. Each of the sub-elements of a blood vessel has an influence based on its distance. These influences accumulate for each element of the blood vessels and get divided by an upper limit to compute a value between zero and one, the so-called blood boost factor. The general algorithm can be seen in Algorithm 1. Variables with entirely capitalized names have to be configured by the user. Further, the function that maps the distance to the influence can also be configured. An exponential function is used for this simulation because the blood boost should span the environment, but the decrease due to the distance should be significant. A plot for the function, as well as the resulting blood boost distribution within the environment, is shown in Figure 3.8. This

Algorithm 1 Computing blood boost

```
1: procedure COMPUTE_BLOOD_BOOST
2:     blood_vessel_neighborhood ← create_neighborhood()
3:     blood_boost ← 0
4:     for bv, dist in blood_vessel_neighborhood do
5:         tmp ← A + B * exp(−C * dist)
6:         tmp ← max(tmp, 0)
7:         blood_boost ← blood_boost + tmp
8:     end for
9:     blood_boost ← blood_boost/LIMIT
10: end procedure
```

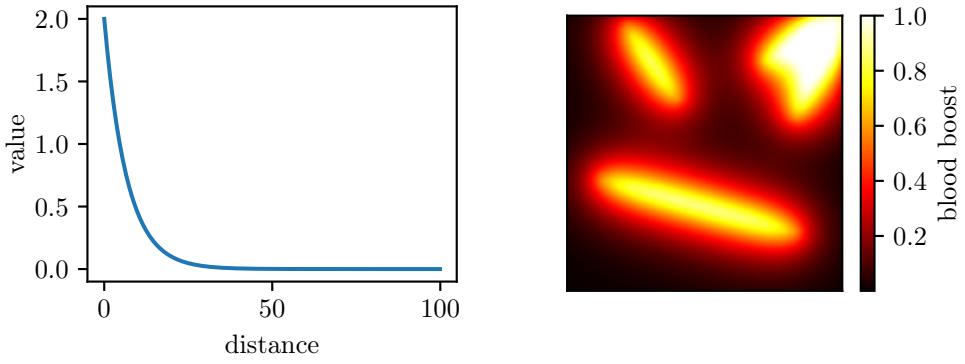


Figure 3.8: Mapping function and resulting blood boost

component does not act itself but provides the calculated value via a signal to the others which use it for their computations.

Attack: The attack component has two primary responsibilities: Finding a target, and if possible, attacking it. When a tumor cell exists within the detection range, it gets selected as the target. Further, if multiple tumor cells are within that range, the closest one becomes the target. Once a T cell has acquired one, it will not switch until the tumor cell is either dead or out of range, and it will provide a reference as a signal for other components to utilize. The T cell will perform multiple attacks in the actual fight when the

Algorithm 2 T cell attack

```
1: procedure T_CELL_ATTACK(delta_time)
2:     if has_target then
3:         health ← get_own_health()
4:         time_to_attack ← time_to_attack − delta_time
5:         while time_to_attack ≤ 0 and health ≥ 0 do
6:             health ← health − HEALTH_DECREASE
7:             time_to_attack ← time_to_attack + 1/ATTACK_SPEED
8:             attack_target()
9:         end while
10:    end if
11: end procedure
```

selected target is within the attack range, which does not necessarily match the detection range. How those attacks are computed is shown in Algorithm 2. Each of those attacks costs a discrete amount of time and health. Further, T cells do not manage their health. They instead attack until they die.

Health: The maximal health value decreases over time to model the life expectancy of T cells. Here, a stochastical reduction is used. Additionally, a T cell can regain health lost due to attacks up to the time-varying maximal value. This positive effect is caused by blood vessels in the vicinity. The actual value is proportional to the blood boost computed in the respective component. An algorithm showing all those computations can be seen in Algorithm 3.

Death: The death of T cells is comparatively simple. T cells die when their health value dwindles below or equal to zero. They get instantly removed from the simulation as they are not needed anymore.

Movement: A T cell has two modes when it comes to its movement. If it has no current target, it crawls around randomly while avoiding regions

Algorithm 3 T cell health

1: **procedure** T_CELL_HEALTH($delta\_time, blood\_boost$)
2: ▷ Regain health due to blood vessels
3: $health \leftarrow health + blood\_boost * FACTOR * delta\_time$
4: ▷ Decrease the maximum possible health over time
5: $max\_health \leftarrow max\_health - \text{random}(0, 2) * DECREASE * delta\_time$
6: $health \leftarrow \min(health, max\_health)$
7: **end procedure**

with denser fibers as well as possible. The latter functionality is implemented using the repulsion-II function introduced in subsection 3.1.2. In addition to the distance, the integrity of the fibers also affects the acting force. If a fiber should not be passed effortlessly, it features a high repulsion. That way, T cells can theoretically go everywhere but tend to find the easiest path. The random movement itself is based on a direction vector that gets altered randomly by adding or subtracting small amounts in each time step. Further, the cells slow down at probabilistic time intervals to allow for sharper direction changes. Additionally, a repulsion factor has the task of preventing cells from being at the same spot. This factor is simply a repulsion-II function acting on all movable other cells within the environment. The second movement mode is active if the T cell has a target. This results in the random movement being deactivated, along with the general repulsion. Instead, it is attracted to the target and follows it to get close and attack. For the attraction function, the repulsion-I function is used, which was described earlier in subsection 3.1.2, with a small value for d that results in physical contact.

Spawning: The final aspect of T cells is their spawning behavior. In the human body, they emerge from blood vessels. In the simulation, that is the first implemented spawning mechanism. Since there are also blood vessels outside the visible environment, the second mechanism spawns T cells at the border,

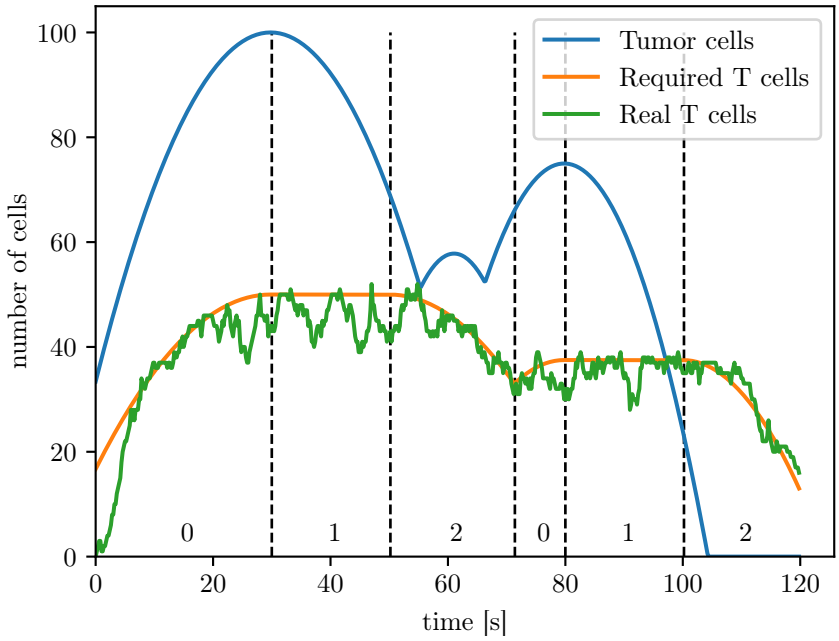


Figure 3.9: T cell spawning with custom tumor cell count and ratio of 0.5

representing those T cells that emerged from other blood vessels that cannot be seen. Further, the spawning rate is an essential factor that significantly influences the outcome of the simulation. It is implemented so that the user can specify the desired ratio between tumor cells and T cells.

Given the number of tumor cells and the desired ratio, the number of required T cells can be calculated. The actual number of T cells then gets adjusted to fit this value by increasing the spawn rate if more T cells are required or decreasing it when too many T cells are alive. However, direct computation prohibits a complete tumor eradication, as fewer T cells are more likely to miss the last isolated tumor cells. The solution for this problem is to introduce a delayed computation once the tumor shrinks. After a plateau phase with a constant value, the required T cells will be calculated with the number of tumor cells from the past. Figure 3.9 shows an example plot, where the number of tumor cells is a custom function. The numbers at the bottom of the plot show the

different stages of the algorithm. State zero is active when the tumor grows. Further, the number of T cells gets computed directly. Once the tumor starts to shrink, the algorithm is in state one, and the number of T cells does not decrease. Instead, it stays constant for a certain period to increase the relative amount of T cells. After the specified time delay, the algorithm switches to the next state, and the required T cells are computed using the past numbers of tumor cells. However, as one can see at second 71, once the number of required T cells at the current time gets higher than the needed T cells due to the delayed value, the tracking is live again.

3.2.2 Neutrophil

The second type of immune cell is the neutrophil. Neutrophils are part of the innate immune system, as introduced in the theory part. They are represented in this simulation as a type of immune cell that promotes the dedifferentiation of tumor cells. That way, they help the tumor elude T cell attacks or at least have an advantage compared to differentiated tumor cells.

The component stack is mainly identical to the one used for T cells. The only difference is that they do not feature the target and attack behavior. As a result, they do not follow tumor cells but only crawl around randomly. In reality, however, they also follow signals. Since this movement style is not relevant to the current medical model, it can be neglected. Further, promoting dedifferentiation in tumor cells is not implemented here but in the tumor cells themselves.

Also, the same spawning system is used for neutrophils as for T cells. The user can adjust the spawning rate for both cell types separately, but the tracking is based on T cells. Therefore, the resulting change in the spawn rate due to the tracking affects both cell types in the same way.

3.2.3 Tumor cell

The tumor cells and their behavior is the central aspect of the simulation. However, the individual cells themselves do not actively make decisions on their own. They only react to their surrounding environment.

The main focus lies on differentiation, as it includes the evolutionary game theory in this simulation. Tumor cells can either be differentiated or dedifferentiated. Both states have different behavioral traits and, therefore, their advantages and disadvantages. The component stack further specifies these characteristics. Depending on their local environment, it might be suitable to be in one state or the other. The conditions to switch between the two states are purely rule-based and modeled on nature.

One aspect of this work is to examine the influence therapies have on the differentiation state and the consequences on tumor growth. At this point, however, the component stack of a tumor cell is presented.

Basic: The basic component is one of the default components and is responsible for multiple values. In this case, it provides the area of the cell given its size and shape. This value is required for computing the local density in another component.

The shape of a tumor cell is assumed to be round. Therefore, the standard formula for a circle is used, as the simulation is fully implemented in only two dimensions.

Blood boost: The blood boost component in the tumor cell is the same as in the T cell. It provides a value based on the accumulated influences of blood vessels in the vicinity of the cell. Once again, it does not act independently but only provides the resulting value to other components, as seen in the following stack description.

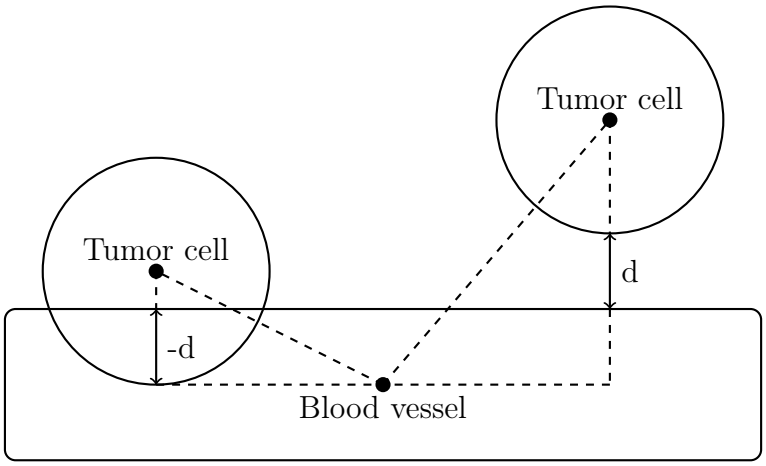


Figure 3.10: Schematic of the blood neighbor computation

Blood neighbor: Sometimes, it makes a difference whether a cell is in the vicinity of a blood vessel or has direct physical contact with one. This component computes the signed distance between the outer borders of this cell and the closest blood vessel. Together with a reference of the closest blood vessel sub-element, this value is stored in two signals so that other components can use it. Figure 3.10 shows a schematic of the two possible scenarios when computing the blood neighbor. Once the distance is negative, physical contact occurs.

Density estimator: This component estimates the local density for the given types of cells. Here, the density of the tumor itself is computed. The calculation consists of summing all cell areas in the vicinity and dividing by the circular search area configured by its radius. Therefore, it does not compute the actual density but an approximation. Nonetheless, it is sufficient, as precise measurements are not required. Since cells can overlap in this simulation, the density value can be greater than one. A density visualization of an example tumor can be seen in Figure 3.11. The resulting density values affect the differentiation, as seen later on in the corresponding component.

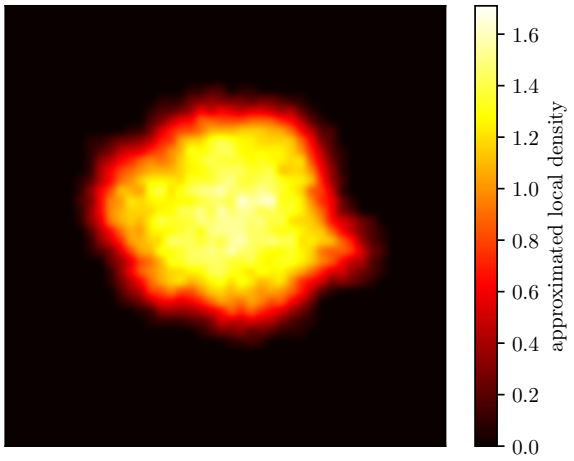


Figure 3.11: Density of an example tumor

Attack: The attack component of tumor cells targets fibers, which are explained later on. Each tumor cell selects the fiber with the lowest health value within its attack range and damages it. Similar to the T cell attack, an attack costs time and health. The computation of when to attack is the same as in Algorithm 2. The difference to T cells is that tumor cells stop before they die of exhaustion.

Health: In contrast to T cells, tumor cells have a longer life expectancy. Since the simulation only represents a short period, no time-dependent health decrease is modeled for tumor cells. However, the healing functionality works in the same way again. Tumor cells can also regain lost health by being in the vicinity of blood vessels. Nonetheless, the most significant function of the tumors' health component is the computation of the damage-induced health decrease caused by T cells. Thereby, two factors influence the way an attack decreases the tumor's health. A tumor cell that has physical contact with a blood vessel only receives half the damage. Secondly, a dedifferentiated tumor cell only takes one-fifth of the damage compared to a differentiated tumor cell. Apart from those two scalars, the damage taken gets mapped directly to the health decrease, as seen in Algorithm 4.

Algorithm 4 Tumor health

```
1: procedure TUMOR_HEALTH(delta_time, blood_boost)
2:     health ← get_health()
3:     damage ← get_damage()
4:     if has_blood_neighbor() then
5:         damage ← damage * C_1
6:     end if
7:     if is_dedifferentiated() then
8:         damage ← damage * C_2
9:     end if
10:    health ← health + blood_boost * C_3 * delta_time
11:    health ← health − damage
12:    health ← min(health, 1.0)
13: end procedure
```

Death: If the health value is below or equal to zero, a tumor cell is considered dead. However, it is not immediately removed from the simulation but after a configurable period in contrast to the T cell. This delayed removal is crucial because dead cells can influence the dedifferentiation of surrounding tumor cells, similar to neutrophils. This interaction works by emitting messenger substances that can be sensed by surrounding tumor cells.

Differentitation: As described at the beginning of subsection 3.2.3, this simulation features a simplified version of differentiation. Each tumor cell has a value ranging from zero to one, denoting the state of differentiation. If it is between zero and one-half, the cell is dedifferentiated. Above one-half and below one, the cell is differentiated. The tumor cells do not actively decide which state is the best one. Transitions only occur based on the surrounding environment. In total, the seven factors shown in Figure 3.12 affect the differentiation value. Under natural circumstances, the tumor tends to be differentiated. Only ten to twenty percent of all tumor cells are dedifferentiated.

Default increase:	Tumor cells have the general tendency to differentiate over time.
Blood boost:	If a tumor cell receives enough nutrients, it is more likely to differentiate.
Density:	Areas with higher densities tend to be dedifferentiated because concurrency produces stress.
Dead neighbors:	Dead tumor cells release substances into the environment that promote dedifferentiation.
Neutrophils:	Have a similar effect to dead tumor cells and also promote dedifferentiation.
Hypoxia:	If tumor cells do not get enough oxygen and nutrients, they become dedifferentiated.
Blood contact:	If a tumor cell touches a blood vessel, it may become dedifferentiated.

Figure 3.12: Differentiation factors

However, therapies and other factors can alter this ratio and increase the number of dedifferentiated tumor cells. The general goal is to keep differentiated tumor cells during therapy as they are easier to kill.

Movement: Generally, only the repulsion-II function describes tumor cell movement, which solely considers other tumor cells. They do not move randomly within the environment. However, due to the mutual repulsion, tumor cells tend to drift apart and get scattered across the environment. Individual cells only get blocked by fibers, which they have to destroy. There is only one exception to this movement behavior. Dedifferentiated tumor cells can crawl on blood vessels. They switch to this mode with a specific user-configured probability and a constant velocity along the blood vessel if they have direct contact. Figure 3.13 shows a schematic of this behavior. While crawling, they are not affected by the fibers, as they squeeze through the gaps around the blood vessels.

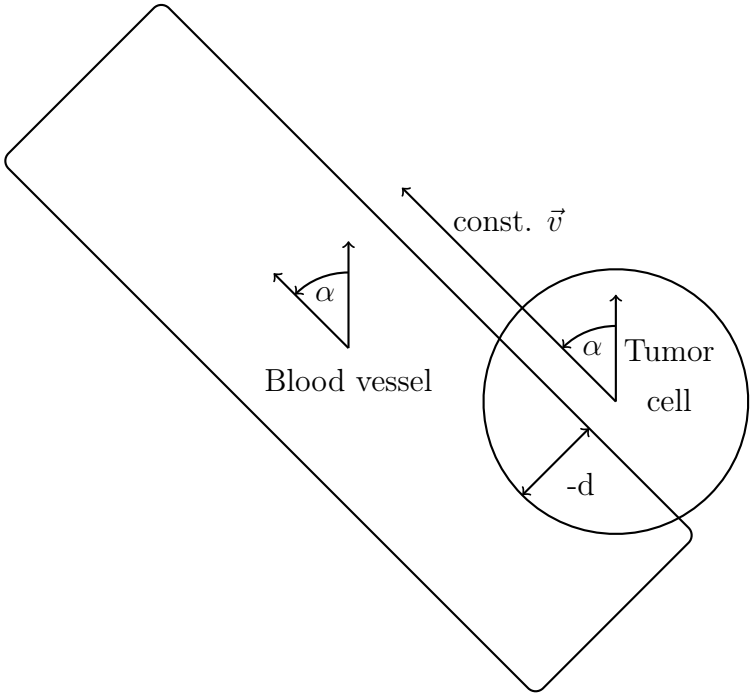


Figure 3.13: Crawling schematic

Spawning: For better comparability of multiple simulations, the tumor gets spawned invariably. A single cluster of tumor cells gets sampled in the center of the environment within a small framed region. The quantity of tumor cells also stays constant across runs.

Due to the initial high density within the cluster, the tumor expands rapidly at first with an unrealistic velocity due to the base repulsion function and high overlap among the cells. In the future, a precomputed tumor might be a better choice to produce realistic results from the start of the simulation without having a start-up phase that distorts some results. This precomputation is possible, as a stored simulation can be used to initialize a new one. In such a case, tumor cells get extracted from the old simulation, along with immune cells and fiber health values. Further, when using the same simulation for initialization every time, this approach also provides comparability.

3.2.4 Blood vessel

Blood vessels are vital since they provide essential nutrients and oxygen for the surrounding tissue. Therefore, cells close to them have an advantage in contrast to cells further away. The effect blood vessels have within this simulation has already been explained in the previously presented cell types. In general, they add a new level of complexity to the simulation and additional dependencies, which might be interesting to analyze.

Blood vessels are modeled as oblong sub-sections. Referring to one of those sections as a cell is inaccurate, as they are composed of multiple different and smaller cells. Since those different cells do not have any effect within the simulation, they can be neglected. Further, modeling the sub-sections as single cells has a computational advantage, as the overall number of required cells is lower.

The actual implementation of blood vessels themselves is pretty simple since they do not have any components. As a result, they are entirely static and do not change throughout the simulation. In the real world, tumors can promote angiogenesis. That is forming new blood vessels from already existing ones. It helps them to grow even bigger and still have enough oxygen and nutrients. However, angiogenesis can be neglected in this work since a single simulation only looks at a short period, while forming new blood vessels requires more time. Therefore, a static blood vessel approach is justified.

The starting point, the endpoint, and the thickness are the only three parameters required to create a linear blood vessel section. Given this information, the specified line will be automatically filled with cells at the correct locations and proper orientation. This implementation allows for an automated generation of blood vessels within an environment. However, the blood vessels are generated by hand to reduce additional parameters and complexity in this first approach.

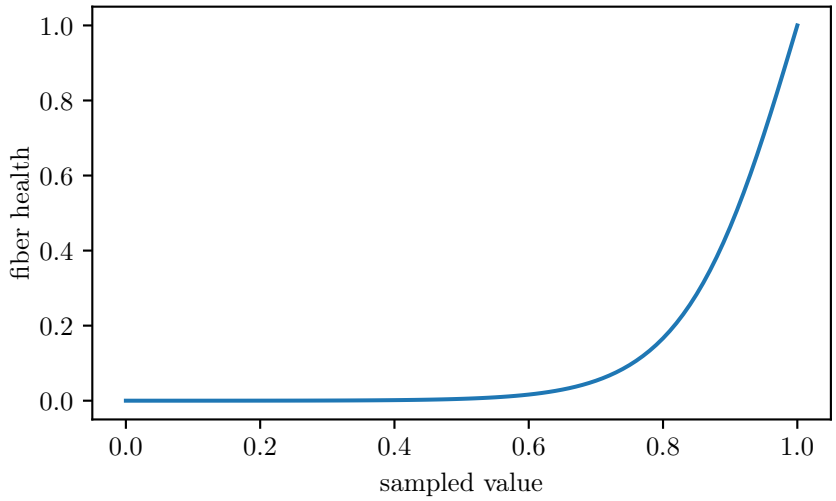


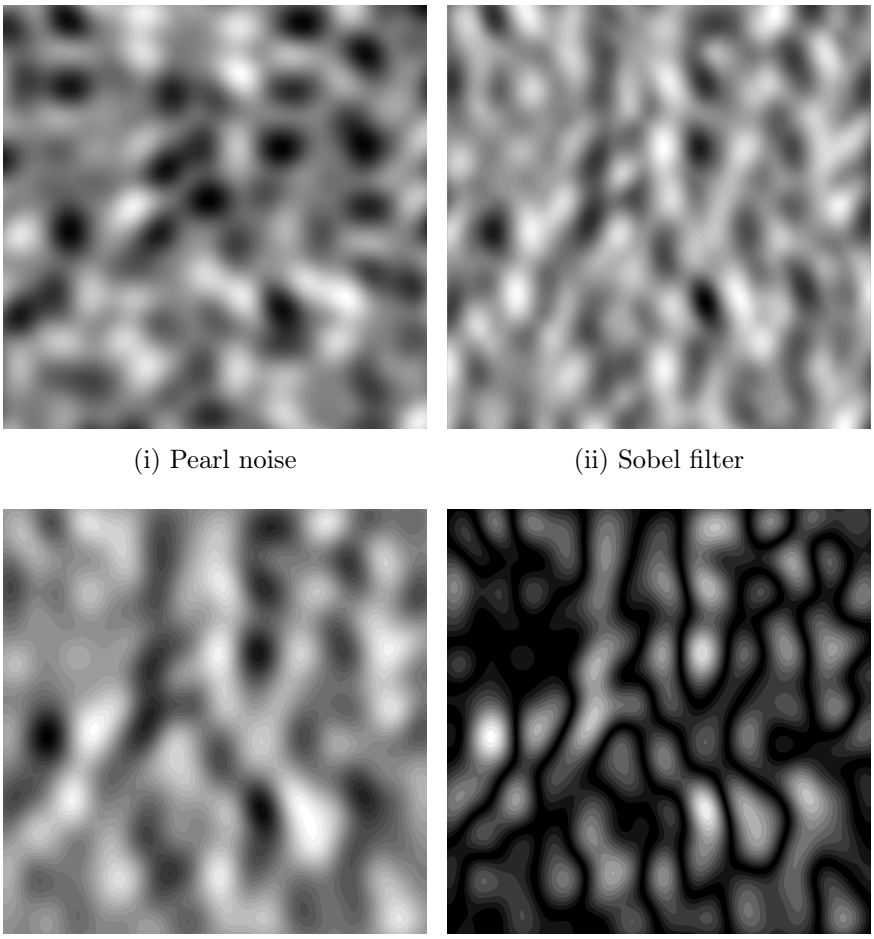
Figure 3.14: Computing fiber health from sampled value

3.2.5 Fiber

The last type of cell implemented in the simulation is fiber. As introduced in section 2.3 and indicated in the section on tumor cells above, fibers are only there to restrict the growth and expansion of the melanoma. In the simulation itself, the fibers have no functionality. They only feature the death component. It removes the fiber as soon as the health is equal to or below zero. This value gets reduced by the tumor cells, as described in subsection 3.2.3.

Similar to the blood vessels, fibers also add to the complexity of the simulation and provide new aspects to investigate. One of them is the role and influence of connective tissue.

The fibers in this simulation are static and are created once at the beginning of the simulation. They have different health values and can thus withstand the tumor for various lengths of time. Thereby, the distribution of more robust and weaker fibers should roughly mimic natural connective tissue. The health values for the fibers are sampled from a specially created map that is constant over all runs. An illustration of the different stages of the map creation process can be seen in Figure 3.15. It consists of the following steps:



(i) Pearl noise (ii) Sobel filter

(iii) Gaussian filter (iv) Inverting lower values

Figure 3.15: Different stages of the health generation map for fibers

(1) The base is a texture generated using the gradient-based simplex noise algorithm invented by Ken Perlin in 2001, the successor and improved version of the famous Perlin noise. It is often used in visual effects due to its organic and realistic look.

(2) In the first processing step, a Sobel filter is applied. It is generally used for detecting edges within an image. Here, it serves a similar purpose. The goal is not to have hills and valleys but to use the transitions between those as strands of denser fibers.

(3) In the next step, a gaussian filter smoothes the image. A high sigma is used to reduce underlying noise further and increase the sharpness.

(4) In the last but one update, the valleys get inverted. As a result, the low spots are no longer the valleys but the longer lines enclosing the hills. These lines represent barriers that are difficult to break for the tumor cells.

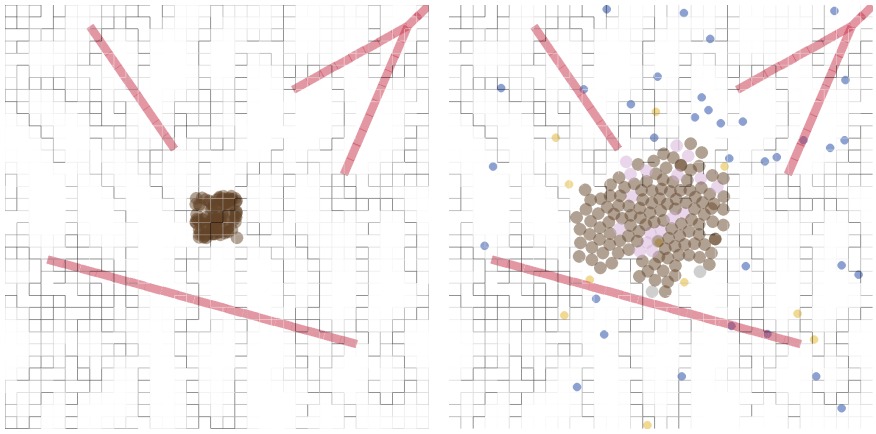
(5) Finally, the generated map gets sampled using a uniformly distributed grid. The resulting values are converted using the function

$$f(x) = \frac{2}{1 + e^{-12x+12}}$$

which is a transformed sigmoid function. Figure 3.14 shows the corresponding plot. The goal is to have clusters of fibers destroyed quickly and stronger fiber barriers that do not get destroyed, or only with a lot of effort. The function separates those two modalities to achieve the desired duality.

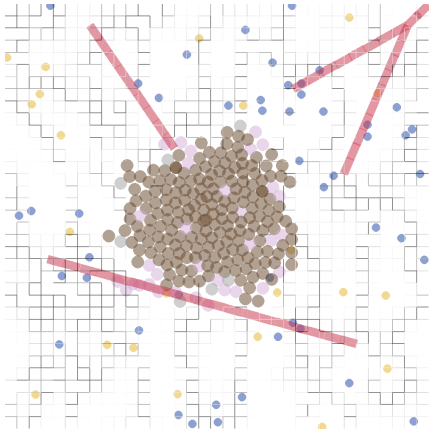
3.3 Example simulation

This final section of the modeling chapter illustrates a visualization of a complete simulation. Figure 3.16 shows the environment for six different stages, at identical intervals in time. The simulation starts with 100 differentiated tumor cells (brown) in the center of the environment. Throughout the simulation, some tumor cells dedifferentiate due to external influences (pink) or die (grey). Further, one can see three blood vessels (red) that do not change during the simulation. The grid structure (gray mesh) represents the fibers, where darker fibers are denser and harder to break. The last cells one can see are from the immune system, namely T cells (blue) and neutrophils (yellow). The simulation displays a case where only the natural immune system responds to the tumor, and no therapy is used.

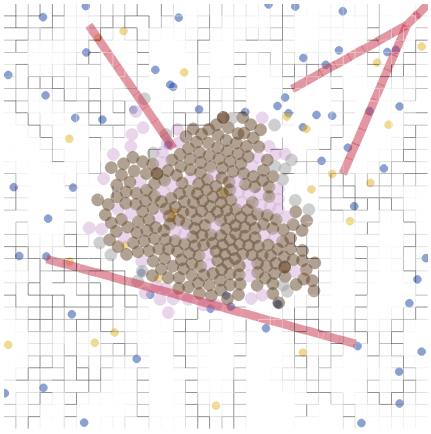


(i) First frame

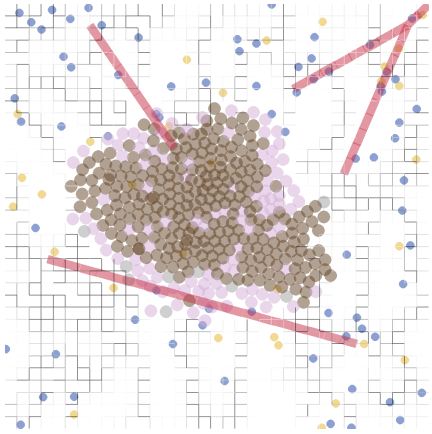
(ii) Stage 1



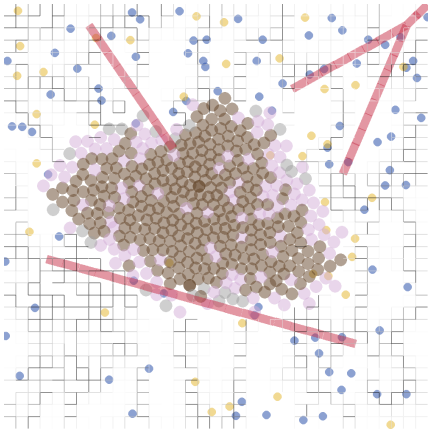
(iii) Stage 2



(iv) Stage 3



(v) Stage 4



(vi) Last frame

Figure 3.16: Example simulation

4 Validation of the simulation

The next step after modeling the simulation is validation. There are two main goals to accomplish. First, from a computer science perspective, it is necessary to validate that the simulation does what it is supposed to do. In other words, the goal is to check for errors and flaws within the implementation. Second, configure the simulation to provide medically correct results, and thus one can validate whether it is possible to draw connections to real-world scenarios. Especially the second goal is not a trivial task since the simulation features many parameters that significantly influence the tumor development and, therefore, the general outcome. There are many possibilities to determine the parameters used in the simulation.

The most apparent method is to use numbers from the medical research field. This strategy is certainly possible for some parameters, like the attack range for T cells since they have to make direct contact with tumor cells, the ratio of the proliferation probability for differentiated and dedifferentiated tumor cells, or the attack speed of T cells. However, since the implemented simulation uses a custom scale for time and size and not the real-world dimensions, those values do not always lead to the desired result. Further, many values are unknown because experts in the medical field do not need them precisely determined. Most of the time, the understanding of biological mechanisms is the crucial part. Additionally, many parameters are implementation-specific and do not appear in the same way in organisms as in the simulation. Examples are parameters related to the life system of cells, like the health decrease over time or due to attacks.

Another possible approach is to gather data through medical imaging from actual tumors. It might be possible to analyze these images using computer vision techniques and construct a fully initialized simulation from the results. Although it is an interesting approach, it is not pursued in this work due to time and complexity constraints, but it might be a new research topic in the future that is worth exploring.

In this work, as stated above, the parameters are a mixture of educated guesses based on medical knowledge about real-world tumors together with known values. However, these values can only serve as a starting point. Fine-tuning is necessary to produce a meaningful simulation. Edge cases that focus on single properties were created to validate the final set of parameters. For these cases, hypotheses were made as to how the tumor should behave. These hypotheses can then be tested using different statistical measures. If these hypotheses are confirmed, and the simulation behaves as expected when changing parameters, the implementation can also be assumed to be correct in an abstract sense. Nonetheless, theoretical testing and debugging have also been deployed but are not shown here. The following parts of this chapter present the different scenarios, the corresponding hypotheses, and the results.

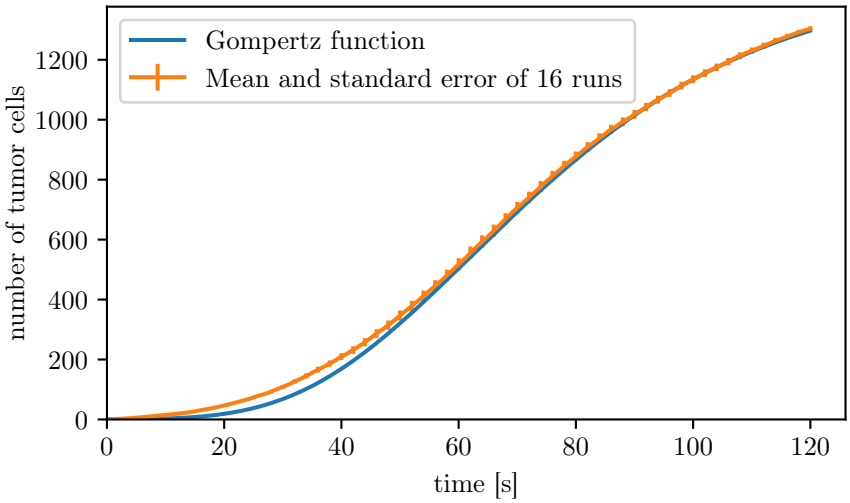


Figure 4.1: Tumor proliferation without any influences

Tumor Proliferation The first test case examines the tumor growth without external influences such as blood vessels, fibers, and immune cells. The simulation starts with a single tumor cell. The extent of the environment limits its growth by prohibiting individual cells from crossing the borders. The goal is to validate whether the simulated tumor features a realistic growth function. This test case is similar to real in vitro tumor growth experiments. Further, creating mathematical models for the growth function of cancer is an active research field. Purely mathematical models, for example, make use of time-delayed ordinary differential equations [10]. Other works, similar to this one, focus on computational approaches [14]. However, there are also some simple standard models. One of which is the so-called Gompertz function:

$$f(t) = a \cdot e^{-b \cdot e^{-ct}}$$

Among other things, it is used to model the growth behavior of tumors. Therefore, it is applied here to evaluate the simulated growth. Figure 4.1 shows a Gompertz function fitted by hand to the mean of 16 different simulations of the described scenario. Even though the curves are not identical, the general behavior is quality-wise the same so that this scenario validates the general tumor growth.

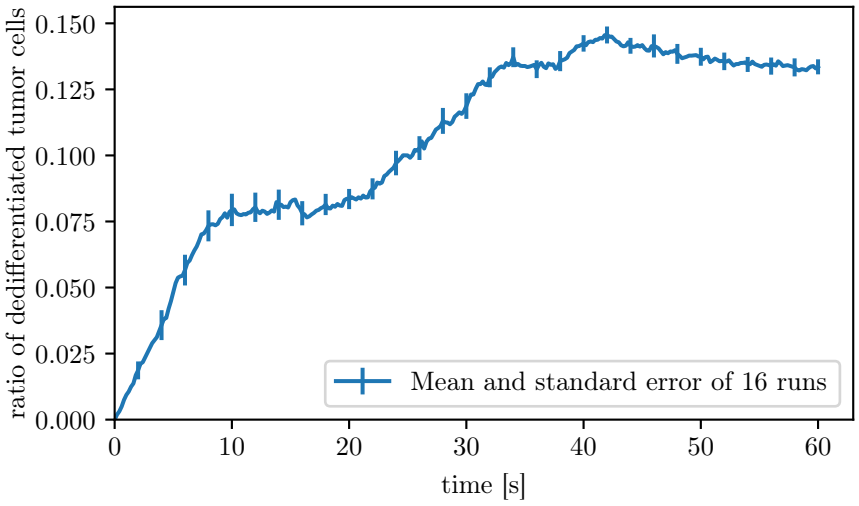


Figure 4.2: Tumor dedifferentiation without the immune system

Dedifferentiation rate The subsequent validation case examines the default dedifferentiation rate. If the tumor does not experience any stress, for example, due to attacks from T cells, it should favor the differentiated state, as it promotes growth. Therefore, T cells and neutrophils get removed in this scenario, but everything else stays original. Especially the blood vessels are essential since they provide oxygen and nutrients. Without those, all tumor cells would have the same probability to dedifferentiate due to the hypoxia-induced stress resulting in a trivial scenario. Nonetheless, a certain percentage of dedifferentiated tumor cells is expected due to cells with direct contact to blood vessels and regions far away from the influence of blood vessels. The target percentage lies somewhere between 10 % and 20 %. Figure 4.2 shows the average rate of dedifferentiated tumor cells over 16 different runs with the corresponding standard error. The rate is computed as the number of dedifferentiated tumor cells divided by overall tumor cells. Therefore, one can observe that the dedifferentiation rate, even with fluctuations, lies under 20 % and is within the expected range. As a result, one can keep the parameters determining the differentiation behavior of tumor cells.

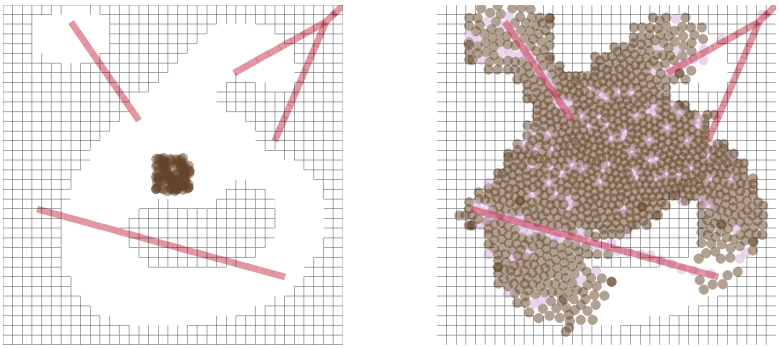


Figure 4.3: First and last frame of the tumor-fiber interaction scenario

Tumor-fiber interaction As the name suggests, this scenario attempts to evaluate the tumor and fiber interaction. As introduced in subsection 3.2.3, the tumor can attack and destroy fibers. However, it will lose health while doing so. As a result, blood vessels are also present in this scenario so that tumor cells can regain their lost health. Immune cells are not required in this scenario. A custom fiber distribution is employed to examine the interaction properly. It consists of broad regions without fibers in which the tumor can expand effortlessly. An actual tumor always uses the simplest path through the surrounding tissue, and it is expected that the simulation shows the same behavior. Over time, however, the tumor should also force its way through denser fiber regions. The second aspect tested in this scenario is the crawling of tumor cells on blood vessels. While doing so, they are not affected by the fibers and, therefore, can reach regions that normal tumor cells cannot reach. Figure 4.3 shows the initial state on the left and the final state of the computed simulation on the right. It can be seen that even if the tumor slowly invades the surrounding fibers, it still generally takes the most accessible route first, as intended. Further, one can see the blood vessels' impact on tumor growth when looking at the blood vessel in the top left. Due to crawling and better healing times, the tumor can break through to the isolated open space. The same can be observed at the left end of the lower blood vessel, where the surrounding fibers were destroyed. Therefore, the expectations were met, and the scenario validates the tumor-fiber interaction.

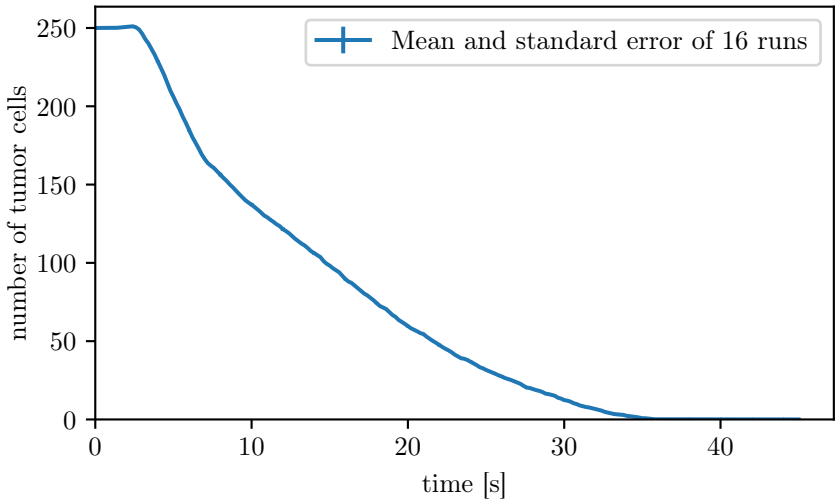


Figure 4.4: Tumor eradication with an excess of T cells

Tumor eradication The following scenario tests whether T cells manage to eradicate a tumor. For the T cells to do this, they have to outnumber the tumor cells by an unrealistic factor. Here a ratio of two to one is used. Real-world experiments use much higher ratios, but the simulation has to use fewer T cells due to the computational restrictions. Nonetheless, the T cells should completely eradicate the tumor for this scenario to be successful. The simulation consists of a tumor in the middle surrounded by a blood vessel square, from which the T cells spawn. This arrangement is essential because the randomly moving T cells spawning from the border might not quickly find the tumor and might die before reaching it. Further, this scenario leaves out fibers and neutrophils.

Figure 4.4 shows the mean number of tumor cells plotted over the simulation time for 16 different runs together with its standard error. One can see that the T cells eradicate the tumor over time. However, they have to reach the tumor first. This aspect of the simulation can be seen in the small plateau at the beginning of the plot. Once the T cells have reached the tumor, the curve gets steeper. At some point, only a few tumor cells are left, and it is less likely that the T cells find them, so the curve flattens a little.

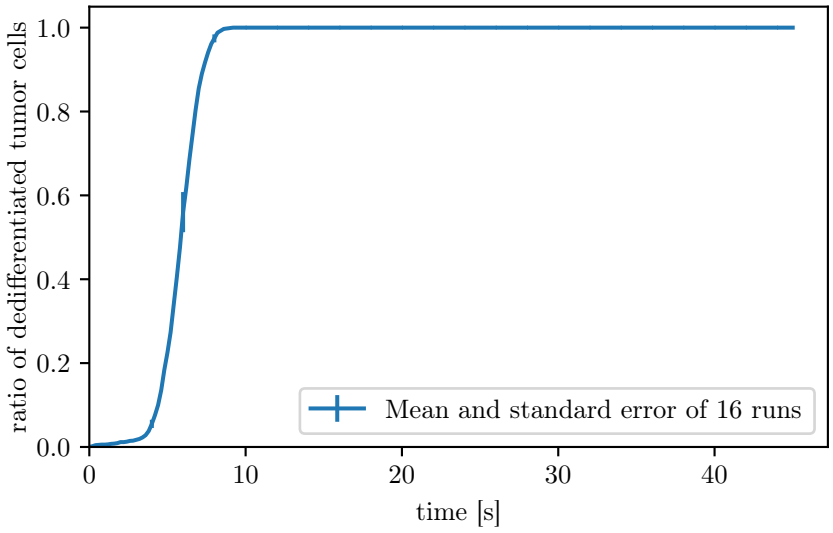


Figure 4.5: Tumor dedifferentiation with an excess of neutrophils

Tumor dedifferentiation using neutrophils This validation case is similar to the previous one. The difference is that this time neutrophils are used instead of T cells, and the goal is not to eradicate the tumor but to force it into an utterly dedifferentiated state. The same simulation composition is used with a square of blood vessels from which the neutrophils spawn, and no fibers are present. The expectation is that all tumor cells dedifferentiate if the ratio of neutrophils is high enough. Since neutrophils do not lose health like T cells do when they interact with the tumor, the ratio can be significantly lower than in the previous scenario. Here, a value of 0.75 is used. The results are shown in Figure 4.5. The high standard error, in the beginning, is based on high fluctuations caused by the randomness of whether the neutrophils interact with the tumor. However, there are so many neutrophils that the tumor cells cannot differentiate again despite the influence of the surrounding blood vessels. This property results in the flat line at a ratio of one for the rest of the simulation. Nonetheless, the general behavior is still as expected, and therefore, the interaction between neutrophils and tumor cells is adequate for the simulation.

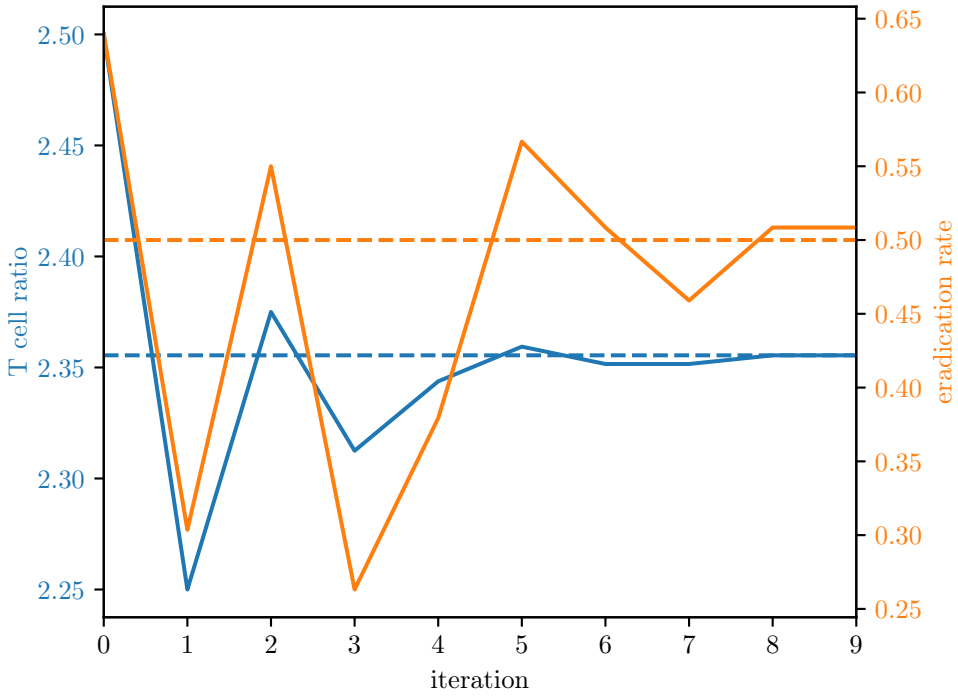


Figure 4.6: Result of the binary search algorithm

Determining the required T cell ratio for tumor eradication The last validation scenario is slightly different from the previous ones. Here, the goal is to find the parameter for the ratio of T cells and tumor cells at which the immune system eradicates the tumor with a 50 % probability. This task can be seen as a simple optimization problem. One evaluation of a ratio parameter requires several runs of the identical simulation to determine the resulting chance of tumor eradication by the immune system. The assumption that more T cells will always have a greater chance of eradicating the tumor is made to speed up the process. This monotonicity constraint allows the use of a binary search algorithm. One has to specify an initial lower and upper bound for the value of interest. The average of the two limits is used as the current parameter. If the percentage of eradication scenarios is higher than 50 %, the lower limit will be set to the currently used value, and the new average gets

computed for the next iteration. The search continues until a fixed number of iterations is done, or a value of 50 % is reached.

Figure 4.6 shows the T cell ratio over the iterations in blue and the probability of eradicating the tumor in orange. As one can see, a final ratio of 2.355 is computed, which appears to be reasonable for such a scenario. One should note that here the complete simulation is used and not a custom one like in the eradication scenario, because of which the two values cannot be compared. The second noticeable aspect is that the monotonicity assumption is not correct. An example that violates this can be seen when comparing iterations two and five. Even though the T cell ratio is higher in iteration two, iteration five has a higher eradication rate. However, the difference is only marginal, so that the used method is still appropriate.

Conclusion Overall, it can be stated that the simulation works as it is supposed to. None of the scenarios produced any unexpected or wrong results. Accordingly, the simulation reflects reality, at least to an abstract extent. The validation did not review all possible circumstances and edge cases that could be examined. Further, it did not compare the results quantitatively by taking actual medical data into account. However, this is not mandatory for this theoretical work. It only aims to show whether such an approach is conceivable in general and does not strive to produce valuable results or gain insights yet. Nonetheless, such procedures are essential in the future. They can be realized eventually by finding parameters based on medical visualization data, as this information can also be used to evaluate predictions.

Apart from the general success of the validation, one crucial feature of the simulation has emerged, namely the variability. It can be best seen in the scenarios determining the regular dedifferentiation rate or searching for the T cell ratio at which the tumor gets eradicated with a 50 % chance. Having a high variability implies that two identically initialized simulations can have completely different outcomes. This phenomenon occurs due to several stochastical decisions within the calculation of the simulation. The random-

ness is justified as representing hidden factors that are not present within the current model used in the simulation. Having a high variability has advantages and disadvantages. From a medical point of view, two tumors and their development are never the same. Having different courses of tumor growth provides more valuable insights. The disadvantages are more of a practical nature. When trying to evaluate, for example, a therapy, it is not enough to run a single simulation, as the variability strongly influences therapy efficacy. As a result, many runs are necessary to get a trustworthy result. The downside to this is the high computational cost. Nevertheless, it is required to get reliable results, and therefore this work computes the same simulation multiple times, as seen later when optimizing the therapy.

5 Methodology

As outlined in section 1.3, the second part of this thesis covers the modeling of therapy for the implemented simulation, employs an evolutionary algorithm to determine optimal therapy parameters, and evaluates the results. In the same way, as the simulation tries to model the real world, the therapies are also based on real-world medications. However, they are fully simplified and adjusted to fit the simulation. The general goal is to investigate whether it is possible to optimize therapy through agent-based simulations and, if yes, to what extent it is valuable in a medical sense. In case of success, one of the aspects to be studied is how similar the resulting optimal therapy plans are to those already used on humans. The second aspect, against the background of evolutionary game theory, deals with the effect of the therapies on the differentiation of the individual tumor cells. Further, it gets examined which insights can be gained from this simple simulation to understand the opportunities that arise from this approach.

This chapter starts by explaining which models and procedures are employed in the optimization process. First, the implementation of the in section 2.3 introduced therapies is presented. The second part of this chapter deals with the evolutionary algorithm's functionality, implementation, and configuration. This explanation includes the variation operators, the fitness evaluation, and the performance metrics. The next chapter will show the actual experimental settings and also evaluates the results, as this one purely focuses on the methodology.

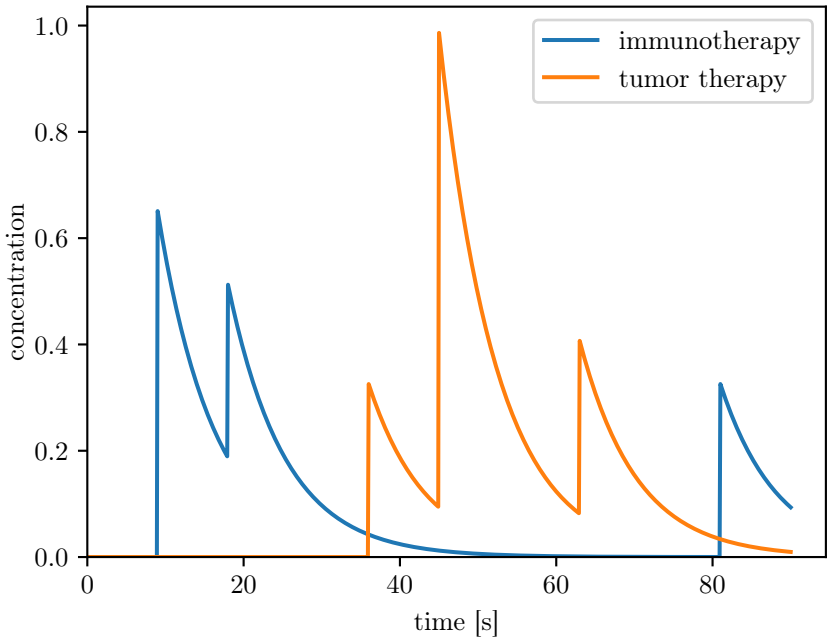


Figure 5.1: Example therapy and the resulting concentrations

5.1 Therapy

The overall medical treatment consists of two individual therapies that target different cells within the organism. A signal transduction therapy, also called tumor therapy or targeted cancer therapy, targets tumor cells, while immunotherapy enhances the T cell attacks. These are two different medications and, therefore, can be given independently of each other. Both of these therapies share the same underlying behavior. The active ingredient concentration in the simulation is always between zero and one. The concentration rises by 33 percentage points with each administration of the corresponding drug. This value is chosen so that the concentration is not always at 100 % but can quickly reach this level. The breakdown of the drug is regulated by an exponential function with a user-configurable half-life. Figure 5.1 shows the concentration of an example therapy plan. At ten seconds, the immunotherapy was given twice, resulting in a concentration of 66 %. By giving a third dose after a few seconds, the concentration increases once again.

Algorithm 5 Apply immunotherapy

```
1: procedure APPLY_IMMUNOTHERAPY(concentration, p_1, p_2)
2:     min_1 ← HEALTH_DECREASE_MIN
3:     max_1 ← HEALTH_DECREASE_MAX
4:     factor_1 ← min_1 + (max_1 − min_1) · p_1 · concentration
5:
6:     min_2 ← DETECTION_RANGE_MIN
7:     max_2 ← DETECTION_RANGE_MAX
8:     factor_2 ← min_2 + (max_2 − min_2) · p_2 · concentration
9:
10:    for tcell in cells do
11:        tcell.set_health_decrease(HEALTH_DECREASE · factor_1)
12:        tcell.set_detection_range(DETECTION_RANGE · factor_2)
13:    end for
14: end procedure
```

5.1.1 Immunotherapy

Immunotherapy enhances the properties of the T cells. Within the simulation, the therapy focuses on the attack component and changes two parameters. First, the T cell loses less life when attacking a tumor cell, and second, it can recognize the tumor cell from a greater distance. This implementation is not necessarily compliant with how the actual medication works within the organism. However, it is a way to implement the behavior here, considering the simulation and its mechanics. Since it is unknown to what extent the therapy should alter the parameters, these values are also to be determined using the evolutionary algorithm. This idea is implemented by specifying a minimal and maximal scaling factor of the original value. Then the algorithm can use values between zero and one to reach any conceivable parameter. Algorithm 5 shows how the computation works. Words in all upper case refer to user-configurable values. p_1 and p_2 refer to the values produced by the evolutionary algorithm and are part of the therapy plan.

Algorithm 6 Apply tumor therapy

```
1: procedure APPLY_TUMOR_THERAPY(concentration, p_3)
2:     min_1 ← MAX_HEALTH_DECREASE_MIN
3:     max_1 ← MAX_HEALTH_DECREASE_MAX
4:     health_decrease ← min_1 + (max_1 − min_1) · p_3 · concentration
5:     max_health ← 1.0 − health_decrease
6:
7:     for tumor_cell in cells do
8:         if tumor_cell.get_health() > max_health then
9:             tumor_cell.set_health(max_health)
10:        end if
11:        tumor_cell.set_max_health(max_health)
12:    end for
13: end procedure
```

5.1.2 Targeted cancer therapy

As the name suggests, the targeted cancer therapy specifically weakens the tumor, as described in section 2.3. Within the simulation, this is implemented by decreasing the maximal health a tumor cell can have. Accordingly, the individual cells are more susceptible to attacks by T cells and have a shorter average lifetime. Furthermore, they cannot spread as quickly because breaking through the fibers exhausts them faster. As introduced in subsection 3.2.3, the tumor cell can regain health that is lost due to attacks from T cells or by attacking fibers. However, the health can only be regained up to the therapy influenced maximal health value. Further, tumor cells that initially have more health than the upper limit lose the corresponding difference when the drug is administered.

Like immunotherapy, the user can specify a minimal and maximal decrease of the maximal health for a tumor cell. Using the therapy parameter and the current concentration, one can calculate the resulting maximal health. Algorithm 6 shows this computation.

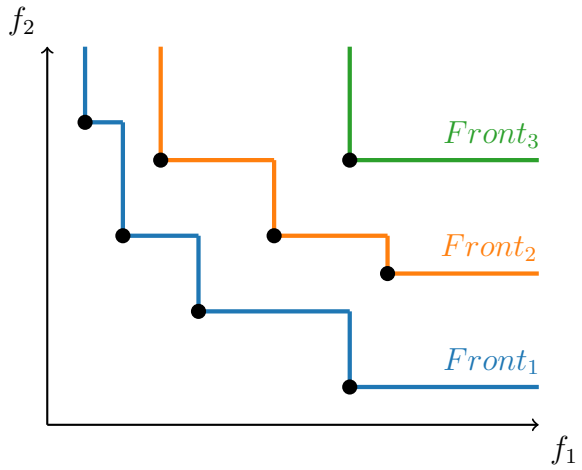
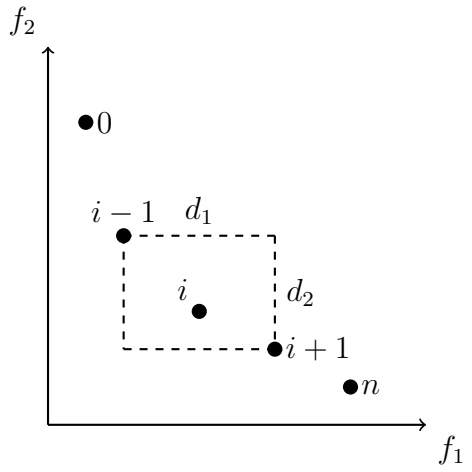


Figure 5.2: Non-dominated sorting for a minimization task

5.2 Evolutionary algorithm: NSGA-II

This work uses standard parameters and a straightforward approach, as no preliminary knowledge is available. Therefore, these values and methods are educated guesses based on other works from different domains. Modifications to the modeling, algorithm, or parameters can be examined in the future based on the findings of this work. Currently, the runtime of the simulation heavily limits the number of possible experiments within the time frame of this work. Nonetheless, it might be enough to determine if it is possible to optimize therapies with the implemented simulation.

The general procedure of an evolutionary algorithm was already explained in section 2.2. This part of the methodology chapter shows the implementation details for the employed evolutionary algorithm. Selecting individuals for the mating pool is quite simple, as the entire population is used for mating. As a result, no selection operator is required. The crossover operator for two individuals will be explained in subsection 5.2.2 and the mutation operator in subsection 5.2.3. The newly created and altered individuals need to be evaluated regarding their fitness. Subsection 5.2.4 explains the fitness functions and how they are computed.



$$norm_1 = f_1(n) - f_1(0) \tag{5.1}$$

$$norm_2 = f_2(n) - f_2(0) \tag{5.2}$$

$$CD(0) = \infty \tag{5.3}$$

$$CD(i) = \frac{d_1}{norm_1} + \frac{d_2}{norm_2} \tag{5.4}$$

$$CD(n) = \infty \tag{5.5}$$

Figure 5.3: Crowding distance

After these steps, the parental individuals must be merged with the newly created children to create the new generation. Since a fixed population size is enforced, not all individuals can survive. For this, the non-dominated sorting genetic algorithm II [4] (NSGA-II) is utilized. It is a widely employed algorithm for multi-objective optimization based on sorting the population into individual fronts. Starting with all individuals, the set of non-dominated ones is part of the first front. If the first front were removed, all individuals that are part of the Pareto front are within the second front. Following this principle, the whole population can be sorted into a set of distinct fronts. Figure 5.2 shows an example, where eight individuals of a minimization task get sorted into three fronts. These fronts then get selected according to their index as long as enough space is available in the new population.

Once a front does not fit into the new population, another method is applied to fill the remaining places. The key aspect for deciding which individuals should survive is diversity, meaning that individuals with far neighbors are preferred because they explore the search space better. One way of measuring diversity is the so-called crowding distance (CD). Figure 5.3 shows the calculation of these values for an example set of solutions. A larger crowding distance is preferable to a smaller one. Therefore, the extreme points receive an infinite

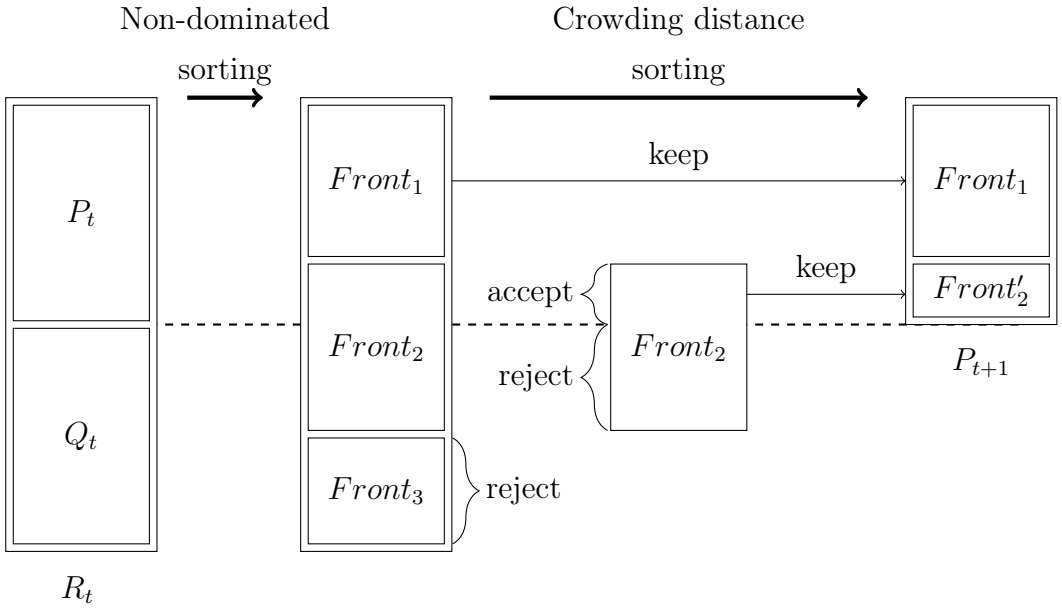


Figure 5.4: NSGA-II selection [4]

crowding distance since they should be kept at all times. For all other points, the crowding distance is computed as the sum of normalized distances to the two neighbors over each dimension of the objective space. After the individuals got their crowding distance assigned, the ones with the highest values get selected into the new population until it has reached its final size. Figure 5.4 visualizes the whole procedure of selecting the new population. Starting with the previous population P_t, a set of offsprings Q_t gets created using crossover and mutation operators. The combined set is called R_t. It gets sorted into separated fronts using the previously explained non-dominated sorting algorithm. $Front_1$ completely fits into the new population and therefore can be selected directly. $Front_2$ does not fit completely. As a result, the best individuals according to the crowding distance get selected to fill up the remaining places in the new population P_{t+1}.

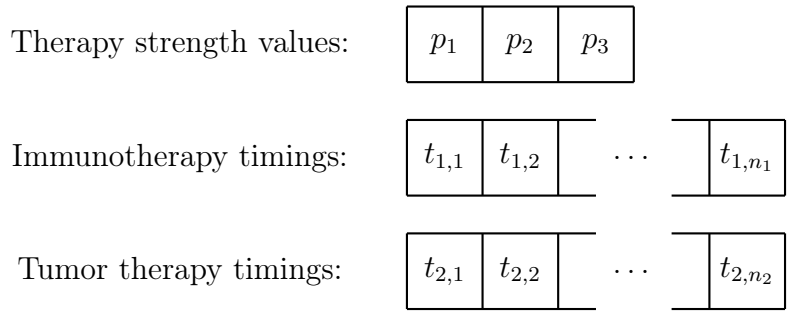


Figure 5.5: Individual modeling

5.2.1 Encoding the therapy

One of the previous sections, section 5.1, explained how the therapies are modeled within the simulation and which parameter they use. These values need to be embedded within a single individual that can be used in the evolutionary algorithm. It must contain the three parameters for the strength of the two therapies and two lists of drug administration times. There are multiple ways to model an individual that contains this information. Figure 5.5 shows the version that is used in this work. If the evaluation shows that the modeling is unsuitable for this task, future work can improve it.

The individual itself consists of three lists. The first list contains the three parameters, p_1, p_2, and p_3, denoting the strength of the therapies. The second list holds time stamps at which the immunotherapy should be administered. Each entry corresponds to one dose. Since different therapy plans should administer the drugs with different frequencies, the list cannot have a fixed length but must be variable in size. Ideally, the entries in this list should be real values between zero and one. It makes it easy to keep the therapies feasible, and one can use standardized variation operators to alter them. That can be achieved by fixing the length of the simulation to a known constant. As a result, zero corresponds to the start of the simulation and one to the end of the simulation. The third list, denoting the administration times for the targeted cancer therapy, works the same way as the second list for immunotherapy.

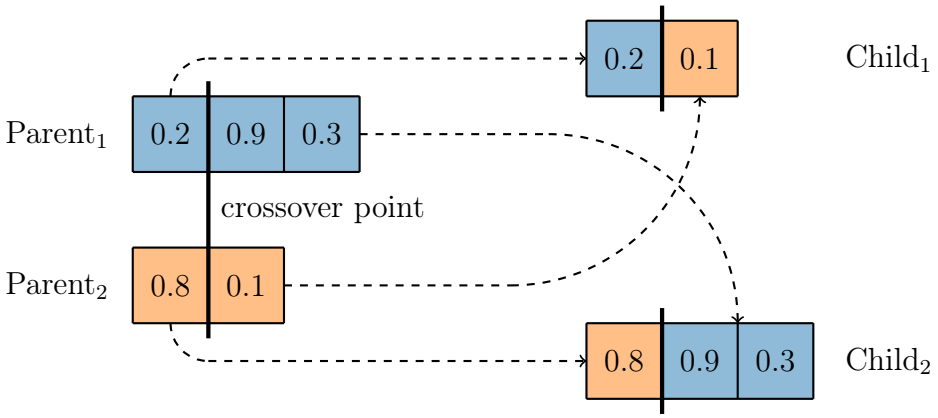


Figure 5.6: One point crossover of different length individuals

5.2.2 Crossover operator

The crossover operator, as introduced in subsection 2.2.1 is responsible for generating offspring given two parental individuals. A simple one-point crossover is used as the actual operator. It chooses a crossover point randomly, at which the genes will be split into two sublists. These then get switched so that each new child contains genes from both parents. Figure 5.6 visualizes this behavior using two dummy individuals consisting of two and three real numbers, respectively. One can see that this method also works with lists of different lengths, as long as both lists contain at least two values. Therefore, it can also be applied for the two lists denoting the time values for the therapies with some minor exceptions when having zero or one administration time. Given two individuals for which the crossover should be computed, the crossover between the strength values gets computed first. Afterward, either the crossover for the matching therapy times gets computed, or the therapies get switched. If they get switched, the crossover operator uses the immunotherapy timings from parent one and the tumor therapy timings from parent two and the other way round. In conclusion, this custom operator based on the one-point crossover can create two new therapy plans given two parental individuals. Whether it is a sufficient operator or needs to be replaced will be seen when evaluation the optimization results.

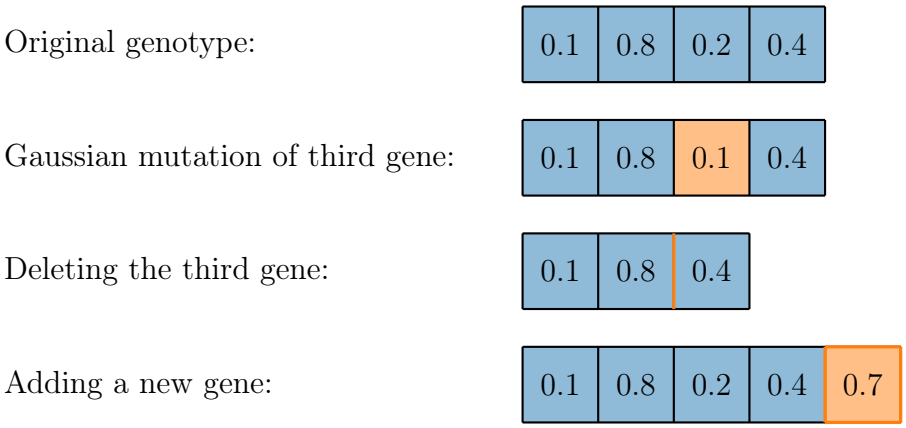


Figure 5.7: Mutation operator

5.2.3 Mutation operator

Since the individual only consists of real values between zero and one, those can be mutated using a simple gaussian mutation operator. If an individual gets selected for mutation, each value has the chance to be mutated. Here a probability of 20 % is used. The mutation operator adds randomly drawn values from a gaussian distribution with a mean of zero and a standard deviation of 0.2.

In addition to the Gaussian mutation, two new operators are introduced. They are specific to the two lists of administration times and manipulate their lengths. The insertion mutation adds a randomly initialized value. It is not essential at which position this new element gets added since the lists are unordered. Each entry denotes the absolute time and no relative time steps. Therefore, the new value gets appended to the end. The deletion mutation removes one value from the list. Since it does make a difference which value gets removed, it is chosen randomly. These two operators help to explore the search space by creating lists of different lengths. Figure 5.7 shows an example for each of the three mutation operators. Blue denotes original genes, while orange shows mutations.

5.2.4 Fitness evaluation

The evolutionary algorithm uses two conflicting objectives to determine the value of an individual. The first objective measures how well the therapy works. Since more drug administrations probably inhibit tumor growth, the second objective must favor therapy plans with fewer drug administrations to be conflicting. This approach is also valid in the real world, where side effects play an essential role. If a patient has more severe side effects due to the medication than the illness itself, it might not be the best therapy plan.

While there are many ways to measure the effectiveness of the therapy, the simplest one is employed. The first objective is equal to the number of tumor cells at the end of the fixed-length simulation. It is easy to compute but does not provide any information about the state of the tumor regarding its differentiation rate, for example. A more complex fitness function and the effects should be examined in future projects. The focus in this work lies on whether it is possible in general. Therefore, the following function is used:

$$f_1 = \|tumorcells\| \tag{5.6}$$

The second objective does not measure the side effects but the fictional cost of the therapy. Since the simulation only represents a small excerpt of the entire organism, it is impossible to make statements about the overall effect. Therefore, the simplifying assumption is made that the side effects increase proportionally with the drugs administered. Additionally, costs also play a significant role in real-world therapy plans so that the here employed approach is reasonable. It makes use of the following function:

$$cost_{immunotherapy} = \frac{p_1 + p_2}{2} * n_1 \tag{5.7}$$

$$cost_{tumor\ therapy} = p_3 * n_2 \tag{5.8}$$

$$f_2 = cost_{immunotherapy} + cost_{tumor\ therapy} \tag{5.9}$$

Both of the objectives need to be minimized.

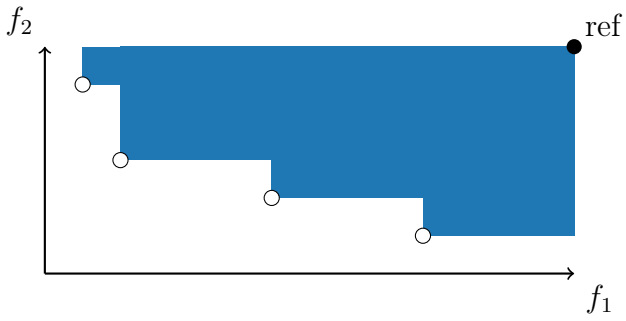


Figure 5.8: Hypervolume

5.2.5 Performance metrics

When evaluating a Pareto front, there are two main criteria, diversity and convergence. Convergence measures how close the points from the computed front are to the actual Pareto front that is the set of globally optimal solutions. Since the problem and its fitness landscape are unknown, using such an evaluation metric is impossible. However, one can compare two subsequent generations during the execution of the evolutionary algorithm to see whether it still improves. The second criteria, diversity, got explained in section 5.2 when talking about the crowding distance. The selection process already considers this measure when choosing the next population. Nonetheless, it should still be used when comparing subsequent populations. A single performance measure that reflects both convergence and diversity is the so-called hypervolume. Figure 5.8 illustrates an example. The hypervolume considers the volume spanned by the individuals in the objective space utilizing a reference point. Here f_1 and f_2 should be minimized, and the reference point must be worse than the worst values in every objective. In this case, the theoretical worst values can be computed for both objectives. The number of tumors cells reaches its maximum when no therapy is applied. Therapy with the maximal number of administrations and strength values of all ones is the most expensive one. The hypervolume is used to determine when the evolutionary algorithm has converged. Once the hypervolume does not increase further, the optimization algorithm has not found any better therapies, and the search can be stopped.

6 Experiments and evaluation

6.1 Experiments

When deciding which scenarios should be used in the evaluation process, there are two different possible strategies. On the one hand, one can employ a computer science approach and use the benchmarks to evaluate the optimization algorithm and how well it solves the problem. On the other hand, one can select the simulation scenarios so that it is possible to gain knowledge about the internal processes and dependencies of the simulation. The latter approach represents the medical point of view. However, there is one obstacle that limits the possibilities in both approaches, namely the runtime. Due to the stochastic nature of a single simulation, multiple runs are required to get a reasonable estimate for the effect of the chosen therapy. Since the evolutionary algorithm also features random operators, it has to be computed multiple times with different random states. With the already high runtime of a single simulation, it is impossible to conduct more than one experiment within the timeframe of this work. Therefore, the one experiment conducted optimizes the therapies for the standard simulation, as it is the most elementary version. For the optimization itself, the algorithm introduced in the previous chapter 5 is applied with a population size of 52 and 50 generations as the termination criterion. A single therapy gets evaluated by computing the same simulation 16 times with different random states to form an average. The whole optimization process is also run five times to cancel out the randomness within the EA. Additionally, a longer test run evaluates more than 100 generations and helps to estimate how many generations are required to find a good solution.

Nonetheless, other aspects worth looking at, mainly from the medical viewpoint, should also be examined. To be able to do this, the optimized therapies from the standard simulation are used to analyze which effect they have on the scenarios explained in the following:

Test variance The first scenario examines to what extent the variability got removed by computing multiple runs of the same simulation. These measurements help to assess the significance of the results. It is essential when deciding whether slight outliers indicate unique solutions or are based on the variation within the simulation. For this, eight equally spaced solutions from the Pareto front are selected. Each of those solutions gets evaluated in the same way as in the optimization by computing multiple runs. However, this gets repeated 50 times. The variations in the different evaluations can then be examined.

Examine isolated therapies The final solutions are broken down into the two partial therapies to examine their isolated effects. In other words, all solutions from the Pareto front are re-evaluated twice, but this time, only one therapy is active at once. It is not the same as optimizing a single therapy, which is not possible due to this work's computational and time limitations. Nonetheless, it might show whether the two therapies amplify each other and a combinational medication is favorable, or a single therapy is sufficient.

Excluding Neutrophils Further, one aspect to examine is the effect neutrophils have within the simulation and the indirect correlation regarding the therapies. This scenario also allows checking whether they help the tumor as it was intended. Since they do not get directly affected by any of the two used therapies, they can just be removed entirely from the simulation. The evaluation then analysis the influence of the optimal therapy for the standard simulation on the adapted simulation that does not feature neutrophils.

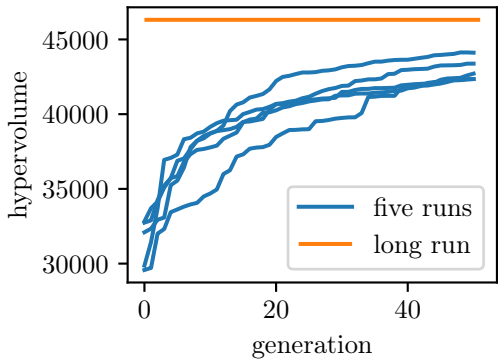


Figure 6.1: Comparison of all hypervolumes

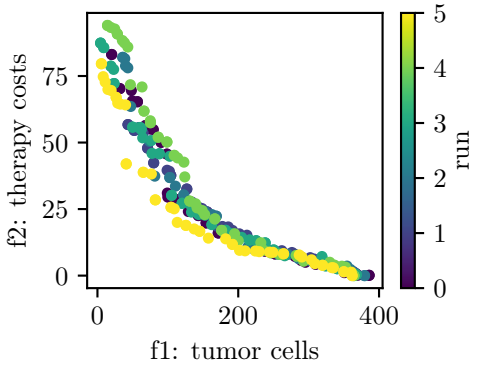


Figure 6.2: Final generations with long run being yellow.

6.2 Results

6.2.1 General optimization results

The first step is to get a basic overview of the general results of the optimization. The goal is to see whether the approach taken was successful. A suitable starting point is to look at the change in the hypervolume over time, as seen in Figure 6.1. It depicts the hypervolume for the five runs and the maximal hypervolume reached by the long run. One can see that all optimizations show similar behavior and seem to converge slowly at the end. However, the gap in the hypervolume to the long run indicates that the five smaller runs are not yet optimal, as a better set of solutions is already known. Figure 6.2 shows the last generation of the overall six runs. One can see that the solutions from the long run are not always part of the Pareto front but in most cases. However, apart from a few solutions that significantly improve the hypervolume, most other therapies are comparatively close to the Pareto fronts of the shorter five runs. As a result, it makes sense to run the optimization as long as possible to find the best therapies. Nonetheless, the five runs are sufficient for the experiment, as the changes are insignificant and do not outweigh the disadvantages of the otherwise long runtime.

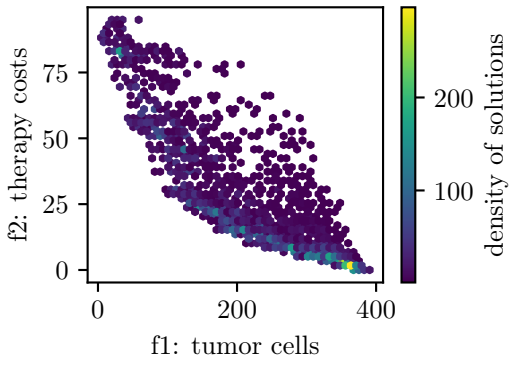


Figure 6.3: Therapies with color-coded density

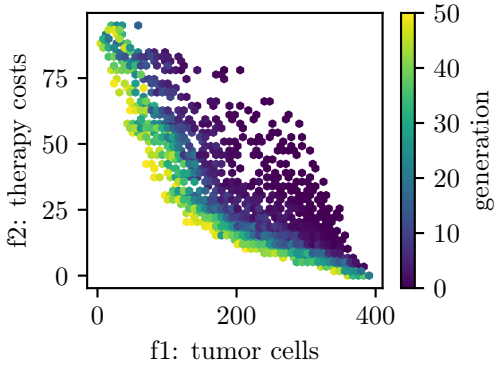


Figure 6.4: Hexbin plot with color coded generations

In the following section, the focus lies on all therapies from all generations. A problem that arises when trying to visualize all of them in the objective space is overplotting. It is often the case that the same therapies live for many generations within the evolutionary algorithm when no better solutions are found. Because they are the same therapy and do not get re-evaluated due to performance reasons, they need to be plotted at the exact location. As a solution, a two-dimensional hexagonal binning plot, also called hexbin plot, is used. Figure 6.3 is such a plot. The color corresponds to the number of solutions within a particular bin. One can see that some areas have nearly 300 solutions per bin while others have only a few. The density distribution also shows that the lower-left border was found quickly and only progressed slowly, which corresponds to the plot of the hypervolume discussed above. The same behavior can be seen in Figure 6.4. Here, the color of each bin corresponds to the average generation. After the first ten generations, the therapies are relatively close to the final front. While no final assessment can be made about the variation operators without knowing the true Pareto front, they did their job increasing the hypervolume. The valuable regions were found quickly, even though new solutions beyond that point took some time to be found. Nonetheless, it makes sense to try out other variation operators and therapy models in future projects to compare different approaches.

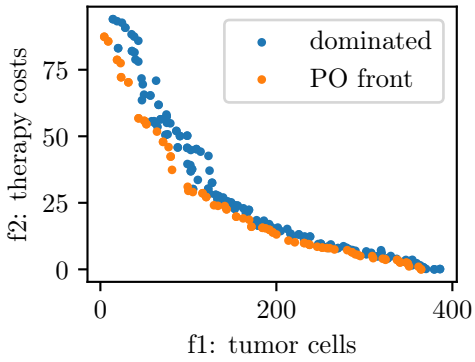


Figure 6.5: Final generations with the Pareto front being marked

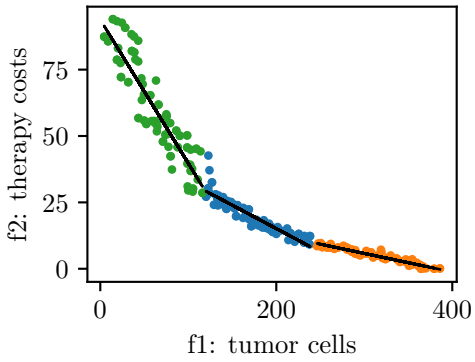


Figure 6.6: Clustered final generations with local linear regression

6.2.2 Analyze optimal therapies

After verifying the general optimization results in the previous section, this one examines the resulting therapies in more detail. The analysis starts with the last generations of the five shorter runs, as they are more homogeneous and introduce fewer outliers than when appending the long run. Furthermore, missing out on a single good solution is not critical for understanding the whole picture. Figure 6.5 shows the resulting dataset with the PO front being marked in orange. When looking at the shape of the dataset, it can be separated into three distinct regions. Figure 6.6 shows such a grouping. It got computed using the k-means algorithm with three clusters. A linear regression line, shown in black, is fitted to each of the individual clusters. One can see that the three lines represent the dataset quite well. The therapies from the orange cluster improve comparably fast without increasing the cost significantly. Nonetheless, their overall effectiveness leaves much to be desired. Therapies in the blue cluster offer a trade-off between efficacy and cost. As seen from the green cluster, a very high price increase must be accepted if the therapy should have an even stronger effect. Further, it seems like finding therapies in that range is more challenging since the density of that cluster is lower in comparison to the other ones.

In the next step, a series of hexbin plots are used, which color-code different aspects using all individuals from all generations and runs. Figure 6.7 shows six of those plots. The color-coded aspect stands next to the color bar. In each plot, the final color gets computed by averaging over all values within an individual bin. The first two plots show the cost of the two therapies, immunotherapy 6.7(i) and tumor therapy 6.7(ii). When looking at the regions close to the Pareto front, it is visible that immunotherapy is the dominating therapy in this simulation. The tumor therapy only gets used in the expensive therapies above a cost of 75. Nevertheless, this finding cannot be applied to reality since only guessed and tuned parameters have been used in the simulation. Other values probably provide other results. As a reminder, the current aim is to see if it is possible to gain insights into therapies from simulations, which is the case here.

Since the tumor therapy is somewhat irrelevant, the two parameters of the immunotherapy are examined in the following. The plot for the stronger parameter of the immunotherapy can be seen in Figure 6.7(iii). It can be seen that this value is crucial in the simulation used, as it is maximal everywhere within the Pareto front. When examining the other parameter of the immunotherapy, the detection range in Figure 6.7(iv), one can see a repetitive pattern on the left side of the plot. The detection range parameter leads to local discrimination between therapies. An increased parameter leads to higher costs but also a slight increase in efficacy. Figure 6.8 and Figure 6.9 are two further visualizations that reflect the observed characteristics. The first plot shows the parameter values over the number of tumor cells for the last generation of the second run. The tumor therapy parameter is mostly near zero, while the stronger T cells parameter is most of the time one. The detection range parameter varies, as described above. On the other hand, the right plot shows the development of these parameters over the optimization process. The lines represent the averages of all individuals from that generation together with the standard error denoted by the vertical tick marks.

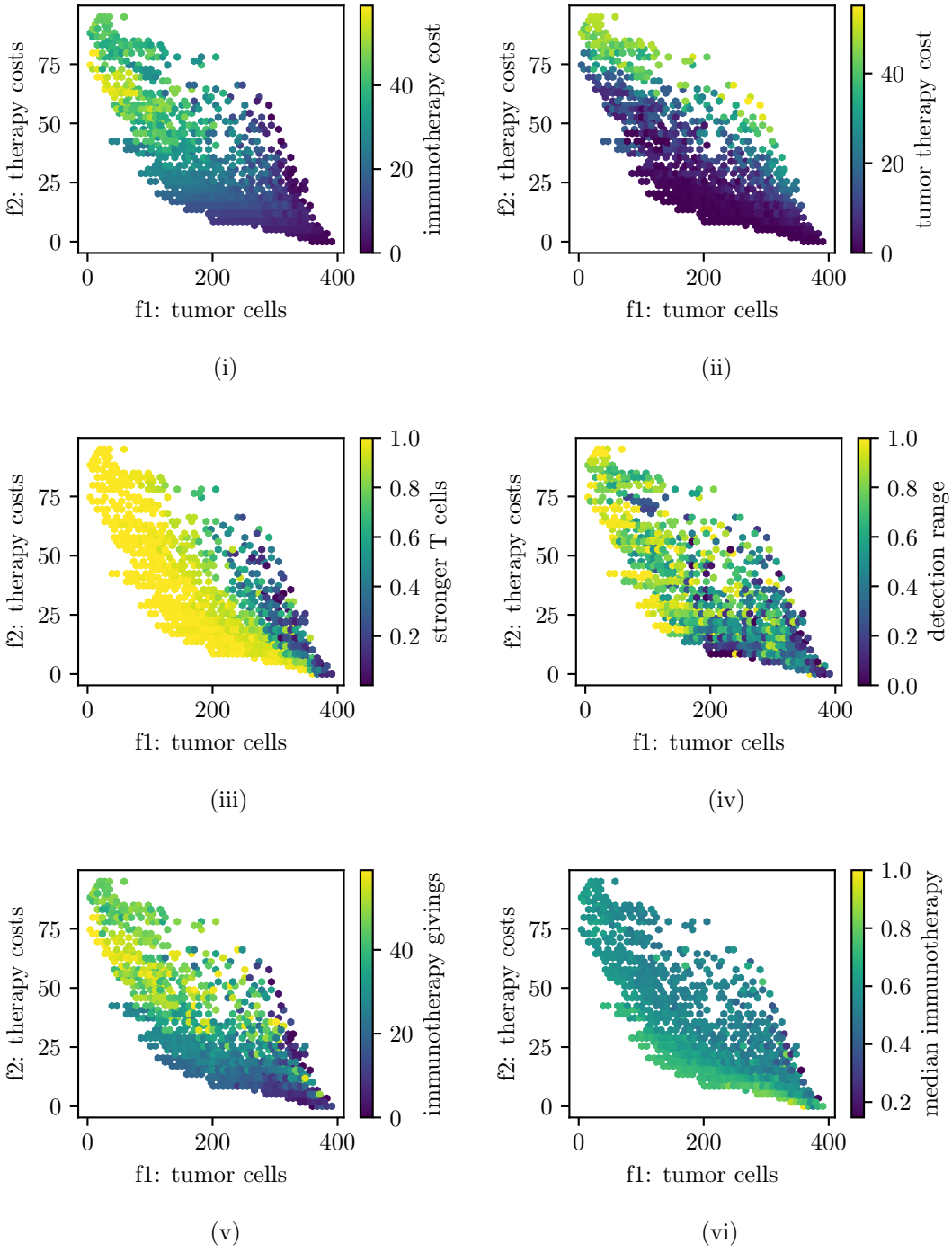


Figure 6.7: Hexbin plots with different color-coded aspects of all individuals

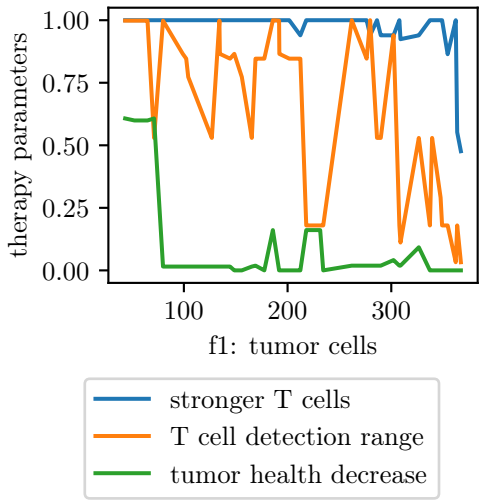


Figure 6.8: Final generation therapy parameters over f1, run 1

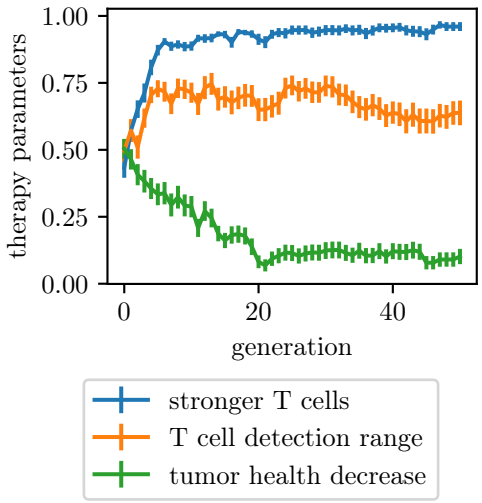
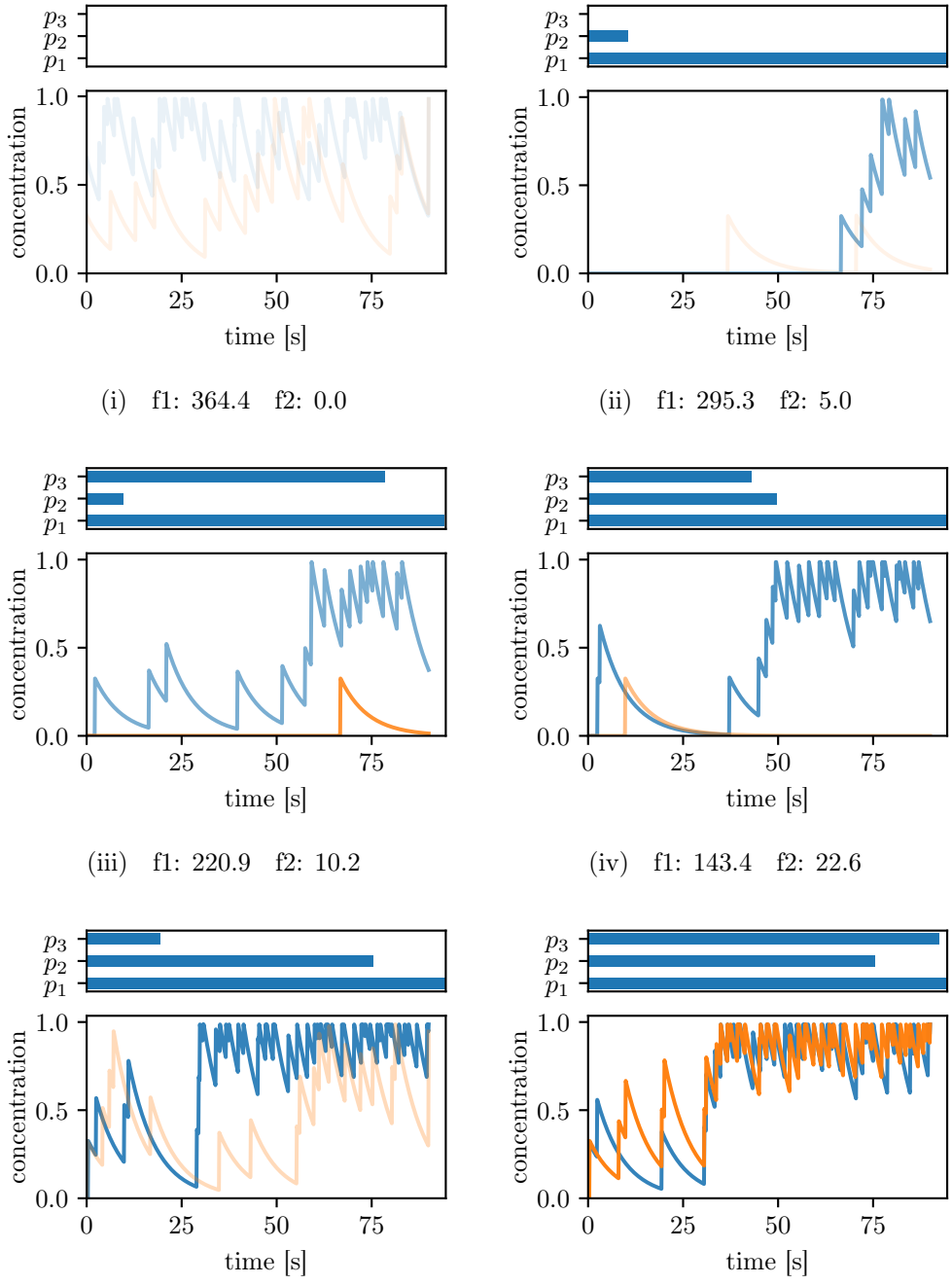


Figure 6.9: Average parameters over the generation, run 1

For the next aspect to be examined, one needs to recall the concentration function of therapy, like it was shown in Figure 5.1. The coverage area of a therapy plan is defined as the area under the concentration curve or, mathematically speaking, the integral of the function. The median can be defined as the normalized point at which 50 % of this coverage area is reached. It is a measure of the skewness of the drug administration's distribution. In other words, the median describes whether the drugs are given more frequently in the beginning, symmetrically, or at the end of the simulation. Figure 6.7(v) shows the overall number of administrations of the immunotherapy and Figure 6.7(vi) the previously introduced median. One can observe that the median has also increased during the optimization. Therapies from the Pareto front feature a relatively high median compared to worse therapies. This observation implies that giving the medications later within the tumor progression is beneficial compared to an early administration. This behavior is especially the case with only a few givings, as seen in the bottom right part of the hexbin plots, resulting in drug administrations at the end of the simulation. It will be further discussed in the following using a different visualization technique.



(i) f1: 364.4 f2: 0.0

(ii) f1: 295.3 f2: 5.0

(iii) f1: 220.9 f2: 10.2

(iv) f1: 143.4 f2: 22.6

(v) f1: 64.3 f2: 51.8

(vi) f1: 4.3 f2: 87.4

Figure 6.10: Equally spaced therapies of the PO front, transparency corresponds to parameter related cost factor

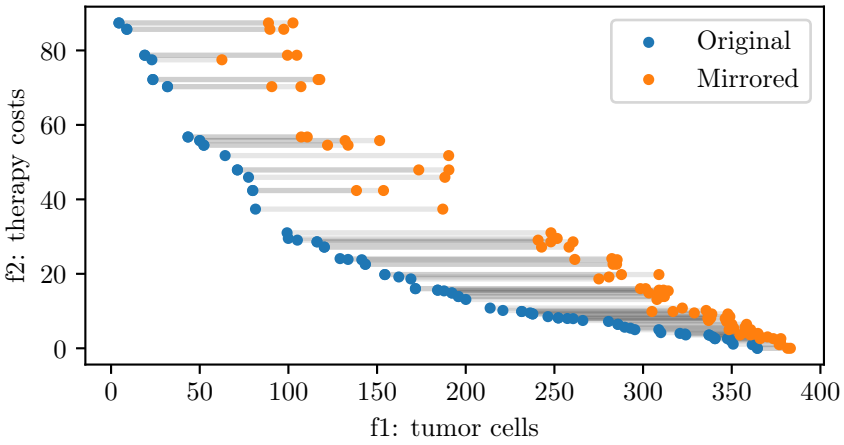


Figure 6.11: Therapy outcome with mirrored administration times

In the last step, six equally spaced therapies from the PO front are visualized in Figure 6.10. Each subplot shows the two concentration curves with the line alpha denoting the average parameter value of the corresponding therapy. The upper bar charts additionally show the actual values of the parameters between zero and one. Comparing these therapy plans to existing ones is challenging, as the frequency and timings are so different. However, it seems realistic since a certain concentration level is being held by iteratively giving the drug, as employed in practice. One thing, however, that seems to be unusual is the skewness. It was already introduced using the median measure and is even more apparent here. One would assume that starting with the therapy as early as possible would result in a high efficacy. For this simulation, however, it is not the case. Figure 6.11 shows a comparison between the original Pareto front and the time-wise mirrored therapies. Starting early with the therapy and administrating no drugs towards the end of the simulation leads to significantly worse results. One of the reasons is probably the unrealistic starting phase of the tumor, as addressed in the modeling section. Analyzing the exact reasons should be part of future research. Nonetheless, tests like these show that it is possible to utilize a simulation to answer all kinds of questions under the assumption of a functioning medical model.

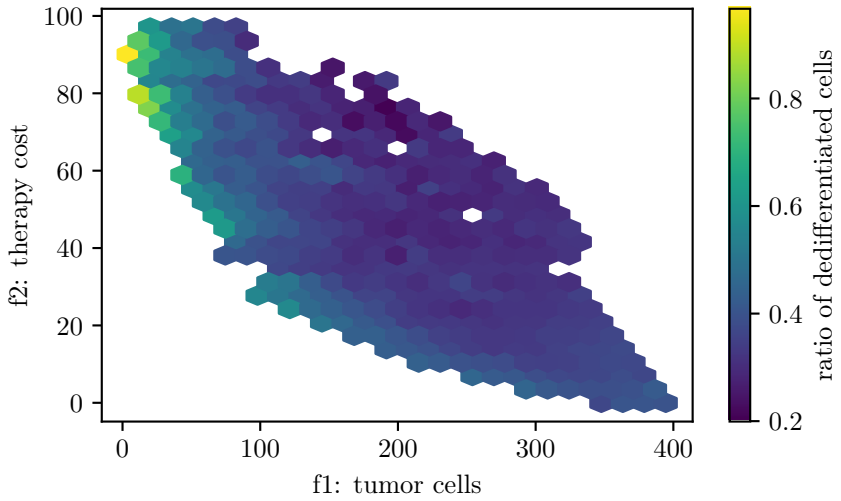


Figure 6.12: Ratio of dedifferentiated tumor cells to differentiated ones

6.2.3 Examine differentiation behavior

The differentiation behavior is, in general, as expected. Figure 6.12 shows a hexbin plot of the solution space with the ratio of dedifferentiated tumor cells divided by differentiated ones being color-coded. One can observe that therapies from the Pareto front tend to have a higher ratio of dedifferentiated tumor cells with increasing efficacy. However, another aspect worth mentioning can be observed when comparing therapies with the same efficacy but different costs. For example, therapies that end up with 200 surviving tumor cells can have different dedifferentiation ratios ranging from 0.2 to 0.45. These observations may indicate that it is possible to optimize and find therapies that do not promote dedifferentiation as much. In case of success, this might allow steering the tumor progression actively. One way of achieving this is by adding the dedifferentiation ratio as a third objective when optimizing the therapies. Other emerging behaviors could not be found in the data due to the lack of complexity in the model. In the future, a more sophisticated version of differentiation, for example, using genes, may allow for more exciting results. However, the general approach and analysis of the data works as intended and can be reused in future works.

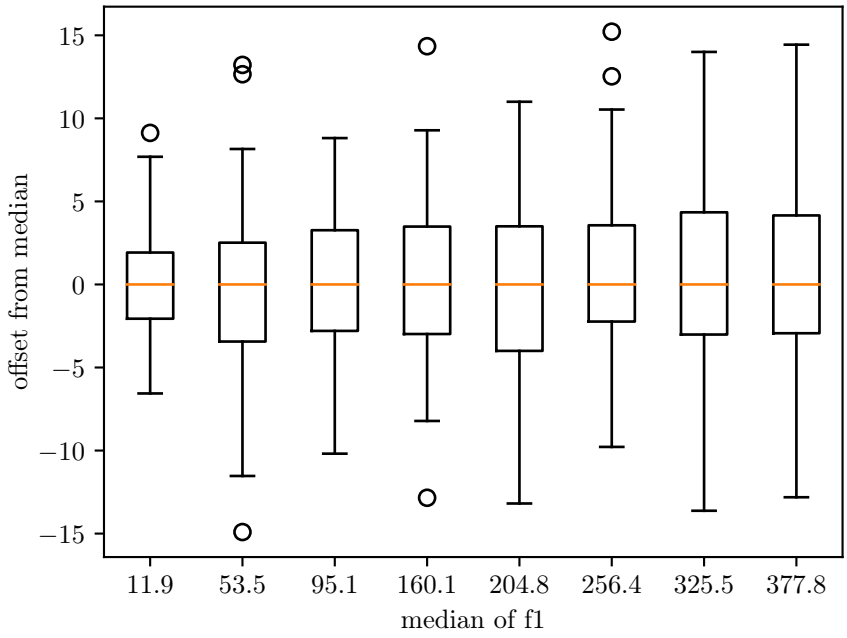


Figure 6.13: Box plot of variance when evaluating individuals 50 times

6.2.4 Benchmark results

Test variance The goal of this benchmark was to determine the remaining variation in the evaluation of therapy, even after computing the simulation 16 times for one individual evaluation. Figure 6.13 shows the results in a boxplot for the eight sampled therapies. They are ordered by their median value for the first objective function while centered on the y-axis. It can be seen that 50 % of all evaluations lie in an interval with plus/minus five tumor cells. The remaining 50 % lie in a range between minus 15 and 15. Additionally, it appears as the variance increases with the median of the corresponding therapy. However, this effect seems relatively small, and there is insufficient test data to make a qualified statement. In conclusion, the variation is slight enough and can be neglected when looking at overall relations and trends. Therefore, the previous evaluations of the results stand and do not need to be questioned. In order to compare two close therapies, however, it makes sense to perform a series of runs and compare the statistical values to get a qualified final result.

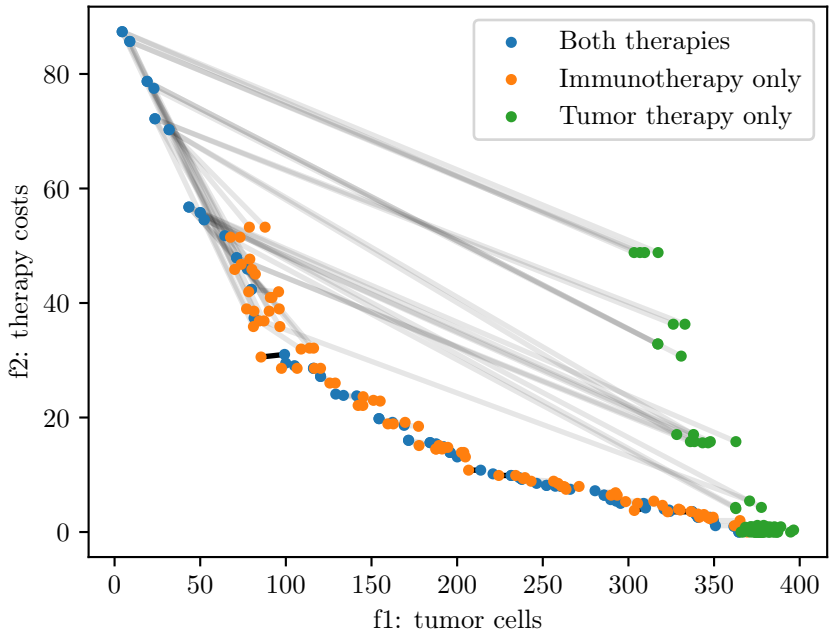


Figure 6.14: Therapy outcome comparison with isolated therapies

Examine isolated therapies An interesting aspect is the interaction of the two individual therapies. For this purpose, the optimized solutions of the Pareto front were split up and evaluated individually. The results are shown in Figure 6.14. Most parts of the Pareto front are somewhat irrelevant regarding this aspect, as only immunotherapy is used. However, if having 100 or fewer tumor cells at the end of the simulation is the goal, tumor therapy comes into play. Without it, the immunotherapy caps at roughly 75 tumor cells. Tumor therapy helps to reduce the number of surviving tumor cells further but requires a significant increase in the costs. Therefore, the two therapies also work together. Nonetheless, the influence of tumor therapy is significantly lower than the influence of immunotherapy. Further, it only seems to be useful when the immunotherapy is already maxed out. One should note that these results cannot be transferred to the actual medicine, as they heavily depend on the simulation parameters, including the modeling of the therapies. Additionally, the single therapies should be optimized separately to investigate the differences in those therapies further.

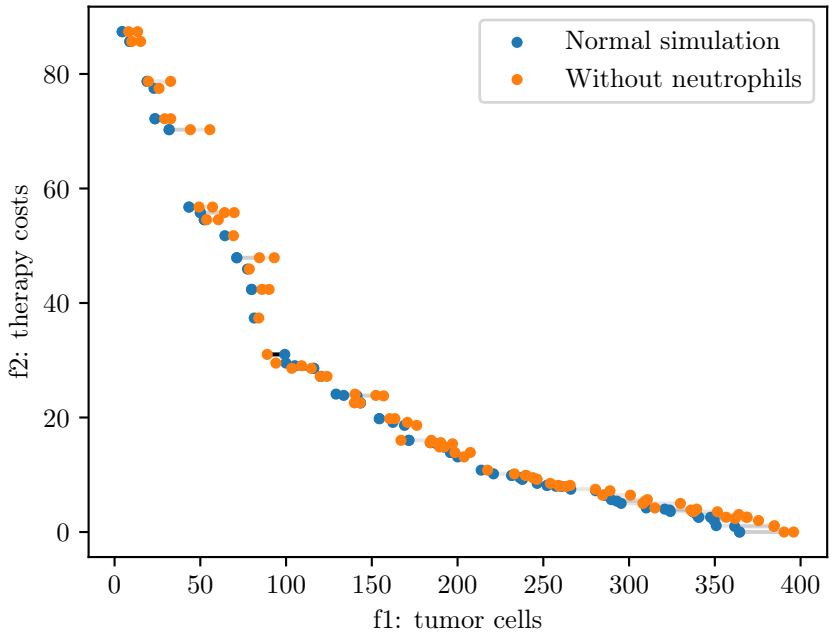


Figure 6.15: Therapy outcome with and without neutrophils

Excluding neutrophils The last benchmark scenario tries to understand better the role of neutrophils in this simulation, especially the influence on the therapy. Initially, the core idea of neutrophils was to help the tumor cells elude the immune system. Therefore, the hypothesis stands that therapies work better in a scenario without neutrophils. Figure 6.15 shows the original Pareto front and the same therapies re-evaluated with a simulation that did not feature neutrophils. At first glance, there is no significant difference, especially considering the observed variation in the evaluation. A subtle difference, however, is noticeable. The therapies seem to perform worse when no neutrophils are present, which contradicts the hypothesis. Another visualization is used to get a better understanding of the differences. Figure 6.16 plots the resulting difference in the first objective function over f_1 and f_2, respectively. A positive value means that more tumor cells survived to the end of the simulation where no neutrophils were present. The red line shows a locally weighted linear regression. As one can see, this regression line is always above zero and therefore indicates that neutrophils do not help the tumor in the used simulation. The

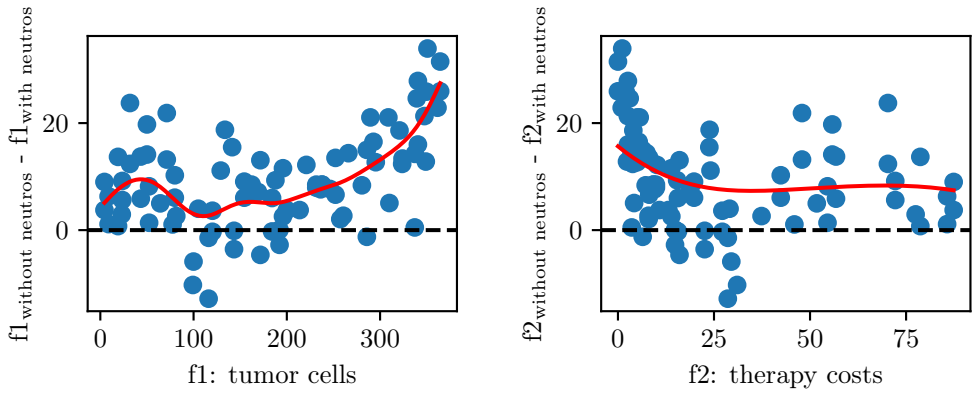


Figure 6.16: Efficiency difference when excluding neutrophils

difference is comparatively high when no therapy is used, probably because more neutrophils are present in those cases, and therefore their overall impact is more significant. This discrepancy between the hypothesis and actual results implies an error in the simulation configuration that needs to be fixed in a further version. However, these results do not allow any conclusions about the actual processes in the simulation. For this, more data needs to be logged during the simulations that describe individual interactions. As a result, such evaluations are not part of this work but are required in the future to get a deeper understanding of the interrelations of different cell types. Nevertheless, the evaluation showed this general approach's promising possibilities, which can be deepened in further works.

7 Conclusion and future

The conclusion is also divided into the two central questions of this work. The first section summarizes the findings of the general simulation. Along with the positive and negative aspects, future relevant issues and tasks are listed. The subsequent section is about the employed optimization and its results. Additionally, the methodology will be evaluated regarding its usefulness and what aspects can or should be improved in the future. The last section of this work summarizes the most important aspects and gives a final outlook on the subject's future.

7.1 Simulation

The implemented simulation did work and provided meaningful data to some extent despite its reduced complexity. The evaluation only revealed some minor problems with the medical model, such as the incorrect effect of neutrophils. However, the manual parameter tuning approach makes it impossible to transfer the results to the real world anyway. An automatic parameter search algorithm or some optimization method is required that uses actual medical imaging data. With this information, the parameters can be tuned to mimic the behavior of human cells correctly. Additionally, they allow comparing predictions generated by the simulation with the actual tumor states.

Another drawback of the current implementation of the simulation is the computational cost. It limits the number of cells within an environment significantly and thus restricts the possible scenarios to small-scale ones. Further,

algorithms using the simulation to determine a fitness value for an individual have to reduce the number of calls to terminate within a reasonable time frame. One can mitigate this drawback to at least some degree by optimizing the implementation. This performance issue should be tackled first, especially since the complexity of the used model increases in the future.

The third and last aspect that emerged when testing the simulation is the variability. Stochastical decisions are used within the simulation to model hidden and, therefore, not implemented factors. On the one hand, this creates different scenarios as they occur in real life, and on the other hand, it makes the optimization of therapies more difficult due to the non-constant results. In future works, it probably is best to limit the amount of randomness and instead add factors that influence the outcome of the simulations. These factors can then also be examined regarding their specific effect on the simulation result. Overall, however, this attempt to simulate skin cancer can be considered successful.

7.2 Optimization

The employed evolutionary algorithm also produced plausible results. However, its performance regarding convergence speed and optimality could not be evaluated due to missing comparisons. These should be created in the future using different and more sophisticated approaches. Nonetheless, the evaluation of the general results showed a clear improvement during the optimization process, and multiple runs confirmed the general trend and Pareto front. Further, the evaluation also showed that it is possible to work with the results and perform tests that bring additional insights into the optimized therapies and their mechanics.

As expected, the variability explained in the last section was also a primary aspect during the optimization. Tests showed that the results of the same therapy could vary to some extend. However, these fluctuations are small enough to be neglected when examining the whole picture and only need to

be considered when comparing similar solutions. Such comparisons, together with an in-depth analysis of the simulation mechanics, were not carried out, as they are not of interest when trying to show whether the general approach is applicable or not. In future works with a more complex medical model, these in-depth analyses are required to understand the interrelations and explain some general results. Additionally, more data needs to be logged to conduct these analyses, for example, interaction data, which was not covered by this work.

Regarding the particular topic of evolutionary game theory and tumor dedifferentiation, the simulation did not work as expected yet. Wrong parameters and the reduced complexity most likely caused those problems. A strong indicator for this assumption is the decreased effectiveness of the therapy when removing neutrophils and thus reducing the dedifferentiation rate. Nonetheless, the trend that therapy caused an increase in dedifferentiated tumor cells was found. It is also interesting to note that therapies with the same efficacy can produce different rates of dedifferentiation. This finding suggests that therapies can also be optimized in this respect, which may ultimately lead to the steering of tumor progression. Overall, therefore, no fundamental problems arose that would invalidate the general approach.

7.3 Future

Overall, the first feasibility study turned out to be successful. An agent-based simulation was modeled and implemented. A therapy plan then got optimized using the simulation and an evolutionary algorithm. However, many questions and possible improvements came up that need to be covered in future works. Therefore it is currently far from being medically relevant, but it showed potential. Exploring such systems is necessary, as in-silico assistance will likely play an increasingly important role in medicine.

Regarding the next steps of this project, the simulation framework needs to be optimized first to allow more runs and thus experiments with more extensive

scenarios. In the second step, reliable methods that determine the parameters of the simulation need to be found. Ideally, they make use of real-world medical data to initialize and verify simulations. Subsequently, the medical model should be expanded step by step to represent a more realistic scenario. This expansion is also the case for the optimization criteria. For example, a better estimation of the side effects is crucial for human trials. Finally, comprehensive tests can be carried out using this advanced simulation to elaborate on the idea of individually tailored therapies that can anticipate and control tumor development.

Bibliography

[1] Gary An. AGENT-BASED COMPUTER SIMULATION AND SIRS: BUILDING a BRIDGE BETWEEN BASIC SCIENCE AND CLINICAL TRIALS:. 16(4):266–273.

[2] Christian Blum and Xiaodong Li. Swarm intelligence in optimization. In Christian Blum and Daniel Merkle, editors, *Swarm Intelligence*, pages 43–85. Springer Berlin Heidelberg. ISSN: 1619-7127 Series Title: Natural Computing Series.

[3] Brendan D. Curti and Mark B. Faries. Recent advances in the treatment of melanoma. 384(23):2229–2240.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. 6(2):182–197.

[5] Morgan A. Giese, Laurel E. Hind, and Anna Huttenlocher. Neutrophil plasticity in the tumor microenvironment. 133(20):2159–2167.

[6] Robert J. Gillies, Daniel Verduzco, and Robert A. Gatenby. Evolutionary dynamics of carcinogenesis and why targeted therapy does not work. 12(7):487–493.

[7] Mark Gluzman, Jacob G. Scott, and Alexander Vladimirsky. Optimizing adaptive cancer therapy: dynamic programming and evolutionary game theory. 287(1925):20192454.

[8] Pavel Hamet and Johanne Tremblay. Artificial intelligence in medicine. 69:S36–S40.

[9] X. Joseph, V. Akhil, A. Arathi, and Pv. Mohanan. Comprehensive development in organ-on-a-chip technology. page S0022354921003956.

[10] James A. Koziol, Theresa J. Falls, and Jan E. Schnitzer. Different ODE models of tumor growth can deliver similar results. 20(1):226.

[11] Lucie A. Low, Margaret Sutherland, Nadya Lumelsky, Seila Selimovic, Martha S. Lundberg, and Danilo A. Tagle. Organs-on-a-chip. In J. Miguel Oliveira and Rui L. Reis, editors, *Biomaterials- and Microfluidics-Based Tissue Engineered 3D Models*, volume 1230, pages 27–42. Springer International Publishing. Series Title: Advances in Experimental Medicine and Biology.

[12] Navid Mahdizadeh Gharakhanlou and Navid Hooshangi. Spatio-temporal simulation of the novel coronavirus (COVID-19) outbreak using the agent-based modeling approach (case study: Urmia, iran). 20:100403.

[13] Joseph A. November. Early biomedical computing and the roots of evidence-based medicine. 33(2):9–23.

[14] H. L. Rocha, R. C. Almeida, E. A. B. F. Lima, A. C. M. Resende, J. T. Oden, and T. E. Yankeelov. A hybrid three-scale model of tumor growth. 28(1):61–93.

[15] Thomas M. Rünger. Ultraviolet radiation and melanoma. In David E. Fisher and Boris C. Bastian, editors, *Melanoma*, pages 51–62. Springer New York.

[16] William H. Sandholm. Evolutionary game theory. In Marilda Sotomayor, David Pérez-Castrillo, and Filippo Castiglione, editors, *Complex Social and Behavioral Systems*, pages 573–608. Springer US.

[17] Merav E. Shaul and Zvi G. Fridlender. Tumour-associated neutrophils in patients with cancer. 16(10):601–620.

[18] Rod Smallwood. Computational modeling of epithelial tissues. 1(2):191–201.

[19] J. Maynard Smith and G. R. Price. The logic of animal conflict. 246(5427):15–18.

[20] Jun Tanimoto. *Fundamentals of evolutionary game theory and its applications*. Number volume 6 in Evolutionary economics and social complexity science. Springer.

[21] Benjamin Wölfl, Hedy te Rietmole, Monica Salvioli, Artem Kaznatcheev, Frank Thuijsman, Joel S. Brown, Boudewijn Burgering, and Kateřina Staňková. The contribution of evolutionary game theory to understanding and treating cancer.

[22] Xinjie Yu and Mitsuo Gen. *Introduction to evolutionary algorithms*. Decision engineering. Springer. OCLC: ocn495781315.

[23] Nastaran Zahir, Ruping Sun, Daniel Gallahan, Robert A. Gatenby, and Christina Curtis. Characterizing the ecological and evolutionary dynamics of cancer. 52(8):759–767.

Declaration of Authorship

I hereby declare that this thesis was created by me and me alone using only the stated sources and tools.

Lukas Bostelmann-Arp Magdeburg, September 27, 2021