Optimal premium strategies with reinforcement learning in push-pull competition between two insurance companies

Jonas Stjernborg

Masteruppsats i försäkringsmatematik
Master Thesis in Actuarial Mathematics

Masteruppsats 2022:2
Försäkringsmatematik
Februari 2022

Mathematical Statistics
Stockholm University
Master Thesis **2022:2**
http://www.math.su.se

Optimal premium strategies using reinforcement learning in push-pull competition between two insurance companies

Jonas Stjernborg*

February 2022

Abstract

How to set the premiums in an appropriate way have been studied ex- tensively by both practitioners and academics during history. The stan- dard approach in literature is to set the premium level according to a model of the expected loss and then adding a safety loading which is related to distributional properties of the risk. In this study, we take another approach and examine the competition between two insurance companies and analyze different versions of the push and pull game between two insurance companies I_1 and I_2 that have initial capital reserves $R_1(0) > R_2(0)$ (one larger than the other). In the game, the larger company aims to maximize the reserve difference $R_1(t) - R_2(t)$ (push the smaller company away) and the smaller company aims to minimize the same (pull closer to the larger company) by using premiums as controls. Using the results in Asmussen et. al (2019) we analytically derive the Nash-equilibrium premiums in the considered games, when taking market frictions $H \sim beta(a, b)$ into consideration. Later on, we implement dynamic programming and reinforcement learning methods to solve the control problem when assuming a simplified state space (reserve difference) and action space (premium levels). The results in this game show that the optimal premium derived with DP and RL is in agreement with the analytical Nash equilibrium solution. The game is later on extended to a case where the market frictions depend on the number of customers currently insured at respective company. Again, a form of analytical result is computed and lastly, the game is analyzed using DP and RL. The results in this game show that if the reserve difference is large, the larger company benefits from offering a lower premium to gain a market advantage. However, when the reserve difference is low, the company can not afford to lower the premium because of the risk of losing the game. The results in the game also indicate that when the company have gained market advantage it benefits from charging higher premiums. This is also in accordance with the analytical results. To summarize, the results show that it is indeed possible to solve and gain insights of the considered optimal control problems using DP and RL. However, in order to use these methods in a real-insurance context, the specific and simplified settings considered in this thesis would need to be evolved and extended in several ways.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: jonas_stj@hotmail.com. Supervisor: Filip Lindskog.

Acknowledgments

I would like to thank my supervisor Filip Lindskog for his advice and valuable feedback during this thesis. I would also like to thank my family, my girlfriend and my friends for their endless support throughout the Actuarial Programme as a whole.

Contents

Glossary

Insurance

Insurance policy: Legal contract between insurer and the policyholder which determines how and when the insurer need to compensate the policyholder if a future uncertain event occur.

Policyholder: The person/s, business or entity buying an insurance policy.

Premium: The price of an insurance policy.

Gross premium: Total premium paid by the policyholder.

Claim: Policyholders request for compensation by the insurance company regarding an occurred event.

Market frictions: Umbrella term collecting all the imperfect market costs that customers face when deciding company. Could for example be, different costs of search and switching company, transportation or preferences.

Reinforcement learning

Reinforcement learning: An area about algorithmic learning that learns which actions that are best in different situations in order to reach some well-defined goal. This is done by taking different actions and then examine how good they are with respect to a numerical reward signal related to the goal.

Agent/Decision rule: The agent/decision rule contain all information of the rewards and is the component that decides which action that should be taken in different states. The decision is made in order to find the optimal policy in an efficient manner.

Exploitation: Selecting actions with the best estimated expected rewards (optimal actions) is called exploiting.

Exploration: Selecting actions that are expected to be suboptimal is called exploring. This is done in order to get more reliable estimates of the expected rewards when taking different actions.

1 Introduction

The information-based society we live in today, combined with the great increase of computer power, have made the application of data-driven methods to make well informed decisions a vital part of almost all businesses. For example, machine learning algorithms and artificial intelligence have become a standard feature in all sorts of decision making. Especially in finance and revenue management the influence of data-driven tools play an important role in order to react rapidly to changed conditions in real time. However, in the insurance industry the digital transformation has been relatively slow. Most of the larger companies want to evolve in the area of data-driven methods and are currently going through changes to make use of the new existing techniques and technologies. One of the major machine learning techniques is reinforcement learning. Reinforcement learning is a computational approach that are able to learn directly from interaction with an environment and it has, for example, been widely used in building powerful computer engines in complex games such as chess and backgammon.

The early history of reinforcement learning can mainly be divided into two threads. One thread is based on trial and error and comes from the psychology of animal learning. The other thread relates to the optimal control problem, using value functions and dynamic programming to find the solution. The two threads were for the main part independent but to some extent they became unified around a third thread of temporal-difference learning methods. During the late 1980s all of these threads came together and developed the reinforcement learning as we know it today. The reinforcement learning algorithm learns what to do in order to maximize a numerical reward signal. As stated before, the method is closely related to the optimal control problem (especially stochastic optimal control formulated as Markov decision processes). Actually all of the work considering optimal control are also seen as work in reinforcement learning (Sutton and Barto, 2018). Today the method is involved in many lines of modernization, for example in the development of self-driven cars or, as in the main focus of this thesis, finding optimal pricing strategies to achieve a specific goal.

Finding an optimal way of pricing its products is an essential part of every business and the insurance industry is not an exception. How to set the premiums in an appropriate way has been studied extensively by both practitioners and academics during history. The standard approach in literature is to set the premium level according to a model of the expected loss and then adding a safety loading which is related to distributional properties of the risk. However, there are also other approaches. For example in Asmussen et al. (2013) and Thøgersen (2016) where the individual customer's decision problem of buying insurance or not at a specific premium level is modeled explicitly. This approach makes it possible to derive the portfolio size as a function of the premium and the insurance company is able to choose optimal premiums that balances revenue and portfolio size so that the risk of ruin is minimal.

In another article Asmussen et al. (2019), this approach of deciding the premium is extended to take the market competition between two insurance companies into consideration. The article consider the two companies I_1 and I_2 with capital reserves $R_1(t)$ and $R_2(t)$ where $R_1(0) > R_2(0)$, namely I_1 is initially larger than I_2. Furthermore, a stochastic variable V is defined to take market frictions into account. The capital reserve difference is taken as state variable and in the article they consider the so called push and pull game where the larger company wants to maximize the reserve difference and the smaller wants to minimize the same. By using premiums as controls they manage to find a pricing strategy that none of the companies are willing to change, a so called Nash equilibrium. The formulation of the game as an optimal control problem makes it feasible to implement reinforcement learning to model the competition situation as a sequential decision process.

In this thesis we will adopt some of the ideas stated in Asmussen et al. (2019) to see how the optimal premium is influenced by the other company in a somewhat simplified insurance setting. Firstly, we will consider the push game of the larger company when the market friction variable V (in our case we will denote it H in order to not confuse it with the value function) does not change for different years and the smaller company has the premium set in a standard way. This will then be extended to the situation where both companies simultaneously try to find the optimal pricing strategy. The problem will be modeled as a Markov decision process (chosen strategy only dependent on the current state) with the reserve difference as state variable. To begin with, the game will be solved analytically and optimal premiums will be derived. Afterwards, the state-transition probabilities for the game will be estimated using a simulation approach. With the state-transition probabilities we can assume that the environment is fully known and from this the problem can be solved using dynamic programming (DP). Lastly, we implement model-free reinforcement learning methods (RL) such as Monte-Carlo (MC) and Temporal-Difference learning (TD) to solve the problem. By analyzing if the RL methods converge to the optimal premiums according to the analytical and DP results we want to conclude if the methods could be appropriate to use in the considered situations.

Later on, the first game situation, will be extended to a game where the market friction variable H changes depending on which company the insured currently holds insurance from. In this scenario we take some inspiration from Krasheninnikova et al. (2019) which uses a reinforcement learning approach to adjust the renewal prices for existing customers. This is a balance between two conflicting objectives, increasing the retention of existing customers and increasing revenue. In many ways we will inherit these conflicting objectives in our problem. Either we could offer a premium that maximizes the revenue in the short perspective or we could offer a lower premium to attract many customers and in this way gain a market advantage. Again, the problem will be modeled as a Markov decision process but this time the state variable will be given as a tuple $s = (d, n)$ where d is the reserve difference and n the proportion of customers. We will present an analytical solution in this problem for different n. Later on, we will estimate the state-transition probabilities (the environment dynamics) using simulation and then make use of dynamic programming to derive some reference of optimal premiums. Lastly, reinforcement learning methods will be implemented and the results will compared to the optimal premiums derived using DP.

1.1 Objectives

The aim of this thesis is to see if it is possible and appropriate to implement reinforcement learning approaches to analyze the market competition part of pricing in a somewhat simplified insurance setting. Surely the large companies in the insurance industry take the competition aspect into account in some way when deciding the premiums. However, using a reinforcement learning approach to support the decision-making in this pricing context is, within my knowledge, not very common yet. Therefore, the main contribution of this thesis is the implementation of the methods in this particular setting.

1.2 Disposition

The thesis will be structured as follows. In section 2 we will go through the theoretical framework of reinforcement learning as a whole. The section will be divided into different underlying parts of the full reinforcement learning problem. Also, the different algorithms used in this thesis will be presented using pseudocode. In section 3, we go through the simulation of insurance data and how the customers choice of company can be modeled explicitly. In section 4, the game situations we want to analyze are described. In section 5, we first derive some theoretical results of the considered games. Then the results of the algorithms implemented in this thesis are presented. Lastly, in section 6, we discuss the results and draw conclusions from the study.

2 Theory of Reinforcement Learning

In this section and subsections we introduce a computational approach to learn from interaction, namely reinforcement learning. Compared to other machine learning algorithms, reinforcement learning is way more focused on goal-directed interaction and are usually effective in solving learning problems of scientific or economic interest. Unless stated otherwise, the theory in this section and subsections follow Sutton and Barto (2018).

Reinforcement learning are methods that learn which actions to take in different situations when the goal is to maximize some numerical reward signal. This is done by trying out different actions, map the results to rewards and in this way discover which actions that create the most reward. After each sample of rewards the followed policy gets updated. A complicating factor is that the actions might not only affect the immediate reward but also the next situation and in this way all subsequent rewards. Since the aim is to learn an optimal strategy to maximize the reward in the long run, we must be careful so that we do not perform too many suboptimal actions if we want to reach our goal. However, in the beginning, the learning agent have no idea of which actions that are optimal and the only way to find out is by learning from interaction with the environment. In this way the agent have learned which actions that give good rewards and make use of this knowledge to make better decisions.

A more formal definition of the reinforcement learning problem could be formulated using the ideas from optimal control of incompletely known Markov decision processes. The idea of this is basically to capture the most important parts of the real problem facing a learning agent which interacts over time with its environment to achieve a goal. This is described in more detail in section 2.3. Furthermore, the learning agent must be able to distinguish the state of its environment in order to take actions that affect the state. The agent also need to have a goal/goals that relate to the state of the environment. Markov decision processes are intended to include just these three key aspects and actually, any method that can be used to solve these kind of problems can be considered as a reinforcement learning method.

Moreover, reinforcement learning is different to both supervised and unsupervised learning in several ways. In supervised learning the correct actions are provided by a knowledgeable external supervisor and the task is then to generalize this to other situations. Unsupervised learning is typically about finding structure hidden in collections of unlabeled data and is similar to reinforcement learning in the way that it does not rely on examples with correct behaviour. However, they are different in the sense that reinforcement has the goal of maximizing the reward signal instead of trying to find hidden structures.

Another (challenging) difference with reinforcement learning in comparison to other learning methods is the task of exploration and exploitation trade-off. In order to obtain a lot of reward the agent should make use of the actions that he knows have been effective in the past but in order to find new possible better actions for the future he also needs to explore. It is not possible to only explore or exploit without failing the task. Therefore, the agent must try a variety of actions and simultaneously try to perform both exploration and exploitation to achieve the goal. For a stochastic task every action must be taken several times in order to get reliable estimates of the expected rewards. Another key aspect of reinforcement learning is that, in comparison to other methods, considers the whole problem of an agent trying to reach its goal in an uncertain environment. For example, other machine learning methods have been studied without specifying how it ultimately could be useful while reinforcement learning the ultimate goal is central. Also, in the beginning it is usually assumed that the agent has to choose his actions even though the significant uncertainty about the environment it faces.

In order to understand the reinforcement learning problem better it can be useful to consider two examples which are given below.

A chess game. In a given position the chess player want to make the best move in order to win the game. This move is chosen by taking the possible replies and counterreplies into account and by comparing the advantages of particular positions and moves.

A robot collecting trash. Based on the current charge level of its battery and how fast it has been able to find the recharger in the past, the robot decides whether it should enter a new room to collect trash or try to find the recharging station.

Both of these examples are easy to overlook and involve the interaction between an active decision making agent and its environment, within which the agent tries to achieve the goal (for example win the chess game) even though its environment is uncertain. The agents actions are allowed to have impact on the future states of the environment as well (for example next chess position) which means that the future actions available to the agent are impacted. These effects of the actions cannot be fully predicted and the agent must because of this monitor its environment frequently in order to be able to react appropriately in any situation.

2.1 Basic elements of reinforcement learning

The agent and the environment are the primary actors interacting and therefore the most crucial parts of the reinforcement learning system. Beyond these, there are essentially four elements to completely explain the system: a policy, a reward signal, a value function, and optionally a model of the environment. These are summarized in the following.

Policy. A policy defines the learning agent's way of behaving at a given time. It is basically a mapping from the perceived states of the environment to actions which should be taken in those states based on the estimated rewards. When taking different actions, we receive different rewards and the policy is updated with respect to this. The policy is seen as the core of a reinforcement learning agent since that it alone is sufficient to determine the behaviour. Furthermore, the policies may be stochastic and map to different actions with different probabilities.

Reward signal. A reward signal defines the goal of the reinforcement learning problem. On each time step we choose an action and in response the reinforcement learning agent gets a single number as a reward. In this way the policy can be updated and we are (hopefully) able to identify bad and good actions at different states. The objective of the agent is then to maximize the total reward it receives over the long run. In general the reward signals may be stochastic functions of the state and the actions taken.

Value function. In comparison to the reward signal the value function describe what is good in the long run. What is good in the long run is decided by taking the states that are likely to follow and their corresponding rewards into account. Roughly speaking, the value of a state is the total accumulated reward an agent can expect over the future when starting from that state. This is somewhat complex and to efficiently estimate values is in many ways the most important component of all reinforcement learning algorithms.

Model of the environment. A model of the environment allow inferences of how the environment will behave and planning actions by considering possible future situations before they are actually experienced. However, as mentioned above, a model of the environment is not necessary in the reinforcement learning problem. These model-free methods are explicitly trial-and-error and can in many ways be viewed as the complete opposite of planning.

2.2 Multi-armed Bandits

Before going through the full reinforcement learning problem in detail we study a simple version of the so called multi-armed bandit problem. The simplified setting that we consider does not involve learning how to act in more than one situation which avoids a lot of the complexity compared to the full reinforcement learning problem. However, we can make use of this particular problem to explore concepts and basic learning methods which we later on can extend to the full reinforcement learning problem. This section and subsections are based on the theory introduced in Sutton and Barto (2018, p. 25-42).

The original multi-armed bandit problem consider a situation where you repeatedly need to make a choice from k different options (actions). The k actions are assumed to be the same every time, namely the environment we consider only has one state. Each choice of action generates a numerical reward that is chosen from a stationary probability distribution dependent on the selected action. The goal is then to maximize the total reward over some long time period, for example 1000 action selections. This is done by considering the expected reward given on each of the k possible actions. The action chosen at time step t and the corresponding reward are denoted A_t and R_t respectively. The expected reward of an arbitrary action a is then given by

$$q_*(a) \doteq \mathbb{E}\left[R_t \mid A_t = a\right].$$

If this quantity was known for every action a we would always choose the action with the highest value and the multi-armed bandit problem would be trivial.

However, we assume that we don't know the actual values of each action but instead consider estimates $Q_t(a)$ that we (obviously) want to be close to $q_*(a)$. There is always at least one of the estimated values that is greatest and the action/actions that generates this value is called greedy. When selecting a greedy action we are exploiting the current knowledge of the action values. If we instead select one of the non-greedy actions we are exploring since we could be able to improve the estimate of the non-greedy actions value. Since it is not possible to both exploit and explore in the same time this is often referred to as the conflict between exploration and exploitation. If we want to maximize the expected reward in one step, exploitation is the right thing to do. If we instead want to maximize the total reward in the long run it might be better to explore since we could find non-greedy actions that actually are better than the greedy action.

The requirement to balance exploration and exploitation is a challenge that repeatedly appears in reinforcement learning. How this should be done depends in a complex way on the precise values of the estimates, uncertainties and the number of remaining steps. However, by considering our simplified multi-armed bandit problem the idea of these methods can be presented in a clear way. Since the true action value is the expected reward when that action is selected a natural estimate is obtained by averaging the actual received reward in the following way.

$$Q_t(a) = \frac{\text{Sum of rewards when } a \text{ taken prior to } t}{\text{Number of times } a \text{ taken prior to } t} = \frac{\sum_{i=1}^{t-1} R_i \cdot I_{A_i=a}}{N_{t-1}(a)}, \qquad (1)$$

where I is a so called indicator function and $N_{t-1}(a) = \sum_{i=1}^{t-1} I_{A_i=a}$. Using the well known law of large numbers it follows that $Q_t(a)$ converges to $q_*(a)$ as the denominator (number of times taking action a) goes to infinity. This method is called the sample-average method and is just one of many ways to estimate action values.

2.2.1 Incremental update

In order to save time and space when implementing the method we notice that the equation (1) can be written as an incremental update procedure in the following way:

$$
\begin{aligned}
Q_{t+1}(a) &= \frac{1}{N_t(a)} \sum_{i=1}^{t} R_i \cdot I_{A_i=a} \\
&= \frac{1}{N_t(a)} \left(R_t \cdot I_{A_t=a} + \sum_{i=1}^{t-1} R_i \cdot I_{A_i=a} \right) \\
&= \frac{1}{N_t(a)} \left(R_t \cdot I_{A_t=a} + (N_t(a) - I_{A_t=a}) Q_t(a) \right) \\
&= \frac{1}{N_t(a)} \left(R_t \cdot I_{A_t=a} + N_t(a) Q_t(a) - Q_t(a) I_{A_t=a} \right) \\
&= Q_t(a) + \frac{I_{A_t=a}}{N_t(a)} \left[R_t - Q_t(a) \right]
\end{aligned}
.
$$

Using this incremental update procedure we don't need to save all rewards and recalculate the mean. Instead we can update the mean easily by the term $\frac{I_{A_t=a}}{N_{t-1}(a)} \left[R_t - Q_t(a) \right]$ which in words can be formulated as the weighted difference of the observed and expected reward. The difference between the observed and expected reward is an error in the estimate and is reduced by taking a step closer to the observed value. When incrementally updating the mean we only need to keep track of $Q_t(a)$ and $N_t(a)$. Because of this, this way of updating is used in many reinforcement learning algorithms to save memory and reduce computation time.

The incremental update rule can also be extended to be useful in nonstationary problems. One way of doing this is by replacing $\frac{1}{N_t(a)}$ with a constant step size parameter $\alpha \in (0, 1]$. The update can then be expressed in the following way (note that we have chosen to not write out the indicator in order to make the steps easier to survey):

$$
\begin{aligned}
Q_{t+1} =& Q_t + \alpha \left[R_t - Q_t \right] \\
=& \alpha R_t + (1-\alpha) Q_t \\
=& \alpha R_t + (1-\alpha) \left[\alpha R_{t-1} + (1-\alpha) Q_{t-1} \right] \\
=& \alpha R_t + (1-\alpha) \alpha R_{t-1} + (1-\alpha)^2 Q_{t-1} \\
=& \alpha R_t + (1-\alpha) \alpha R_{t-1} + (1-\alpha)^2 \alpha R_{t-2} + \cdots + (1-\alpha)^{t-1} \alpha R_1 + (1-\alpha)^t Q_1 \\
=& (1-\alpha)^t Q_1 + \sum_{i=1}^{t} \alpha (1-\alpha)^{t-i} R_i
\end{aligned}
$$

Since the sum of the weights is 1, this is a weighted average. Moreover, since $1-\alpha$ is less than 1 we can conclude that the sum gives less and less weight to older observations. Actually the weights decay exponentially and for that reason this weighted average is often referred to as an exponential recency weighted average. From the sum we also conclude that we give more weight to recent observations for larger parameter α.

2.2.2 Balancing exploration and exploitation

As described earlier, one of the most challenging parts of reinforcement learning is the balance between exploration and expoitation. The simplest policy is to always be greedy (exploiting) and take the action which maximize the estimated value, namely the action selected by the following expression.

$$A_t \doteq \underset{a}{\operatorname{argmax}} Q_t(a).$$

However, since these greedy actions never explore possible better actions it is rarely the best approach in the long run. One simple way of maintaining exploration is to consider the so called ε-greedy policies which act greedily most of the time but with a small probability ε choose from the non-greedy actions at random. The ε-greedy policies have an advantage over the greedy policy since it guarantees that all actions will be sampled infinitely many times in the limit and every $Q_t(a)$ will converge to $q_*(a)$. However, these are only asymptotic results and it does not say much about the methods efficiency in practice.

There are many other ways of balancing exploration and exploitation. For example the upper confidence bound method (UCB) and the gradient bandit algorithm which is described in Sutton and Barto (2018). In order to limit the scope of this thesis we will not go through these methods in more detail. Instead we choose to define the interaction between the learning agent and its environment in a more formal way. This is done in the following section using the framework of Markov decision processes. Using this, the most important features of the artificial intelligence problem can be represented in terms of states, actions and rewards in a simple manner.

2.3 Finite Markov Decision Processes

In this section we represent the reinforcement learning problem in a more formal way using the framework of finite Markov decision processes (finite MDPs). MDPs are meant to be a straightforward framing of the problem of learning from interaction to achieve a goal. The interaction is between the agent and environment and these interact continually when the agent selects actions and the environment responds to these actions with rewards and presenting new situations to the agent. The aim for the agent is then to maximize these rewards over time. The theory in this section and subsections follow the theory presented in Sutton and Barto (2018, p. 47-68).

We can assume that the interaction is made in a sequence of discrete time steps, $t = 0, 1, 2, ...$ At each time step, the agent receives some representation of the environments state $S_t \in \mathcal{S}$. With respect to the current state the agent then selects an action $A_t \in \mathcal{A}(S_t)$ which in the next time-step generates a numerical reward $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ and the agent finds itself in a new state S_{t+1}. In this way the interaction give rise to a sequence of states, actions and rewards in the following way.

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, \ldots$$

If the random variables R_{t+1} and S_{t+1} have well defined discrete probability distributions and only depend on the immediately preceding state S_t and action A_t (given them, not at all on earlier states and actions) then this is said to be a MDP (Markov property holds). Furthermore, if the sets $\mathcal{S}, \mathcal{A}$ and $\mathcal{R}$ all have a finite number of elements it is said to be a finite MDP. In a finite MDP the probability of the specific values of $s' \in \mathcal{S}$ and $r \in \mathcal{R}$ occurring at time t given any preceding state s and action a completely characterize the environment's dynamics and can be defined as follows.

$$p\left(s', r \mid s, a\right) \doteq \operatorname{Pr}\left\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\right\},$$

for all $s', s \in \mathcal{S}, r \in \mathcal{R}$, and $a \in \mathcal{A}(s)$.

By considering the dynamics function p it is possible to compute other quantities of interest in the environment. For example the state-transition probabilities

$$p\left(s' \mid s, a\right) \doteq \operatorname{Pr}\left\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\right\} = \sum_{r \in \mathcal{R}} p\left(s', r \mid s, a\right),$$

the expected rewards for state-action pairs

$$r(s, a) \doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a\right] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p\left(s', r \mid s, a\right)$$

and the expected rewards for state-action-next-state triples

$$r\left(s, a, s'\right) \doteq \mathbb{E}\left[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'\right] = \sum_{r \in \mathcal{R}} r \frac{p\left(s', r \mid s, a\right)}{p\left(s' \mid s, a\right)}.$$

Throughout this thesis we will assume that the Markov property holds even though it is possible to accomplish approximation methods that not rely on it. Using this framework, reinforcement learning methods specifies a way of the agent to improve its policy with the goal of maximizing the long run reward. This idea is expressed as a reward hypothesis in Sutton and Barto (2018, p. 53):

"That all of what we mean by goals and purposes can be well thought of as the maximization of the expected value of the cumulative sum of a received scalar signal (called reward)".

Since the agent always learns to maximize the rewards, the reward signal must be defined and connected to the problem in such a way that maximizing them also achieves our goals. The task of maximizing the rewards can itself be represented in two different ways, one in which the interaction between the agent and environment breaks down into a sequence of separate episodes (episodic task) and one which it does not (continuing tasks). However both these tasks can be expressed in unified notation which enables us to talk about both cases simultaneously.

2.3.1 Return in episodic and continuing tasks

In episodic tasks we need to consider a series of episodes (each with a finite sequence of time steps) rather than a long sequence of time steps which we consider in continuing tasks. In this setting we let the state at time t in episode i be denoted as S_{ti} and in the same way we denote A_{ti}, R_{ti}, π_{ti}, T_i etc. However, when discussing episodic tasks we are almost always considering a specific episode (or stating that something is true for all episodes) and therefore almost never need to distinguish between different episodes. Because of this we almost always abuse this notation in practice and use S_t to refer to S_{ti} and so on. Given the sequence of rewards at each time step the goal of the agent is then to maximize the expected return. The return G_t is defined as a function of the sequence of rewards and in a way which unifies the two tasks considered. This is done by considering G_t as the following sum

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k-t-1} R_k,$$

where $0 \leq \gamma \leq 1$ is the discount factor and T is the last time step which could be both finite and infinite, however γ can't be 1 when $T = \infty$. For the case where T is finite the sum represents the episodic task and the other case represents the continuing task. Moreover the environment we consider could be deterministic or non-deterministic. In a deterministic environment the reward and state following the same state and action of different time steps are the same and in non-deterministic they can differ. The environments could also be stationary and non-stationary, which means that the state transitions and reward dynamics are either the same in different time steps or not.

2.3.2 Optimal policies and Optimal Value functions

Essentially all reinforcement learning rely on estimating value functions of either states (or state-action pairs). The purpose of the value function is to estimate how good (in terms of expected return) it is for the agent to be at a certain state (or performing an action in a given state). Since the expected return in the future depends on the actions of the agent the value function is defined with the different ways of acting, namely policies.

Under a policy π, the value function $v_\pi(s)$ of a state s, is the expected return when starting in s and then following the policy π. In case of a MDP the value function can be defined by the following expression.

$$v_\pi(s) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right], \text{ for all } s \in \mathcal{S},$$

where $\mathbb{E}_\pi[\cdot]$ denotes the expected value of the random variable given that the agent follows policy π and t represents any time step. In a similar way the corresponding action-value function $q_\pi(s,a)$ can be defined by the following expression.

$$q_\pi(s,a) \doteq \mathbb{E}_\pi\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right].$$

These value functions can be estimated by experience. For example, an agent which follows a policy π and get an average of the actual returns following different encountered states, the average will converge to the value of the state as the number of times the state is encountered approaches ∞. In a similar way the action-values can be obtained. Both of these value functions also satisfy recursive relationships between the value of any state s and the value of its possible successor states for any policy π. This equation is more known as the Bellman equation for v_π and can be obtained in the following way.

$$\begin{aligned} v_\pi(s) &\doteq \mathbb{E}_\pi\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p\left(s', r \mid s, a\right)\left[r + \gamma \mathbb{E}_\pi\left[G_{t+1} \mid S_{t+1} = s'\right]\right]. \\ &= \sum_a \pi(a \mid s) \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right], \quad \text{for all } s \in \mathcal{S} \end{aligned} \tag{2}$$

The so called Bellman equation (2) expresses a relationship between the value of a state and the value of its successor states. The value function v_π is the unique solution to its Bellman equation and in this way these equations form the basis in many ways of learning v_π. Reinforcement learning aims to find a policy that maximize the reward obtained in the long run and for a finite MDP these equations can be used to precisely define an optimal policy. Since value functions define a partial ordering of the policies a policy π is said to be better or equal to the policy π' if and only if $v_\pi(s) \geq v_{\pi'}(s)$ for all states $s \in \mathcal{S}$. There is always at least one policy that is better or equal to all other policies. These policies (in case of more than one optimal policy) are all said to be optimal policies and are denoted by π_*. That more than one policy could be optimal don't have any influence on the solution since they are equivalent in the terms of value functions. The optimal state value function can be expressed as

$$v_*(s) \doteq \max_\pi v_\pi(s),$$

for all $s \in \mathcal{S}$ and the corresponding action-value function can be expressed as

$$q_*(s,a) \doteq \max_{\pi} q_\pi(s,a),$$

for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$. The action-value function can also be obtained in terms of v_π by considering the expected return of taking action a in state s and then follow an optimal policy. In this way the following relation is obtained

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right]. \tag{3}$$

Because $v_*(s)$ is the value function with respect to some policy, the Bellman equation holds. However, because this is the the optimal value function it can also be written without any reference to a specific policy in the following way

$$\begin{aligned} v_*(s) &= \max_{a \in \mathcal{A}(s)} q_{\pi_*}(s,a) \\ &= \max_a \mathbb{E}_{\pi_*}\left[G_t \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}_{\pi_*}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a\right] \\ &= \max_a \mathbb{E}\left[R_{t+1} + \gamma v_*\left(S_{t+1}\right) \mid S_t = s, A_t = a\right] \\ &= \max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_*\left(s'\right)\right] \end{aligned}. \tag{4}$$

Equation (4), called the Bellman optimality equation, expresses the fact that the value of a state following an optimal policy must be equal to the expected return when the best action is taken from that state. The Bellman optimality equation for q_* could furthermore be written as

$$\begin{aligned} q_*(s,a) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*\left(S_{t+1}, a'\right) \mid S_t = s, A_t = a\right] \\ &= \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma \max_{a'} q_*\left(s', a'\right)\right] \end{aligned}. \tag{5}$$

In finite MDPs, the Bellman optimality equations for $v_*(s)$ and $q_*(s,a)$ have unique solutions and if the dynamics p of the environment are known both of these can be solved using methods for solving systems of non-linear equations. From this an optimal policy can be obtained in a fairly simple way and form the solution for the reinforcement learning problem. However, this solution rely on at least three assumptions that are rarely true in real world. Firstly, as said before, the dynamics of the environment need to be accurately known. Secondly, the computational power must be enough to complete the calculation and thirdly, the Markov property must hold for the states.

Since these assumptions in general is not met for the problems we consider, other approximate solution methods are more efficient in practice. Some of these are model-based and try to first estimate the model of the environment and then uses it to derive the optimal policy. Others are model-free and instead learns the optimal policy without knowing the model of the environment. In this thesis we mainly focus on the model-free algorithms. However, in order to build a foundation of how these model-free methods work it is useful to first consider the collection of algorithms that can be used to derive an optimal policy given the perfect model of the environment as a MDP (dynamic programming).

2.4 Dynamic Programming

In this section we go through the concept and algorithms of dynamic programming (DP) which can be used when the perfect model of the environment is known and is assumed to be a finite MDP. The idea of DP (which can be extended to reinforcement learning in general) is to use the value functions to structure the search for good policies. The DP-algorithms are constructed by considering the Bellman optimality equations stated earlier and turning these into update rules which improve approximations of the value function. When the optimal value functions have been found using the DP-algorithms we are, as explained in earlier chapters, able to find the optimal policy in a fairly simple way. This section and subsections are based on the theory introduced in Sutton and Barto (2018, p. 73-88).

The procedure of the algorithms can be structured into two main parts. The first part consider the policy evaluation and the second part policy improvement. The policy evaluation part uses an iterative solution method for the set of linear equations which the Bellman equations following policy π give rise to. In this iterative policy evaluation we consider the initial approximate value function at time 0 be arbitrarily chosen and each successive approximation be given by

$$\begin{aligned} v_{k+1}(s) &\doteq \mathbb{E}_\pi\left[R_{t+1}+\gamma v_k\left(S_{t+1}\right) \mid S_t=s\right] \\ &= \sum_a \pi(a \mid s) \sum_{s', r} p\left(s', r \mid s, a\right)\left[r+\gamma v_k\left(s'\right)\right], \quad \text{for all} \quad s \in \mathcal{S}. \end{aligned} \tag{6}$$

The policy improvement part is then used to decide whether it would be preferable compared to the policy followed with respect to the value function. This is done by considering the action a in state s and then following the current policy π. The value of this is given by $q_\pi(s, a)$ and is compared to the value function $v_\pi(s)$. If $q_\pi(s, a) \geq v_\pi(s)$ then it suggests that it is better or at least equal to consider a new policy choosing action a every time state s is encountered. This is a general result called the policy improvement theorem. The policy improvement theorem states that if

$$q_\pi\left(s, \pi'(s)\right) \geq v_\pi(s),$$

for any pair of deterministic policies π and π', then the policy π' must be at least as good as π. This means that the expected return from all states $s \in \mathcal{S}$

must hold the following identity

$$v_{\pi'}(s) \geq v_\pi(s).$$

The proof of the policy improvement theorem is given in appendices A. Moreover, it can be proved that the policies only are equally good when they are both optimal policies.

This way of changing to a greedy action can be extended to consider changes at all states with respect to the best action selected according to $q_\pi(s,a)$ and leads to a new greedy policy π' according to

$$\begin{aligned} \pi'(s) &\doteq \underset{a}{\arg\max} q_\pi(s,a) \\ &= \underset{a}{\arg\max} \mathbb{E}\left[R_{t+1} + \gamma v_\pi\left(S_{t+1}\right) \mid S_t = s, A_t = a\right] \\ &= \underset{a}{\arg\max} \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_\pi\left(s'\right)\right] \end{aligned} \tag{7}$$

in which $\arg\max_a$ denotes the value of a that maximizes the expression. In case of a tie it is chosen arbitrarily between these.

2.4.1 Deriving an optimal policy

By combining the evaluating and improvement parts we end up in an algorithm procedure called policy iteration. By repeatedly evaluating and improving the policy the algorithm are able to obtain a sequence of monotonically improving policies and value functions. Since a finite MDP only consist of a finite number of policies this sequence must converge to an optimal policy and optimal value function in a finite number of iterations. This procedure is described in algorithm 1 below.

Algorithm 1 Policy iteration.

```
1. Initialize value function $V(s) \in \mathbb{R}$ and policy $\pi(s) \in \mathcal{A}(s)$ arbitrarily for
all states $s \in \mathcal{S}$. Assume $V($ terminal state $) \doteq 0$.
2. Policy Evaluation:

while $\Delta < \theta$ (where $\theta$ is a small positive number which determine the accu-
racy of estimation) do
    $\Delta \leftarrow 0$
  for each $s \in \mathcal{S}$ do
    $v \leftarrow V(s)$
    $V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) [r + \gamma V(s')]$
    $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
  end for
end while
3. Policy Improvement
policy-stable $\leftarrow$ true

for each $s \in \mathcal{S}$ do
    old-action $\leftarrow \pi(s)$
    $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a) [r + \gamma V(s')]$
  if old-action $\neq \pi(s)$ then
    policy-stable $\leftarrow$ false
  end if
end for
if policy-stable then
    Return $V \approx v_*$ and $\pi \approx \pi_*$
else
    Go to 2.
end if
```

Policy iteration often converge in relatively few iterations. However, each iteration in policy iteration takes a long time because of the policy evaluation part that itself can be an extensive iterative procedure. Without losing the convergence guarantees of policy iteration this can be handled in several ways by stopping the policy evaluation part before exact convergence. A special case of this truncated policy evaluation technique is the so called value iteration algorithm which stops the evaluation after only one update of each state. This algorithm uses the Bellman optimality equation (2) as update rule, namely

$$v_{k+1}(s) = \max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma v_k\left(s'\right)\right],$$

for all $s \in \mathcal{S}$. Under the same conditions that guarantee the existence of v_*, the sequence $\{v_k\}$ can be shown to converge to v_* for arbitrary v_0. When the optimal value function is found the optimal policy easily can be derived. The full procedure is described in algorithm 2.

Algorithm 2 Value iteration.

```
Algorithm parameter $\theta$ (a small positive number which determine the accuracy of estimation). Initialize value function $V(s)$ arbitrarily for all states $s \in \mathcal{S}$ except $V($ terminal state $) \doteq 0$.

while $\Delta < \theta$ do
    $\Delta \leftarrow 0$
    for each $s \in \mathcal{S}$ do
        $v \leftarrow V(s)$
        $V(s) \leftarrow \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma V(s')]$
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$
    end for
end while
Return deterministic policy, $\pi \approx \pi_*$, such that
```

$$\pi(s) = \arg\max_a \sum_{s',r} p\left(s', r \mid s, a\right)\left[r + \gamma V\left(s'\right)\right].$$

Value iteration combines one sweep of policy evaluation and policy improvement in a clever way. This can be extended by allowing multiple evaluation sweeps between each policy improvement and in this way convergence can be accomplished in less time. For a discounted finite MDP, all of these variations can be seen to converge to an optimal policy.

Compared to the policy iteration algorithm, it takes more iterations to obtain convergence when using the value iteration algorithm. However, each iteration takes less time. When assuming perfect knowledge of the model of the environment both of these methods can be used to solve the reinforcement learning problem but since this assumption is not met in practice these methods are of limited use in real life. However, the general idea of combining policy evaluation and policy improvement are used as foundation in the approximate methods that we will through in later sections.

2.4.2 Generalized policy iteration

In this section we go through generalized policy iteration (GPI) which refers to the general idea of policy evaluation and policy improvement interacting with each other. To be precise, the policy evaluation makes the value function compatible with the current policy and policy improvement make the policy greedy with respect to the current value function. In policy iteration described earlier these two processes alternate but in general this is not needed. For example, in value iteration this is not the case since only one iteration of policy evaluation is done between each policy improvement. However, as long as both these processes keep updating all states, convergence to the optimal value function and an optimal policy are achieved. Furthermore, almost all reinforcement learning methods are well described as GPI which makes it a very applicable tool.

In GPI, the evaluating and improvement processes are in some sense working in opposing directions. Making the policy greedy with regard to the value function typically make the value function inaccurate for the changed policy. Moreover, making the value function compatible with the policy typically makes the policy non-greedy. However, in the long run, the two processes interact with each other and find the solution that make the value function compatible with the current policy and the policy greedy with respect to the current value function and because of this, both processes stop shifting. Thus, both processes only stabilize when the policy is greedy to its own evaluation function. This implies that the Bellman optimality equation holds, which in turn means that the policy and value function must be optimal.

2.5 Monte Carlo Methods

In this section we go through Monte Carlo methods which in comparison to dynamic programming does not assume complete knowledge of the environment.

Instead, these methods only need experience-sample sequences of states, actions and rewards from actual or simulated interaction with an environment. This is very useful since learning from actual experience means that it does not require any prior knowledge of the dynamics of the environment but still can obtain the optimal behaviour. Learning from simulation is also useful since it only need sample transitions from a model, not the complete probability distributions which is needed for DP. In this section and subsections the theory is based on Sutton and Barto (2018, p. 91-115).

The Monte Carlo methods solve the reinforcement learning problem based on averaging sample returns and in order for all returns to exist we only consider episodic tasks. Moreover, the value estimates and policies are only updated when an episode are completed. The procedure is much like the bandit method that we explained earlier but in this case we consider multiple states, each one acting like a different bandit problem. Since the return taking an action in one state also depends on the actions in later states in the same episode these bandit problems are interrelated.

2.5.1 Monte Carlo control with exploring starts

Since state values alone not are sufficient for deciding a policy when a model is not available it is, in these cases, very useful to estimate action values instead of state values. In order to do this we consider the policy evaluation problem for action values that consist of estimating $q_\pi(s, a)$. In the Monte Carlo methods this estimate can be obtained considering the visits of state action pairs s, a (visiting state s and taking action a) in an episode. The first visit MC-method estimates $q_\pi(s, a)$ as the average of the returns given only by the first visits to s, a while the every-visit MC-method averages the returns considering all visits. Furthermore, these methods can be seen to converge quadratically to the true expected values as the number of visits to each state action pair approaches infinity.

The only complication is the general problem of balancing exploration and exploitation that has been discussed earlier. If not taken care of, many state action pairs may never be visited. This is a serious problem since the estimates from these will not be improved with experience, meaning that the method might not be able to choose correctly among available actions in each state. One way of assuring continual exploration are the assumption of exploring starts. The assumption of exploring starts specify that the starting state action pair in every episode are selected from all pairs with non-zero probability and all state action pairs will therefore be visited an infinite number of times in the limit of an infinite number of episodes. This assumption can sometimes be useful but in general it can not be relied on, particularly when learning by actual interaction with an environment.

In order to use Monte Carlo estimation to approximate optimal policies we use the idea of GPI that was explained in the section about DP. In GPI, both an approximate policy and an approximate value function is maintained. Specifically, the value function is changed repeatedly to more closely approximate the value function for the current policy and the policy is improved repeatedly with respect to the current value function. As explained earlier, these procedures somewhat work in opposite directions but together they approach optimality for both the policy and the value function. The policy evaluation is made in the way explained earlier for estimating $q_\pi(s,a)$. Under the assumptions of that we observe an infinite number of episodes and exploring starts the MC-methods will compute q_{π_k} exactly for an arbitrary policy π_k. The policy improvement part can furthermore be done using the current action value function to construct the greedy policy π_{k+1} (no underlying model is needed). The policy improvement theorem then holds for π_k and π_{k+1} since

$$\begin{aligned} q_{\pi_k}\left(s, \pi_{k+1}(s)\right) &= q_{\pi_k}\Big(s, \arg\max_a q_{\pi_k}(s,a)\Big) \\ &= \max_a q_{\pi_k}(s,a) \\ &\geq q_{\pi_k}\left(s, \pi_k(s)\right) \\ &\geq v_{\pi_k}(s), \quad \text{for all} \quad s \in \mathcal{S} \end{aligned} .$$

The assumptions of observing an infinite number of episodes and exploring starts is however not very likely to hold. A way of handling the first assumption for policy evaluation is to not complete the policy evaluation before returning to policy improvement. In this way, we move the value function towards q_{π_k} in every evaluation step, however we do not expect to come close until many steps are done. For MC-methods it is reasonable to alternate between evaluation and improvement on an episode-by-episode basis. The assumption of exploring starts can also be removed by considering only policies that are stochastic with nonzero probability of selecting all actions in every state. This can be obtained using two different approaches, namely on-policy methods and off-policy methods.

2.5.2 Monte Carlo Control-on policy

In this section we go through the MC on-policy methods for maintaining continual exploration. The only way of doing this is by ensuring that all the actions keep on being selected by the agent. One example of these methods is indeed the assumption of exploring starts that we considered earlier but we can also construct others without this unrealistic assumption. Generally the on policy control methods are considered to be soft. This means that $\pi(a \mid s) > 0$ for all states and actions but is gradually getting closer and closer to a deterministic optimal policy.

We consider the ε-greedy policies that were mentioned in the multi-armed bandit section. This method choose a greedy action with probability $1-\varepsilon$ and at random choose one of the actions with probability ε. This means that the probability of selecting the greedy action is $1-\varepsilon+\frac{\varepsilon}{|\mathcal{A}(s)|}$ and all other actions have the probability of $\frac{\varepsilon}{|\mathcal{A}(s)|}$ to be selected. Moreover, the ε-greedy policies are examples of ε-soft policies which are defined as policies where, for some $\varepsilon > 0$, it holds that $\pi(a \mid s) > \frac{\varepsilon}{|\mathcal{A}(s)|}$ for all states and actions.

Using this, the general idea of GPI can be applied since GPI only require that the policy are moved towards a greedy policy, not all the way to it. Furthermore, the policy improvement theorem holds (for proof, see appendices A). This means that for any ε-soft policy π we have that any ε-greedy policy with respect to q_π is better than or equal to π. The equality, again, can be showed to only hold when both of the considered policies are optimal among ε-soft policies. Using this we are able to find the best policy among all ε-soft policies. In other words, the policy iteration works for ε-soft policies. That we only will be able to find the best policy among ε-soft policies is not optimal but a reasonable cost for getting rid of the unrealistic assumption of exploring starts. Using the first visit method explained in earlier sections for estimating the action value function the complete algorithm can then be written in pseudocode as in algorithm 3 below.

Algorithm 3 On-policy first-visit MC control (ε-greedy).

```
Algorithm parameter: small $\varepsilon > 0$.
Initialize:
    $\pi \leftarrow$ an arbitrary $\varepsilon$-greedy policy
    $Q(s,a) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
    Returns $(s,a) \leftarrow$ empty list, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$
Loop forever:
Generate an episode following $\pi : S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$
    $G \leftarrow 0$
for each step of episode $t = T-1, T-2, \ldots, 0$ do
    $G \leftarrow \gamma G + R_{t+1}$
   if the pair $S_t, A_t$ not appears in $S_0, A_0, S_1, A_1 \ldots, S_{t-1}, A_{t-1}$ then
    Append $G$ to Returns$(S_t, A_t)$
    $Q(S_t, A_t) \leftarrow$ average (Returns$(S_t, A_t)$)
    $A^* \leftarrow \arg\max_a Q(S_t, a)$ (if it is a tie $\rightarrow$ broken arbitrarily)
      for all $a \in \mathcal{A}(S_t)$ do
    $\pi(a \mid S_t) \leftarrow \begin{cases} 1-\varepsilon+\varepsilon/|\mathcal{A}(S_t)| & \text{if } a = A^* \\ \varepsilon/|\mathcal{A}(S_t)| & \text{if } a \neq A^* \end{cases}$
      end for
   end if
end for
```

2.6 Temporal-Difference Learning

In this section we combine the ideas from DP and MC-methods in the earlier sections to define the temporal-difference (TD) learning. TD-methods both inherit the ability to learn directly from experience without a model of the environment (as in MC-methods) and update estimates partly based on other learned estimates without waiting for the final outcome (as in DP). When TD-methods for estimating the value function (policy evaluation) has been established the control problem (finding an optimal policy) also follow a variation of GPI. This section and subsections follow the theory introduced in Sutton and Barto (2018, p. 119-138).

2.6.1 Policy evaluation

The policy evaluation part of TD-methods is similar to the MC-methods in the sense that it uses experience to update the estimate of the value function for all non-terminal states when following a policy π. However, in MC-methods the return is not known until the episode has ended and because of this, the value function can not be updated until then. This is not the case in TD-methods which only need to wait until the next time step $t+1$ to directly form a target and make a useful update using the observed reward R_{t+1} and the estimate $V(S_{t+1})$. The simplest form of TD-methods are called TD(0) and make the update in every time-step in the following way

$$V\left(S_t\right) \leftarrow V\left(S_t\right) + \alpha\left[R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right)\right].$$

While the update in MC has the return G_t as target, the update of TD(0) have the target $R_{t+1} + \gamma V\left(S_{t+1}\right)$. Since TD(0) partly is updated based on an existing estimate it also uses bootstrapping (as DP). In this way TD(0) are said to combine the sampling of MC-methods with the bootstrapping of DP and the advantages of both these methods can in this way be obtained. For example some applications have very long episodes (or no episodes at all for continuing tasks) and it will be too slow if we delay all learning until the end of the episode. It has also been showed that TD(0), for any fixed policy π, converge to v_π in the mean when using a sufficiently constant step-size parameter and with probability 1 if the step-size parameter decreases according to the usual conditions regarding stochastic approximation.

An important identity of the theory and algorithms of TD-learning is the term inside the brackets of the TD(0) update. This term is a sort of error between the estimated value of S_t and the better estimate $R_{t+1} + \gamma V\left(S_{t+1}\right)$. This is accordingly called the TD error and arises in many forms in reinforcement learning

$$\delta_t \doteq R_{t+1} + \gamma V\left(S_{t+1}\right) - V\left(S_t\right).$$

Since V does not change in MC-methods the MC error can moreover be written as the sum of TD errors.

2.6.2 Sarsa: On-policy TD control

In this section we go through the control problem by again following the pattern of GPI but in this case using TD methods for the policy evaluation part. The problem of balancing exploration and exploitation is very much relevant here as well and are again handled using on-policy and off-policy methods. In the following we present an on-policy TD control method called Sarsa.

As in MC-methods this method also start by learning a action-value function instead of a state-value function, namely $q_\pi(s,a)$ following current policy π must be estimated for all states s and actions a. This can be done using more or less the same approach as for estimating the value function with TD-methods but instead of states we consider state-action pairs. Moreover, the theorems that assure convergence for states values in TD(0) also holds for the corresponding update algorithm for action values. The update is made after every transition to a non-terminal state and can be written in the following way

$$Q\left(S_t, A_t\right) \leftarrow Q\left(S_t, A_t\right) + \alpha\left[R_{t+1} + \gamma Q\left(S_{t+1}, A_{t+1}\right) - Q\left(S_t, A_t\right)\right]. \qquad (8)$$

When making a transition from one state-action pair to the next, this update rule uses every element of the quintuple of events $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ (this is also the reason behind the name "Sarsa"). Constructing a control algorithm for the Sarsa policy evaluation is made in the same manner as for all on policy methods. To be precise, it continually estimate q_π for the policy π and at the same time change π towards a greedy policy with respect to q_π. According to this, the algorithm for a general Sarsa is given in pseudocode in algorithm 4 below.

Algorithm 4 Sarsa on-policy TD-control (ε-greedy).

```
Algorithm parameter: small $\varepsilon > 0$, step size $\alpha \in (0,1]$.
Initialize:
$Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$ except that
$Q(\text{terminal state, a}, \cdot) = 0$.

for each episode do
    Initialize $S$.
    Choose $A$ from $S$ using policy derived from $Q$ (in our case $\varepsilon$-greedy)
  for each step of episode until $S$ is terminal do
    Take action $A$ and observe $R, S'$.
    Choose $A'$ from $S'$ using policy derived from $Q$ (in our case $\varepsilon$-greedy)
    $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q\left(S', A'\right) - Q(S,A)\right]$
    $S \leftarrow S'; A \leftarrow A'$ ;

  end for
end for
```

Convergence of the Sarsa algorithm is dependent on the nature of the policy's dependence on Q. When, for example, using ε-greedy (or ε-soft policies in general), Sarsa converges to an optimal policy and action value function with probability 1 as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the greedy policy.

2.6.3 Q-learning: Off-policy TD control

In this section we describe the Q-learning algorithm (Watkins, 1989) which is an off-policy TD method that has been very influential in the development of reinforcement learning. Independent of the policy being followed, the Q-learning algorithm directly approximates the optimal action-value function in the following way:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right] \tag{9}$$

The policy still has effect on which state-actions that are visited and updated but as long as all state-action pairs continue to be updated, the algorithm converges correctly. Following this assumption and the usual stochastic approximation conditions on the step-size parameters, the learned action-value function Q converge to the optimal action value function with probability 1. The Q-learning algorithm is given in pseudocode below (algorithm 5).

Algorithm 5 Q-learning off-policy TD-control (ε-greedy).

```
Algorithm parameter: small $\varepsilon > 0$, step size $\alpha \in (0, 1]$.
Initialize:
$Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}^+$, $a \in \mathcal{A}(s)$ except that $Q(\text{terminal state, a}, \cdot) = 0$.

for each episode do
    Initialize $S$.
  for each step of episode until $S$ is terminal do
    Choose $A$ from $S$ using policy derived from $Q$ (in our case $\varepsilon$-greedy)
    Take action $A$ and observe $R, S'$.
    $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
    $S \leftarrow S'$
  end for
end for
```

In many ways, this algorithm is similar to the Sarsa algorithm explained earlier. The difference is that Q-learning update the value of Q when assuming a greedy policy action is followed in the next state even though it does not currently follow a greedy policy. For this reason, the Q-learning algorithm is, in contrast to Sarsa, considered to be an off-policy method.

3 Simulation of data

In this section we present how the data will be simulated. The simulation will be produced using the programming language R.

3.1 Compound distribution

An appropriate way to simulate claims data is by considering the total claim cost S to be given by a compound distribution. In accordance with Wüthrich (2020), a compound distribution are defined in the following way. Assume that N is the total number of claims. Moreover, assume that X_k, $k = 1, ..., N$ is the size of claim k and that the following three assumptions are fulfilled:

1) N is discrete stochastic variable which only takes values $0, 1, 2....$ For $N = 0$ no claims have been made and the total claim cost is accordingly 0.

2) $X_1, X_2, \ldots$ iid stochastic variables with distribution function $G(x) = P(X_k \leq x)$, namely the individual claim sizes are assumed to follow the same distribution.

3) N and $(X_1, X_2, ...)$ are independent.

If these assumptions are satisfied, the total claim cost S is given by a compound distribution which can be expressed in the following way

$$S = \sum_{k=1}^{N} X_k$$

and has the following properties

$$\begin{aligned} \mathbb{E}[S] &= \mathbb{E}[N]\,\mathbb{E}[X_1]. \\ \text{Var}[S] &= \text{Var}[N]\,\mathbb{E}[X_1]^2 + \mathbb{E}[N]\text{Var}[X_1]. \end{aligned}$$

3.2 Choice of claim distributions

To not make it unnecessary complicated and make use of the compound distribution properties, we assume that the risk exposure of every potential customer follow the same distribution. The claims are assumed to arrive according to a homogeneous Poisson process with intensity parameter α, namely the number of claims $Y_{j,t}$ for customer j up to and including time t are assumed to be Poisson$(\alpha \cdot t)$-distributed. The probability mass function of $Y_{j,t}$ is then given by

$$p_k = P(Y_{j,t} = k) = e^{-\alpha \cdot t}\frac{(\alpha \cdot t)^k}{k!}, \text{ for } k = 0, 1, 2, ...$$

and we have that $\mathbb{E}[Y_{j,t}] = \text{Var}[Y_{j,t}] = \alpha \cdot t$.

The claim sizes X_k are assumed to be exponentially distributed with parameter λ and accordingly they have probability density function

$$f(x) = \lambda e^{-\lambda x}, \text{ for } x \geq 0.$$

Moreover, it follows that $\mathbb{E}[X_k] = \frac{1}{\lambda}$ and $\text{Var}[X_k] = \frac{1}{\lambda^2}$.

Lastly, the number of claims and claim sizes are assumed to be independent of each other. The total insurance cost S_t in year t is then said to be given by a compound Poisson distribution, that is

$$S_t = \sum_{k=1}^{Y_t} X_k,$$

where $Y_t = \sum_{j=1}^{N} Y_{j,t}$ is the total number of claims year t. Using the properties of compound distributions we have that

$$\begin{aligned}
\mathbb{E}[S_t] &= \mathbb{E}[Y_t]\,\mathbb{E}[X_1] = N\alpha t\,\mathbb{E}[X_1]. \\
\text{Var}[S_t] &= \text{Var}[Y_t]\,\mathbb{E}[X_1]^2 + \mathbb{E}[Y_t]\text{Var}[X_1] = N\alpha t\,\mathbb{E}[X_1^2].
\end{aligned}$$

In all games considered in this thesis the claim distributions will be simulated with $\alpha = 0.5$ and $\mathbb{E}[X_1] = \frac{1}{5}$, meaning that the yearly mean claim cost for one customer is 0.1.

Modeling the times and sizes of claims in the above manner leads to the classical model in ruin theory, the so called Cramér-Lundberg model. The simplest form of the Cramér-Lundberg model assumes that the considered company has an initial surplus $u > 0$ and charges customers with a constant premium rate $p > 0$. This means that the total premium payments up to time t is $p \cdot t$. Moreover, the cumulative amount of claim payments up to time t is given by $S(t)$. The ruin event can then be expressed as the time of a negative surplus, namely

$$\text{Ruin } = \{U(t) < 0 \text{ for some } t > 0\}, \quad U(t) := u + pt - S(t).$$

Because of the constant income of premiums the surplus process $(U(t))_{t\geq 0}$ is increasing between jump times of $(S(t))_{t\geq 0}$. This means that ruin only can occur at these jump times $T_n > 0$. Since the time of the nth claim is given by $T_n = G_1 + \cdots + G_n$, where the G_ks are iid and exponentially distributed it follows that

$$\begin{aligned}
\text{Ruin } &= \{U(T_n) < 0 \text{ for some } n \geq 1\} \\
&= \left\{u + pT_n - \sum_{k=1}^{n} X_k < 0 \text{ for some } n \geq 1\right\}. \\
&= \left\{\sum_{k=1}^{n} (X_k - pG_k) > u \text{ for some } n \geq 1\right\}
\end{aligned}$$

The ruin event can in this manner be seen as a random walk with iid step sizes $(X_k - pG_k)$ exceeding the surplus μ at some time point (Mikosch, 2009). Given N_i customers insured at company I_i and initial capital reserve $R_{i,0}$ it follows that the reserve is decided by the dynamics

$$\mathrm{d}R_i(t) = (\mu_i + R_i(t))\mathrm{d}t + \sigma_i \mathrm{d}W_i(t) \tag{10}$$

where $(W_{i,t})_{t\geq 0}$ is a Wiener process, $\mu_i = N_i\,(p_i - \alpha\bar{x})$ and $\sigma_i^2 = N_i\alpha\bar{x^2}$. This can be seen as a so called diffusion approximation to the Cramer-Lundberg process. Note that $\bar{x} = \mathbb{E}[X_1]$ and $\bar{x^2} = \mathbb{E}[X_1^2]$ are used in order to simplify notation for the remainder of the thesis (Asmussen et al., 2019).

Furthermore, we will in the remainder of the thesis consider the competition between two insurance companies I_1 and I_2 and model the number of customers choosing company i explicitly using the approach in the following section.

3.3 Customer preferences

In all games we consider two different insurance companies I_1 and I_2 which offer an identical insurance product. Moreover, we assume that all N customers need and reduce their risk when buying insurance from any of the companies. In this way we focus on the market competition between the two companies and neglect the case of a customer rejecting insurance. If the insurance companies offer an identical insurance product, intuitively it feels reasonable that the customer will choose the company offering the lowest premium. However, when a given customer decides which company to buy the insurance from it may face different market frictions. For example it takes time and effort to search for the best company or to switch companies. Customers might also have different access to information and/or different abilities to process it. These type of frictions have been studied several times in economics during history. In this thesis we use a simple approach that can be both used for capturing different type of market frictions and customer preferences. We follow the setting in Asmussen et al. (2019) which has adopted the method from Hotelling (1929).

In this approach the market is said to be represented by an unit interval $[0, 1]$ and the two different insurance companies is placed at the end points, I_1 is placed at 0 and I_2 is placed at 1. A given customer j is then placed on a point h_j in the interval. This can be seen as a street where $a_1 = h_j$ describe the customers distance from I_1 and $a_2 = 1 - h_j$ the distance from I_2 (see picture below). The customer is then assumed to pay an extra cost $c \cdot a_1 = c \cdot h_j$ to buy the insurance from I_1 and $a_2 \cdot (1 - h_j) = c \cdot (1 - h_j)$ to buy the insurance from I_2. The parameter c can be viewed as a measure of the degree of market frictions. By using this marginal friction cost it is possible to model the additional cost for switching companies or buying insurance from a less preferred insurance company.

As stated earlier, we assume that all N customers reduce their risk when buying insurance from any of the companies and the question is therefore only which company they will choose. Given premiums p_1 and p_2 to be insured in I_1 an I_2 respectively we have that customer j will choose I_1 before I_2 if $p_1 - p_2 < c(1-2h_j)$ and in contrary I_2 if $p_2 - p_1 < c(2h_j - 1)$. In other words, if $h_j < 1/2(1-(p_1-p_2)/c)$ then I_1 is chosen and I_2 is chosen otherwise. If the equality holds the customer is indifferent but this will not be a problem since h_j is decided by a continuous distribution H. In our game setups, the market frictions that every potential customer will face are assumed to be $H \sim$ beta(a,b). The probability density function of a beta(a,b)-distribution is, for $0 \leq h \leq 1$, given by

$$f(h) = \frac{1}{\mathrm{B}(a,b)} h^{a-1}(1-h)^{b-1},$$

where

$$\mathrm{B}(a,b) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

is the so called beta function and Γ is the gamma function. Furthermore, we have that $\mathrm{E}[X] = \frac{\alpha}{\alpha+\beta}$ and $\mathrm{Var}[X] = \frac{ab}{(a+b)^2(a+b+1)}$. The beta distribution is a reasonable choice in this case since we want the market frictions only take values in $[0,1]$.

By modeling the market frictions in this manner, the expected portfolio sizes for the different companies can be written as a function of the premiums p_1 and p_2 in the following way.

$$\begin{aligned} N_1(p_1,p_2) &= NP(H < 1/2(1-(p_1-p_2)/c)) = NP(H<h) \\ N_2(p_1,p_2) &= N(1-P(H<1/2(1-(p_1-p_2)/c))) = N(1-P(H<h). \end{aligned} \tag{11}$$

With $H \sim$ beta(a,b), it follows that

$$\mathbb{P}(H<h) = B(h;a,b)/B(1;a,b) \text{ for } h \in (0,1),$$

where

$$B(h;a,b) = \int_0^h t^{a-1}(1-t)^{b-1}\ \mathrm{d}t.$$

Note that $B(1;a,b)$ is equal to the beta function $B(a,b)$. Using this, we have that the portfolio sizes can be expressed as

$$\begin{aligned} n_1(p_1,p_2) &= N\frac{B(h;a,b)}{B(a,b)} \\ n_2(p_1,p_2) &= N\left(1-\frac{B(h;a,b)}{B(a,b)}\right) \end{aligned} \tag{12}$$

The total number of customers needing insurance, N, can be treated in different ways. In real world it would be reasonable to let $N = N(t)$ vary between different years but since this will have little impact in the game theoretic part of the problem we consider we choose to treat N as constant, more exactly $N = 1000$ in all our game setups.

4 Game descriptions

In this section we will explain the main parts of the stochastic control problem that we consider. Usually the premium level of a certain insurance product is decided by underlying knowledge of the claim processes and make use of Generalised Linear Models. However, in this task we consider the other part of pricing, the market competition between different insurance companies. In order to do this we instead view the pricing as a sequential stochastic decision problem which is inspired by the stochastic differential game considered in Asmussen et al. (2019). In this article they analyze two competing insurance companies (one larger than the other) and find a premium strategy (taking market frictions into account) in which none of the companies are willing to change, a so called Nash equilibrium. More exactly, the Nash equilibrium is a concept of game theory in which the optimal result of a game is one where none of the players have any incentive to change from their chosen strategy given the competitors choice (Chen, 2021).

In this way the pricing of insurance can be seen as a decision process with a stochastic environment and we make use of the general idea to model the different games as a MDP. Furthermore, when modeling the different problems, we assume that the underlying model dynamics not are known for the learning agent and that we need to make use of model-free methods. Because of this, we will implement the Monte Carlo and Reinforcement Learning algorithms presented in the theory part of this thesis aiming to obtain the optimal premium strategy when facing the different situations. These results will then be compared to theoretical results of the problem. In all games we consider an episodic finite MDP with the following discrete time components.

Time horizon $t = 0, 1, ..., T$ in episode i where T is the terminal state of the episode.

State s_t in time t. The state will be either one-dimensional or two-dimensional in the different considered situations.

Action $A_t \in (A) = [\underline{p}, \overline{p}]$ in time t. The actions we consider are the different possible premium levels p_{1t} and p_{2t} which the insurance company can decide between in the different states.

Policy π. The policy decides the premium levels in different states.

Reward signal r_t. The actions made will lead to a reward r_t.

In this thesis we will consider two versions of the push and pull game, one where we assume an equal market and one where the market situation is related to the number of customers the company has.

4.1 Push and pull: Unchanged market

In the first game scenario we consider two insurance companies I_1 and I_2 offering the same insurance product and having capital reserve R_{1t} and R_{2t} respectively at time t, where $R_{10} > R_{20}$. Following Asmussen et al. (2019), we assume that the initially larger company I_1 wants to find an optimal way of pricing their product in order to maximize the difference $\Delta_t = R_{1t} - R_{2t}$ (push the competitor further away) and that the smaller company wants to minimize the same (pull closer to the competitor). Every year the market friction for every customer will be simulated from $H \sim \text{beta}(2, 2)$ with mean 0.5, namely none of the companies are assumed to have any market advantage in this game. To begin with, we will focus on the strategy for the larger company. Later on, this will be extended to analyzing the optimal strategy for both of the companies simultaneously. When analyzing both companies simultaneously we will also examine the results when I_1 has a market advantage. This will be constructed by instead simulating H from a beta$(2, 4)$-distribution with mean $1/3$. The purpose of finding the optimal strategies simultaneously is to see if the results of the algorithms match the strategy from which none of the companies earns by deviating from (Nash equilibrium) derived in Asmussen et al. (2019). The task can be formulated as an undiscounted episodic finite MDP.

We let the capital reserve decide the current economic status s_{1t} and s_{2t} of the different companies at time t. To simplify the state variable it could be reasonable to ignore the possibility of ruin. The motivation behind this is that in real life the initial capital reserve of the companies can be assumed to be so large that the probability of ruin is minimal or that investors would reinvest in the case of ruin. This simplification is also assumed in our problem and instead of using the capital reserve of the different companies as state variable we say that the current state s_t are decided by the difference of the economic status of the companies $d_t = s_{1t} - s_{2t}$. In this way we are able to reduce the number of states and make the pricing competition problem easier to survey and apply when implementing the methods considered in this thesis. The time horizon T is decided by the time until the terminal state is reached (game is over). In this case, the terminal state is reached when d_t is either 0 or 50, namely when the smaller company has equalled the status of the larger company or when the the larger company has increased the difference between the companies to 50. Related to this, we allow d_t to vary on the discrete values in the interval $[0, 50]$ and the considered system will accordingly contain 51 states.

In the game, we assume that both of the companies are assumed to have some prior knowledge of the underlying dynamics of the claims processes. The expected value of the individual total claim cost is 0.1. Related to this, the insurance company I_1 will use MC and reinforcement learning approaches to choose actions A_t among different premium levels

$$p_1 = (0.09, 0.1, 0.11, 0.12, 0.13, 0.14)$$

with the goal of maximizing the difference between the capital reserves of the two companies in the following T years. Since I_1 has 6 different actions to choose from in the states $1, 2, ..., 49$, the total number of state action pairs will be $49 \cdot 6 = 294$. To begin with, we will assume that the other company, I_2, takes a constant approach and set the premium to the expected value plus a 10 % safety margin. Namely, the premium will be 0.11 during the whole game. As described earlier, this will be extended to simultaneously using MC and reinforcement learning approaches for I_2 as well. Actions made will lead to rewards r_t. The reward will differ slightly depending on which method that is used. This will be described in more detail when the algorithms are implemented in the upcoming sections.

4.2 Push and pull: Changing market

In the second scenario the distribution of the friction variable H will be related to which company the customer currently is insured. The first year every potential customer again will face market frictions according to $H \sim \text{beta}(2, 2)$, meaning an initial equal market. However, for years $t = 2, ..., T$ the customers market friction will give favour to the company in which the customer currently is insured. If customer j currently is insured at I_1 the market friction will be distributed as $H \sim \text{beta}(2, 4)$ and if the customer currently is insured at I_2 it will be distributed as $H \sim \text{beta}(4, 2)$. Adjusting the market friction in this way is done to represent the cost of changing company. A possible pricing strategy might then be to win the market by first offering a lower premium and then make use of this advantage to win the game. A new market friction value $H = h_j$ for each customer j is generated every year. Again, we will focus on the larger company I_1 and the task is formulated as an undiscounted episodic finite MDP.

In this MDP, the state s_t in time t is given by a tuple (d_t, n_t), where d_t is the same as in the first scenario (reserve difference) and n_t contain information about the market situation. In order to not increase the number of states exceedingly, the market situation variable is given by the number of customers currently insured by I_1 rounded to hundreds. This is enough to know the full current market situation since we know that the total number of potential customers is $N = 1000$. The need for this extra variable comes from the fact that the number of customers (as explained above) affects the market friction variable and because of this influence the optimal strategy. Again, the terminal state is reached when d_t is either 0 or 50, namely d_t will vary in the interval $[0, 50]$. Since we have added the market situation variable the total number of states in the system we consider will be $51 \cdot 11 = 561$.

We assume that the companies initially have the same knowledge about the underlying dynamics of the claim processes as in the first scenario, namely that the expected value of the individual total claim cost is 0.1. Moreover, the insurance company I_1 has the possibility to choose among the premium levels

$$p_1 = (0.08, 0.085, 0.09, 0.095, 0.1, 0.105, 0.11, 0.115, 0.12)$$

and takes machine learning approaches with the goal of maximizing the difference between the capital reserves of the two companies in the following T years. Since I_1 has 9 different actions to choose from in the states

$$(1, 0), (1, 100), ..., (1, 1000), (2, 0), ..., (2, 1000), ..., (49, 0), ..., (49, 1000),$$

the total number of state action pairs will be $49 \cdot 11 \cdot 9 = 4851$. The other company, I_2, will in this case only be assumed to use a constant premium level. This time, the constant premium level will match the optimal one in the equal market scenario, namely 0.12. Actions made will lead to a reward r_t. The reward will differ slightly depending on which method that is implemented and will be described in more detail in the upcoming sections. The goal is to find the optimal premium strategy for the larger company in the considered game. In this way, we will see how the optimal strategy vary for different states.

5 Results

In this section we present the results of the considered games using different approaches. We begin with the push and pull problem where the friction variable is simulated from the same beta-distribution every year and then we examine when it is changed related to the number of customers currently insured at the different companies. First we will present theoretical results. Later on, we will present the results derived by dynamic programming and lastly, the reinforcement learning results.

5.1 Push and pull: Unchanged market

5.1.1 Theoretical solution

We start by analyzing the first scenario where the friction variable H is simulated from a beta$(2, 2)$ during the whole game procedure. Following a simplified version of Asmussen et al. (2019), the problem considers the current premium strategy to only be dependent of the current reserve difference d (Markovian) of

$D^\pi(t) = R_1^\pi(t) - R_2^\pi(t)$ between the controlled reserves $R_1^\pi(t)$ and $R_2^\pi(t)$. Using the fact that the uncontrolled reserves behaves like (10) it follows that $D^\pi(t)$ is a diffusion process, namely

$$\mathrm{d}D^\pi(t) = \mu^\pi\left(D^\pi(t)\right)\mathrm{d}t + \sigma^\pi\left(D^\pi(t)\right)\mathrm{d}W(t)$$

where

$$\begin{aligned}\mu^\pi(d) &= \mu_1\left(p_1^\pi(d), p_2^\pi(d)\right) - \mu_2\left(p_1^\pi(d), p_2^\pi(d)\right) + d,\\ \sigma^\pi(d)^2 &= \sigma_1\left(p_1^\pi(d), p_2^\pi(d)\right)^2 + \sigma_2\left(p_1^\pi(d), p_2^\pi(d)\right)^2,\end{aligned} .$$

$W = \left(W_1 - W_2\right)/\sqrt{2}$ is a Wiener process and following the problem formulation we have that $D(0) = R_1(0) - R_2(0) > 0$. Combining the equations (10) and (11) we have that

$$\begin{aligned}\mu_1\left(p_1, p_2\right) &= N\mathbb{P}\left(H < h\right)\left(p_1 - \alpha\bar{x}\right)\\ \sigma_1\left(p_1, p_2\right)^2 &= N\mathbb{P}\left(H < h\right)\alpha\overline{x^2}\\ \mu_2\left(p_1, p_2\right) &= N\left(1 - \mathbb{P}\left(H < h\right)\right)\left(p_2 - \alpha\bar{x}\right)\\ \sigma_2\left(p_1, p_2\right)^2 &= N\left(1 - \mathbb{P}\left(H < h\right)\right)\alpha\overline{x^2}\end{aligned}$$

Following a quite long and complicated discussion in Asmussen et al. (2019, p. 95-96), the considered problem can be simplified to the problems of point-wise maximization/minimization of the real valued ratio

$$\begin{aligned}k\left(p_1, p_2; d\right) = \frac{\mu^\pi(d)}{\sigma^\pi(d)^2} &= \frac{\mu_1\left(p_1^\pi(d), p_2^\pi(d)\right) - \mu_2\left(p_1^\pi(d), p_2^\pi(d)\right) + d}{\sigma_1\left(p_1^\pi(d), p_2^\pi(d)\right)^2 + \sigma_2\left(p_1^\pi(d), p_2^\pi(d)\right)^2}\\ &= \frac{N\mathbb{P}\left(H < h\right)\left(p_1 - \alpha\bar{x}\right) - N\left(1 - \mathbb{P}\left(H < h\right)\right)\left(p_2 - \alpha\bar{x}\right) + d}{N\mathbb{P}\left(H < h\right)\alpha\overline{x^2} + N\left(1 - \mathbb{P}\left(H < h\right)\right)\alpha\overline{x^2}}\\ &= \frac{N\mathbb{P}\left(H < h\right)\left(p_1 - \alpha\bar{x}\right) - N\left(1 - \mathbb{P}\left(H < h\right)\right)\left(p_2 - \alpha\bar{x}\right) + d}{N\alpha\overline{x^2}}\end{aligned}$$

From the last expression, we see that the denominator and also the term d does not depend on the premium strategy. Because of this, they can be omitted in optimization with respect to the premium. We also note that N cancels in the remaining term which explain why it is not necessary to let N fluctuate in this problem. That the optimal premiums strategy is independent of the current reserve difference make it possible to derive the theoretical optimal premiums p_1 by only considering the difference in drift. Using equation (12) we have that the value function to optimize then can be expressed in the following way

$$v\left(p_1, p_2\right) = N\frac{B\left(h; a, b\right)}{B(a, b)}\left(p_1 - \alpha\bar{x}\right) - N\left(1 - \frac{B\left(h; a, b\right)}{B(a, b)}\right)\left(p_2 - \alpha\bar{x}\right).$$

In the first case, we consider the premium level of p_2 to be constant and because of this, the value function will simplify to

$$v(p_1) = N\frac{B\left(h; a, b\right)}{B(a, b)}\left(p_1 - \alpha\bar{x}\right) - N\left(1 - \frac{B\left(h; a, b\right)}{B(a, b)}\right)\left(p_2 - \alpha\bar{x}\right).$$

Differentiating with respect to p_1 and setting to 0 we have that

$$\frac{dv(p_1)}{dp_1} = \frac{d}{dp_1} N \frac{B(h;a,b)}{B(a,b)}(p_1 + p_2 - 2\alpha\bar{x}) =$$

$$N\frac{B(h;a,b)}{B(a,b)} + \frac{N}{2c \cdot B(a,b)} \cdot h^{a-1}(1-h)^{b-1}(p_1 + p_2 - 2a\bar{x}) = 0, \cdot$$

$$\text{where} \quad h = \frac{1}{2}(1 - \frac{p_1 - p_2}{c})$$

In our problem, we have that $a = 2$, $b = 2$, $p_2 = 0.11$. For different friction marginal costs $c = (0.02, 0.04, 0.06)$ we get that the respective solutions are given by $p_1 = (0.1068543, 0.1134247, 0.1205201)$. The result is also illustrated in figure 1 where the different functions are plotted for p_1 in the interval $[0.09, 0.13]$.

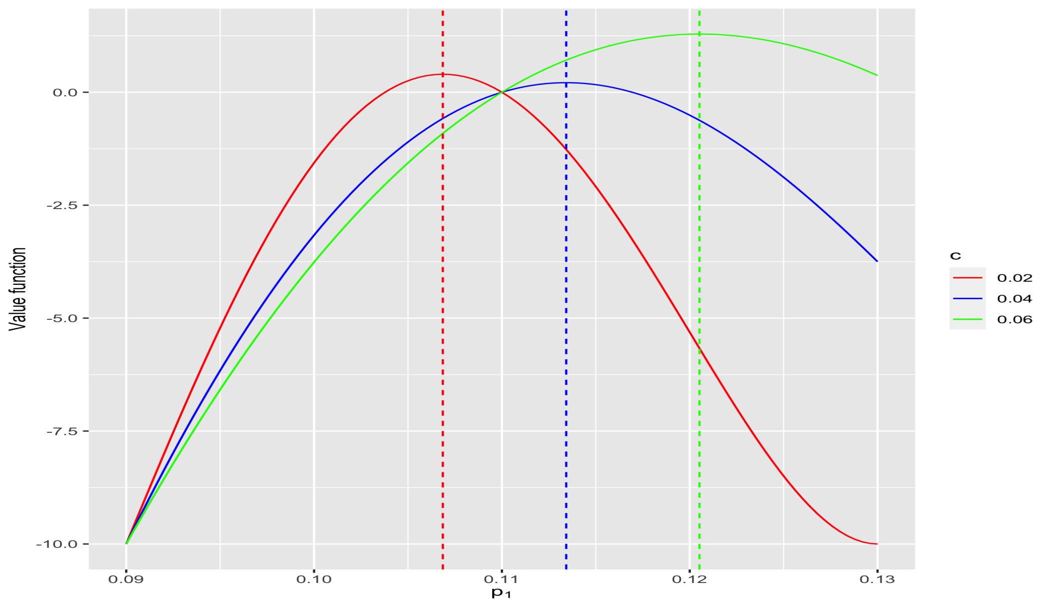

Figure 1: Theoretical optimal premium in push and pull game derived based on market friction H ~beta(2, 2), for different marginal friction costs $c = (0.02, 0.04, 0.06)$ and constant premium level $p_2 = 0.11$ for the competing insurance company I_2.

This shows how the optimal premium level vary for different values of the marginal friction costs. When c is higher, the number of customers choosing the company is more affected by the friction and less of the premium level. Because of this, it is optimal to charge a higher premium in order to maximize revenue. When $c = 0.04$, the optimal premium level is close to 0.11 which means that it is close to the premium chosen by I_2. Since the strategy of I_2 is close to optimal in this case it will be hard to find a better suited premium level and the value function has its maximum close to 0. When implementing the algorithms in the upcoming sections we will assume that $c = 0.06$, namely the theoretical optimal premium level derived is 0.1205201. In this way we are able to see if the considered methods are able to take the market friction into consideration and adapt the premium to this higher level.

Next, we consider the case where the smaller company simultaneously tries to find the optimal premium. This relates to finding the strategy where none of the companies have any incentive to change from their chosen strategy given the competitors choice (Nash equilibrium). In Asmussen et al. (2019) the following theorem is stated:

Theorem 5.1. *Let $H \sim beta(a, b)$ and let m_β be the median of $beta(a, b)$. Then a Nash equilibrium exists at*

$$p_1^* = \alpha\bar{x} + \frac{\rho c}{2}\left(\frac{B(a,b)}{m_\beta^{a-1}\left(1-m_\beta\right)^{b-1}} + 1 - 2m_\beta\right),$$

$$p_2^* = \alpha\bar{x} + \frac{\rho c}{2}\left(\frac{B(a,b)}{m_\beta^{a-1}\left(1-m_\beta\right)^{b-1}} - 1 + 2m_\beta\right),$$

provided that p_1^*, p_2^* are in a feasible region and that

$$-4 \leq \left(\frac{a-1}{m_\beta} - \frac{b-1}{1-m_\beta}\right)\frac{B(a,b)}{m_\beta^{a-1}\left(1-m_\beta\right)^{b-1}} \leq 4. \tag{13}$$

In equilibrium, $n_1\left(p_1^*, p_2^*\right) = n_2\left(p_1^*, p_2^*\right) = N/2$.

The feasible region and parameters a and b in equation (13) for which the theorem holds is analyzed thoroughly in the article by Asmussen et al.. It turns out that, in the range $0 \leq a, b \leq a$, the conditions for the theorem are only violated if a or b is very close to 0. In particular, the theorem holds for the distributions considered in this thesis. For the sake of keeping a common thread, these results will not be described further but, for the interested reader, a more extensive discussion can be found in Asmussen et al. (2019, p. 98). In the first case $H \sim$ beta$(2, 2)$, the median m_β is 0.5 and $c = 0.06$. Following theorem 5.1, the Nash equilibrium premiums are the same for both companies and they can be calculated to 0.12. In the second case, we assume $H \sim$ beta$(2, 4)$ and the median m_β is 0.25. In this case the Nash equilibrium premiums are computed to $p_1^* = 0.1292$ and $p_2^* = 0.0992$. Since company I_1 has a market advantage in this case, it should charge a higher premium. The results are also illustrated by the surfaces in figure 2 which shows the value of $k\left(p_1, p_2; d\right)$ when $d = 25$. The points illustrate the Nash equilibrium in the two different cases.

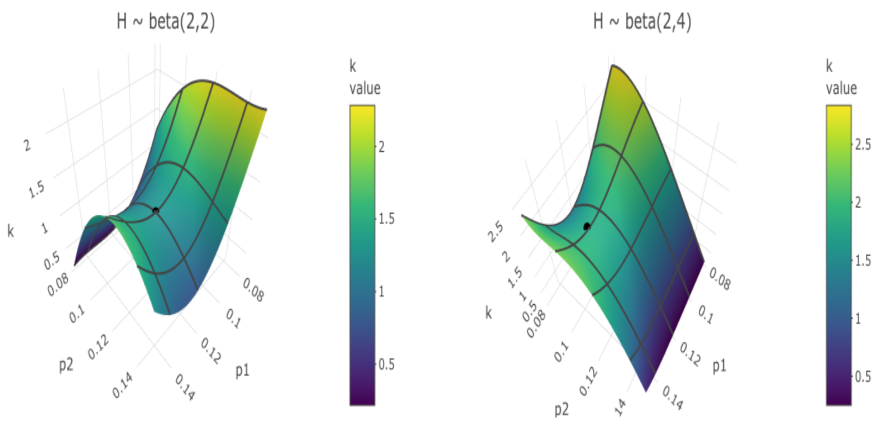

Figure 2: Premium levels p_1 and p_2 and value of $k\,(p_1, p_2; d)$ in push and pull game, derived based on market friction H ~beta$(2, 2)$ (left plot) and H ~beta$(2, 4)$ (right plot), marginal friction cost $c = 0.06$ and $d = 25$. The black point shows the derived Nash equilibrium.

As explained earlier, the theoretical solutions derived above assume that the optimal strategy is independent of the current reserve difference. This means that the derived premium should be taken no matter the current state. However, in the considered game, another premium level could be less risky than the optimal premium level and because of this preferable in states with small reserve difference to reduce the risk of losing. This will be examined more in the next section where we take a DP approach that more accurately describe the dynamics of the considered game.

5.1.2 Dynamic programming

As described in section 3.4 we need to know the model of the environment to solve the problem using dynamic programming. The environment is fully described by the state transition probabilities and to get reliable estimates of these we will take a simulation approach. For large enough iterations, the claim processes should behave as the theoretical compound distribution described earlier. Moreover, in each iteration, we simulate a new market friction parameter h_j for every customer and for large enough N, the behaviour of this parameter should correspond to the desired theoretical beta-distribution. For all states $s_t \in [0, 50]$ and all actions $p_1 = (0.09, 0.1, 0.11, 0.12, 0.13, 0.14)$ we will simulate 10000 transitions to the next state $s_{t+1} = s'$. The state transition probability $P(s_{t+1} = s' \mid s_t = s, A_t = p_1)$ is then estimated as the percentage of transitions from s to s'. An extract of the estimates is shown in table 1 in appendix B.

In order to solve the problem we implement the value iteration algorithm (2). The results after $1, 2, 10, 50, 100$ iterations are presented in figure 3 below.

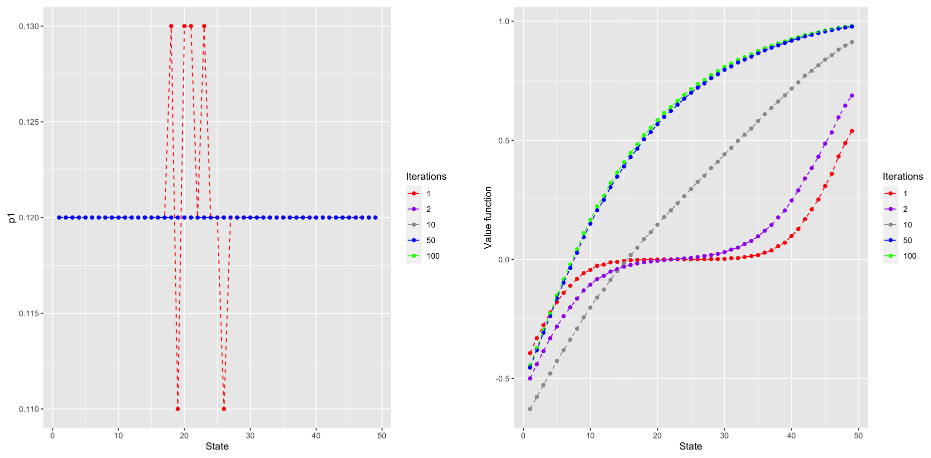

Figure 3: Value iteration algorithm. The optimal premium policy (left plot) and state-value function (right plot) for push and pull game, with market friction $H \sim$beta$(2, 2)$, for marginal friction costs $c = 0.06$ and constant premium level $p_2 = 0.11$ for the competing insurance company I_2.

In the left plot of figure 3 we can conclude that the value iteration algorithm converges to the optimal premium level $p_1 = 0.12$ in only 2 iterations. However, in the right plot, we see that the value function still gets updated and that it takes around 50 iterations until it converges.

5.1.3 RL-methods

In this part we implement some of the model-free RL-methods that we presented in section 3.5 and 3.6. We compare the results with the theoretical and dynamic programming solutions. First we implement the ε-greedy on-policy first-visit MC control algorithm (3). In this algorithm the reward $+1$ is given to all first visit state-actions during an episode if the episode ends in state 50 or higher (the larger company wins). If the episode ends in 0 or lower (smaller company equals the economic status of the larger company), the reward -1 will be given to all first visit state-actions.

We also implement the Sarsa TD(0)-learning (4) and Q-learning algorithm (5) with ε-greedy as decision rule. In these methods the action value function are updated every year of the episode according to the equations (8) and (9) respectively. The immediate reward r_t in the equations will be given as the difference of the companies revenue the given year. This means that the reward will correspond to the actual value that the state approach the goal. When ending in state 50 or higher (winning) an extra reward of +25 will be given to the last state-action pair and when ending in state 0 or lower (losing) a punishment reward of -25 will be given to the last state-action pair. In all methods we choose $\varepsilon = 0.1$ and $\alpha = \frac{1}{N_s(a)}$, where $N_s(a)$ is the number of times the action a has been performed in state s. Since we want to examine the premium level for all reserve differences, we do not assume any specific starting state. Instead, we will sample the first state of every episode from all possible non-terminal states, namely $1 - 49$. In some of the games this does not really matter but in others, we see that this is essential to keep visiting and updating some of the states.

The result when assuming $p_2 = 0.11$ is shown in figure 4 below. The left part of the figure illustrates the produced optimal policy for different states after $j = 10000$ episodes and the right part shows the scaled values of the action-value function for all states when following this policy. In figure 9 and 10 in appendix B, the results of the methods after $j = 100$ and $j = 1000$ episodes are also illustrated.

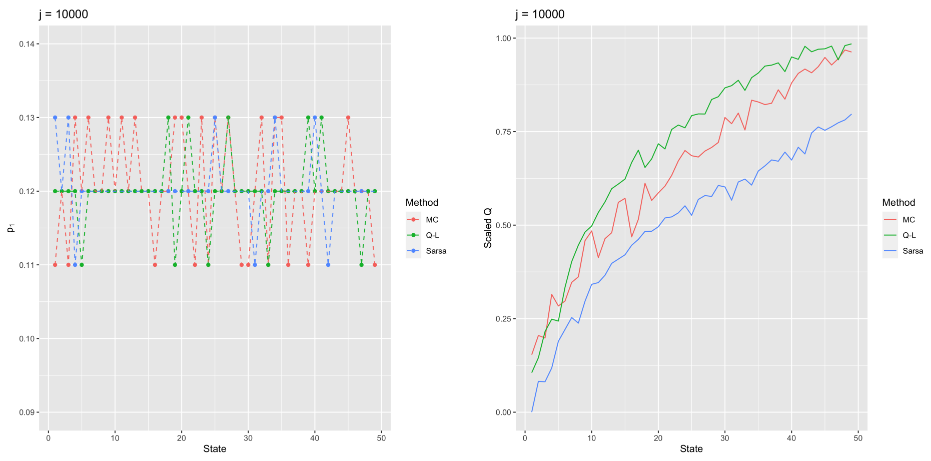

Figure 4: MC, Sarsa and Q-learning: The optimal premium policy (left plot) and action-value function (right plot) for push and pull game, with market friction $H \sim$beta$(2, 2)$, for marginal friction costs $c = 0.06$ and constant premium level $p_2 = 0.11$ for the competing insurance company I_2.

In the left part of the figure we conclude that, after 10000 episodes the optimal premium is 0.12 for most states in all methods. This also match the theoretical and DP results. In the right part, we also see that all reinforcement learning algorithms provide similar results of the action value function. As expected, when the reserve difference increase, the larger company has a higher possibility of winning the game. The action value function also has a similar shape as the action value function achieved with DP. We also conclude that Q-learning seem to be the method that converges the fastest against the theoretical results. However, we notice in both plots that the optimal premium and the corresponding action value function still fluctuate, even for states that are close to each other and should provide similar results. This suggests that, even though the algorithm converges in the direction of the optimal result, it still struggles to converge to the policy for all states. One reason behind this is that some state-actions not are performed as frequently as others in the considered game.

The are several ways to get faster convergence and improve the unstable results. For example, we could use an appropriate kernel function to update the action value function for several states simultaneously. Another simple approach is to just aggregate states that are close and because of this should be related. Of course, using this procedure has its limitations but it is fairly reasonable in this case since close states (reserve differences) should have similar rewards. We will make use of this approach in the game where we consider the case of a changing market but for now we settle for the results obtained above.

Instead, we examine the case where both companies implement the reinforcement learning methods to decide premiums in an optimal way. As discussed earlier, the goal of the larger company is to increase the reserve difference to 50 and the goal of the smaller company is to close the gap (reserve difference is 0). Because of this, the rewards for the larger company will be the same as before and the rewards for the smaller company will be the same but in the opposite direction (-reward). The extra reward when losing or winning the game will be modified in the same way.

We will examine this both when the market is equal and when we assume a market advantage for the larger company. Since the results obtained with the different algorithms are similar and Q-learning seem to be the best, we settle for only showing the results from this algorithm in this task. In figure 5 the optimal policies after 10000 episodes are illustrated for the two different cases.

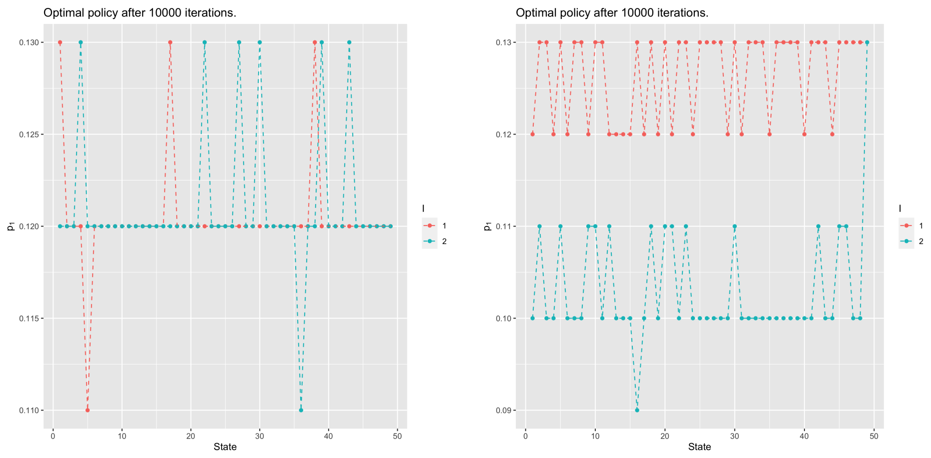

Figure 5: Simultaneous optimal policy with Q-learning when assuming market friction H ∼beta$(2, 2)$ (left plot) and H ∼beta$(2, 4)$ (right plot) for push and pull game with marginal friction costs $c = 0.06$.

From figure 5, we conclude that in both cases, the optimal premiums converge in the direction of the optimal theoretical premiums in Nash-equilibrium. In the case where we assume an equal market, the results are, despite a few points, very clear and the premium is 0.12 (Nash-equilibrium) in most cases. When the larger company has a market advantage, $p_1 = 0.13$ and $p_2 = 0.1$ (Nash-equilibrium) for most cases but we also notice some deviations, especially for the smaller company. When implementing the method, we also noted some computational difficulties in this case. Because of the market advantage, it leads to less observations in the lower reserve difference states and the corresponding actions. In this particular game the arbitrary sampling of the first state in every episode is essential to keep exploring some of the states.

In figure 11 in appendix B, the corresponding action value functions are also illustrated. In this figure, the fact that both of the companies have equal but opposite action value functions is illustrated in a clear way.

5.2 Push and pull: Changing market

5.2.1 Theoretical solution

We take the same approach as in the theoretical solution when beta does not change. In this case we have that the value function for the N_1 customers belonging to I_1 will be

$$v(p_1; I_1) = N_1 \frac{B\,(h; a, b)}{B(a, b)} (p_1 - \alpha\bar{x}) - N_1 \left(1 - \frac{B\,(h; a, b)}{B(a, b)}\right) (p_2 - \alpha\bar{x}) \,,$$

with $a = 2$ and $b = 4$. For the N_2 customers belonging to I_2 the value function will be

$$v(p_1; I_2) = (1000{-}N_1)\frac{B\,(h; a, b)}{B(a, b)}\,(p_1 - \alpha\bar{x}) {-} (1000{-}N_1)\left(1 - \frac{B\,(h; a, b)}{B(a, b)}\right)(p_2 - \alpha\bar{x})\,,$$

with $a = 4$ and $b = 2$. Adding these two we get the value function for all customers

$$v(p_1) = N_1\frac{B\,(h; 2, 4)}{B(2, 4)}\,(p_1 - \alpha\bar{x}) - N_1\left(1 - \frac{B\,(h; 2, 4)}{B(2, 4)}\right)(p_2 - \alpha\bar{x}) +$$
$$(1000 - N_1)\frac{B\,(h; 4, 2)}{B(4, 2)}\,(p_1 - \alpha\bar{x}) - (1000 - N_1)\left(1 - \frac{B\,(h; 4, 2)}{B(4, 2)}\right)(p_2 - \alpha\bar{x})$$
$$\text{where} \quad h = \frac{1}{2}(1 - \frac{p_1 - p_2}{c}).$$

For $p_2 = 0.12$, $c = 0.06$ and $N_1 = (100, 200, 300, 400, 500, 600, 700, 800, 900)$, we get that the respective solutions are given by

$$p_1 = (0.1087, 0.1130, 0.1174, 0.1212, 0.1240, 0.1260, 0.1275, 0.1286, 0.1294).$$

This is also illustrated in figure 6 below where the different functions are plotted for p_1 in the interval $[0.08, 0.15]$. The points illustrates the maximums in the different cases.

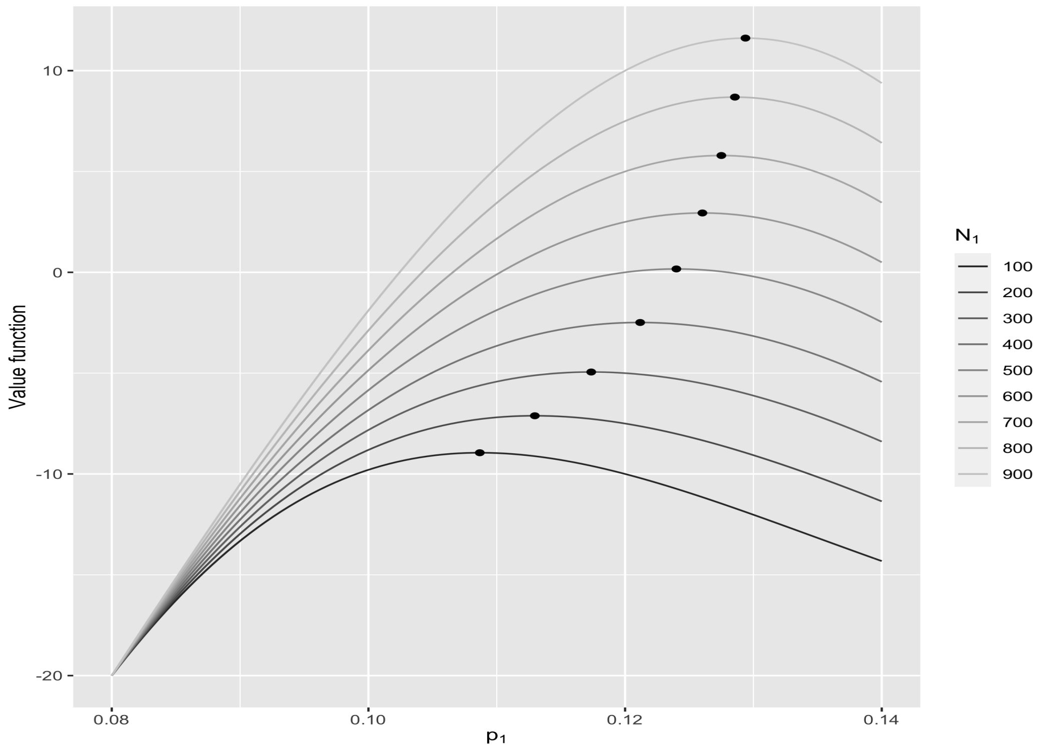

Figure 6: Theoretical optimal premium derived based on market friction $H \sim$beta$(2, 4)$ or $H \sim$ beta$(4, 2)$ depending on which company the customer currently belongs, $c = 0.06$ and constant premium level $p_2 = 0.12$ for the competing insurance company I_2.

In figure 6 it gets clear that the desired behaviour is achieved. When I_1 has more customers, the company has a market advantage and can make use of this to get higher revenues and increase the reserve difference. From these derived theoretical optimal premiums, we also conclude that for more customers N_1, the company can afford and gain from charging a higher premium without losing to many customers. However, this is just the optimal theoretical solution if we want to maximize the reserve difference the given year. In our game setup, we want to maximize the revenue in the long run and then it could be more optimal to charge a lower premium to gain a market advantage in the following years. In the next section, we analyze this further by estimating the dynamics of the environment and take a dynamic programming approach in the considered game.

5.2.2 Dynamic programming

In order to get reliable estimates of the environment model we take the same approach as in the first problem. Again we simulate the transitions 10000 times. For the interested, an extract of the estimates is shown in table 2 in appendix B.

To solve the problem we again implement the value iteration algorithm (2). The results after $1, 2, 10, 50, 100$ iterations are presented in figure 7 below.

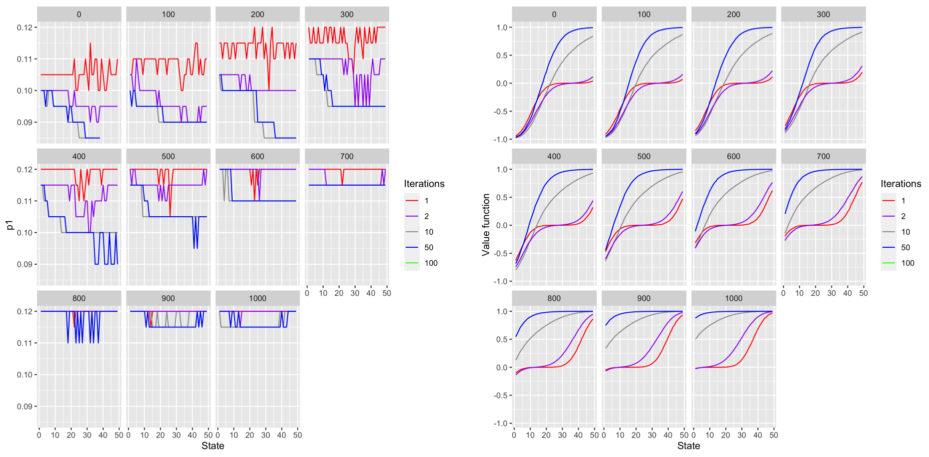

Figure 7: Value iteration algorithm. The optimal premium policy (left plot) and state-value function (right plot) for push and pull game, with market friction H ~beta$(2, 4)$ or H ~beta$(4, 2)$ depending on which company the customer currently belongs, $c = 0.06$ and constant premium level $p_2 = 0.12$ for the competing insurance company I_2.

In the left plot of figure 7 we see that the value iteration algorithm converges to some optimal premium level in the different states in about 50 iterations (the blue line). We see that this also is the case for the value function. From the figure, we can also distinguish that the optimal premium level depends on the current state. For small N_1 and larger reserve difference it is better to offer a very low premium to attract more customers and increase the reserve difference in the upcoming years. However, as the current reserve difference gets lower, the optimal premium increases. The reason behind this is that the company no longer can afford to take a year of losses because of the risk of losing the game.

For larger N_1 it is in general, as was expected from the theoretical solutions, better to offer a higher premium in order to maximize the reserve difference. This also makes the pattern between reserve difference and premium for large N_1 $(700, 800, 900, 1000)$ to be less clear compared to when N_1 is small. It is optimal to charge a a higher premium, both for smaller current reserve difference and very high reserve difference. In the middle states we note some tendencies that premiums offered should be a bit lower. The reason for this is probably that the game still will go on for some years before winning and the company cannot risk losing the current market advantage for the future years. For higher reserve difference the game is won in a short period and losing some customers is not that big of a deal. Of course, in a real life insurance scenario, the competition is ongoing and this sharp line of losing and winning the game does not really apply. Losing customers is not good for the years after the reserve difference has increased to over 50 either but since we do not consider these years in this game, this explains the result obtained by the algorithm. To connect this particular situation to real-life we could see it as impossible for the smaller company to catch up when the reserve difference is over 50.

5.2.3 RL-methods

The state space of this problem has increased tremendously in comparison to the first problem. Since the algorithms struggled to converge for all the states separately even in the first game, we directly consider RL-method approaches that make it possible to update more than one state at a time. We don't want to over-complicate things and choose the simple approach of to aggregating the reserve differences close to each other (5 and 5). This approach end up with the following 99 states

$$\begin{gathered}((1,2,3,4,5),0),((6,7,8,9,10),0),...,((46,47,48,49),0),...,\\((1,2,3,4,5),100),((6,7,8,9,10),100),...,((46,47,48,49),100),....\\((1,2,3,4,5),1000),((6,7,8,9,10),1000),...,((46,47,48,49),1000)\end{gathered}$$

We settle for only implementing the Q-learning algorithm and the rewards are structured in the same way as in first problem. The results after 50000 episodes are shown in figure 8 below.

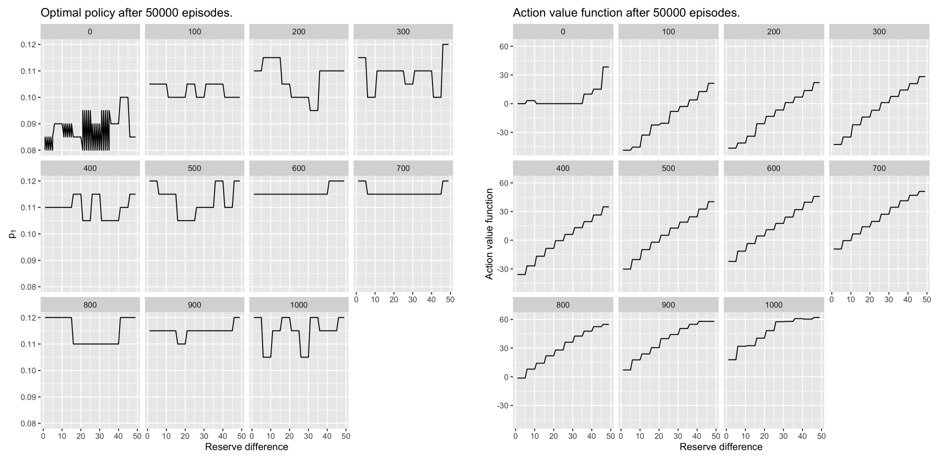

Figure 8: Q-learning with aggregated states. The optimal premium policy (left plot) and action-value function (right plot) for push and pull game, with market friction H ∼beta$(2, 4)$ or H ∼beta$(4, 2)$ depending on which company the customer currently belongs, $c = 0.06$ and constant premium level $p_2 = 0.12$ for the competing insurance company I_2.

From the figure, we see some patterns that are similar to the DP-results. For example, except for the case when $N_1 = 0$, it is better to offer a low premium when $N_1 \leq 500$ and the current reserve difference is large enough. The figure also indicate that, for larger N_1, the optimal premium increases for larger reserve differences. This also match the pattern of the DP results.

That N_1 provides so unstable results can be explained by the fact that these states not are entered as frequently. Actually, most state-action pairs when $N_1 = 0$ are only updated one time. This could be adjusted, by for example considering another decision rule. However, since this case is very unrealistic and not that interesting we choose to not examine this further.

6 Discussion and conclusions

In this thesis we have analyzed an insurance market where two different companies offer the same product and compete for customers by choosing premium level and taking market frictions into consideration. More precisely, we have examined the push and pull game where we consider the two companies I_1 and I_2 with initial capital reserves $R_1(0) > R_2(0)$ (one larger than the other) and the goal of the larger company is to push the smaller company further away. By modeling the market frictions explicitly and assuming that the smaller company use a constant premium approach, theoretical optimal premiums for the larger company have been derived in two different push and pull scenarios. In one of the scenarios, we assume that the market friction is given by a beta$(2, 2)$-distribution every year for every customer j. In the other scenario we let the market friction variable give favour to the company which the customer currently is insured. The market friction will then be simulated from a beta$(2, 4)$ if it belongs to the larger company I_1 and a beta$(4, 2)$ if it belongs to I_2. We have also computed the Nash equilibrium premiums for two games with different market conditions, based on results in Asmussen et al. (2019).

After deriving theoretical solutions, we have considered the same push and pull games played until either the larger company has increased the reserve difference to 50 or the smaller company has equalled the larger company. In order to derive optimal pricing strategies in the different situations, we gain complete knowledge of the environment using simulation and the problems then can be solved using dynamic programming. In both games, the DP-method converge quickly when we have a well simulated model of the environment and the results correspond to the derived theoretical results.

In real-life, the full model of the environment is not known and we need to rely on other methods. Because of this, we have implemented the model-free learning methods MC, Sarsa and Q-learning to see if these methods converge to similar results as those achieved with DP. Analyzing the competition part of pricing using reinforcement learning was also one of the main objectives in this thesis. The results show that the methods converge, at least, in the direction of the DP results. The results when simultaneously deciding the optimal policy for both companies also match the theoretical Nash equilibrium derived. Indeed, this shows that it is possible to model the problem using reinforcement learning and gain knowledge from viewing the problem in this manner. However, in the particular methods implemented in this thesis, we note some struggles with convergence. For reinforcement learning methods to be applicable in the insurance industry it requires much faster and appropriate on-policy updates. Fortunately, there are many helpful extensions of these methods that could be used to achieve this if implemented correctly.

The results also show that modeling the competition explicitly using market friction costs, can lead to different insights compared to modeling fair pricing with a classical approach. For example, in this case, how to balance the contradicting tasks of attracting customers and increasing marginals in order to maximize/minimize the reserve difference between the companies. In the first considered game when the second company offers a constant premium level 0.11, the results actually suggest that the company should charge a higher premium to maximize the reserve difference for these particular market friction costs. The result itself, namely that we can use the imperfect market conditions and gain from charging higher premiums is not very surprising. Also, this way of maximizing profits at the expense of customers could be questionable for several reasons.

In the second game, the results show that the optimal premium strategy differs between states. When the current reserve difference is large, it is better to charge a lower premium to attract customers and gain a market advantage for the future. In particular, this is the case when the current number of customers is low and the company needs an increase in this area. For states where the current reserve difference is low, the risk of offering a low premium is substantial. The larger company can not afford a year of losses because of the risk of losing the game. Because of this, it is more appropriate to choose a a higher and less risky premium for states where the reserve difference is lower. From the results, we could also distinguish that we in general should charge a higher premium when having the market advantage. This is also in agreement with the results in Assmussen et al. (2019).

To summarize, the results provided show that it is indeed possible to view the problem of pricing as a sequential decision process and with an appropriate model of the market frictions we can gain insights from this way of viewing the problem. Of course, the simplified situations considered in this thesis have limited applicability in real world but the problem and solution methods can be extended in several ways so that it can be applied to more complex tasks. For example, it could be possible to combine the more traditional methods of pricing with machine learning approaches to evolve premium strategies for different customer segments. Of course, the implementation of powerful data-driven methods could also lead to risks. For example, price discrimination and companies, at the expense of customers, only setting the premium in order to maximize revenue. However, if implemented correctly, the premiums could be decided more fairly by taking more individual data into consideration.

7 Future work

The main purpose of this paper was to see if the competition between companies could be modeled using reinforcement learning in a simplified setting. The study have many drawbacks and limitations. For example, we only consider a small set of premiums to choose from and the state variable is very simplified for computational reasons. However, both the problem formulation and the solving methods could be adapted and extended in several ways to be more reasonable in a real-life insurance context. It would, for example, be interesting to implement the methods in a more advanced model where the claim processes is allowed to differ between individuals. The methods could then make use of a more complex state that contain more information about the specific customers. In this way, information from classical pricing could be combined with reinforcement learning to make updates in a reasonable way.

However, since this would make the state space increase extensively, the models would have to be extended with some appropriate function approximation that make it possible to update the value function for several states simultaneously. With appropriate update and pricing functions, it would also be interesting to examine reinforcement learning against other pricing methods in a more on demand dynamic pricing setting. Instead of using a few number of actions and states as we have done in this thesis, these could be given by some appropriate distribution which is updated according to some decision rule. The extensions are many and the results could be improved in several ways.

Appendix A

Policy improvement theorem

The proof is fairly intuitive to understand. We start from the result $v_\pi(s) \leq q_\pi(s, \pi'(s))$ and expand $q_\pi(s, \pi'(s))$ using (6) and applying $v_\pi(s) \leq q_\pi(s, \pi'(s))$ until we end up with $v_{\pi'}(s)$, namely the value function following the new policy.

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

Extended to the ε-greedy case:

Conditions of the policy improvement theorem holds since for any $s \in \mathcal{S}$:

$$
\begin{aligned}
q_\pi(s, \pi'(s)) &= \sum_a \pi'(a \mid s) q_\pi(s, a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1 - \varepsilon) \max_a q_\pi(s, a) \\
&\geq \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + (1 - \varepsilon) \sum_a \frac{\pi(a \mid s) - \frac{\varepsilon}{|\mathcal{A}(s)|}}{1 - \varepsilon} q_\pi(s, a) \\
&= \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) - \frac{\varepsilon}{|\mathcal{A}(s)|} \sum_a q_\pi(s, a) + \sum_a \pi(a \mid s) q_\pi(s, a) \\
&= v_\pi(s),
\end{aligned}
$$

where the inequality follows because the sum is a weighted average of non-negative weights summing to 1 and because of this it must be less or equal to the largest number averaged (Sutton and Barto, 2018).

Appendix B

Plots

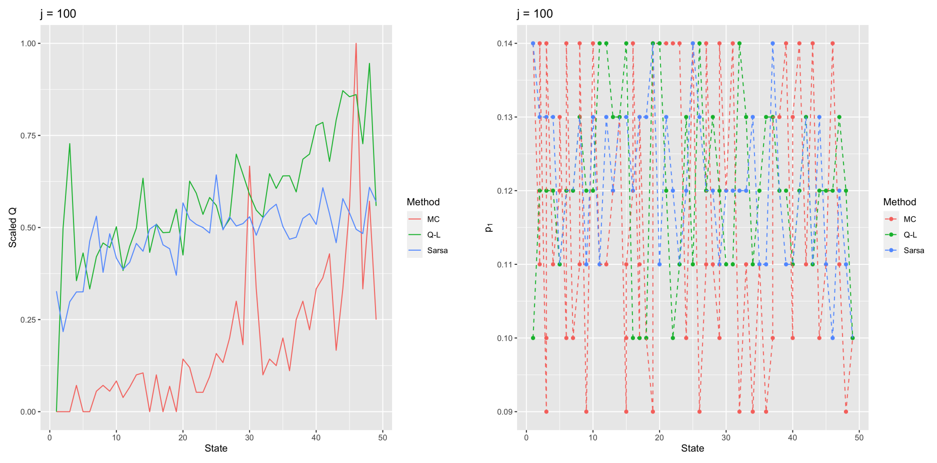

Figure 9: MC, Sarsa and Q-learning: The optimal premium policy (left plot) and action-value function (right plot) for push and pull game after 100 episodes, with market friction H ∼beta$(2, 2)$, for marginal friction costs $c = 0.06$ and constant premium level $p_2 = 0.11$ for the competing insurance company I_2.

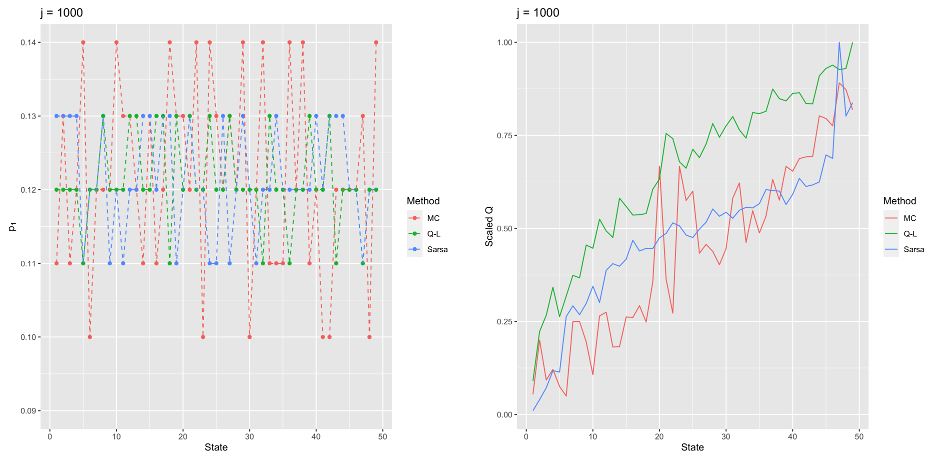

Figure 10: MC, Sarsa and Q-learning: The optimal premium policy (left plot) and action-value function (right plot) for push and pull game after 1000 episodes, with market friction H ∼beta$(2, 2)$, for marginal friction costs $c = 0.06$ and constant premium level $p_2 = 0.11$ for the competing insurance company I_2.

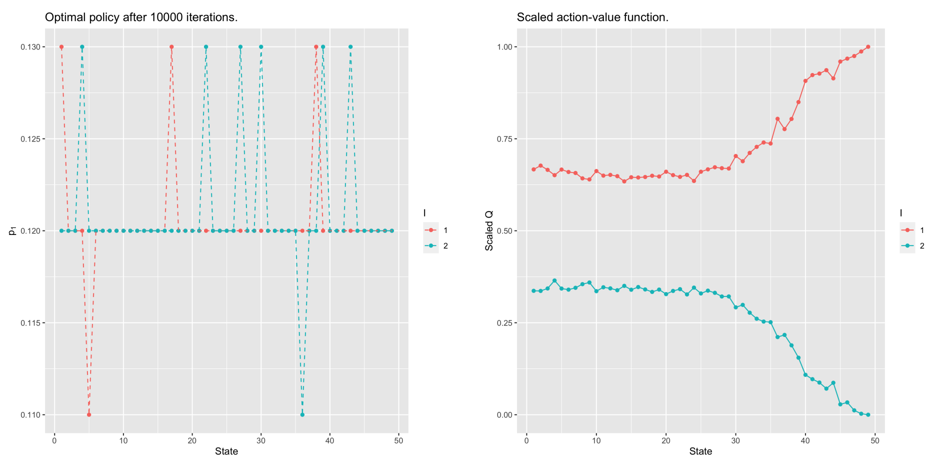

Figure 11: Simultaneous optimal policy with Q-learning when assuming market friction H ∼beta$(2, 2)$ (left plot) and H ∼beta$(2, 4)$ (right plot) for push and pull game with marginal friction costs $c = 0.06$.

Tables

Table 1: Simulated state transition probabilities when H ∼beta$(2, 2)$.

$R_1 - R_2$	Next $R_1 - R_2$	p_1	%
1	0	0.09	0.933
1	0	0.1	0.700
1	0	0.11	0.464
1	0	0.12	0.394
1	0	0.13	0.444

Table 2: Simulated state transition probabilities when H ∼beta$(2, 4)$ or H ∼beta$(2, 4)$ depending on which company customer j is insured.

$R_1 - R_2$	N_1	Next $R_1 - R_2$	Next N_1	p1	%
1	0	0	100	0.12	0.0009
1	0	0	200	0.125	0.66
1	0	0	200	0.12	0.969
1	0	0	300	0.105	0.018
1	0	0	300	0.11	0.942

References

[1] Asmussen, Søren, Bent Jesper Christensen, and Michael Taksar, (2013). Portfolio size as function of the premium: modeling and optimization. Stochastics 85 (4), 575–588.

[2] Asmussen, Søren, Bent Jesper Christensen, and Julie Thøgersen. (2018). *Stackelberg Equilibrium Premium Strategies for Push-Pull Competition in a Non-Life Insurance Market with Product Differentiation.* Risks, 7(2): 49. https://doi.org/10.3390/risks7020049.

[3] Asmussen, Søren, Bent Jesper Christensen, and Julie Thøgersen. (2019). *Nash equilibrium premium strategies for push–pull competition in a frictional non-life insurance market.* Insurance: Mathematics and Economics, 87, 92-100. In press. https://doi.org/10.1016/j.insmatheco.2019.02.002.

[4] Chen, James (2021). *Investopedia: Nash equilibrium.* Available at: https://www.investopedia.com/terms/n/nash-equilibrium.asp

[5] Krasheninnikova, Elena, Javier García, Roberto Maestre, and Fernando Fernández. (2019). *Reinforcement learning for pricing strategy optimization in the insurance industry.* Engineering Applications of Artificial Intelligence, 80, 8-19. 10.1016/j.engappai.2019.01.010.

[6] Mikosch T. (2009) *Ruin Theory. In: Non-Life Insurance Mathematics.* Universitext. Springer, Berlin, Heidelberg.

[7] Sutton, Richard S., and Andrew G. Barto. (2018). *Reinforcement Learning: An Introduction, second edition.* Cambridge, MA: The MIT Press.

[8] Thøgersen, Julie, (2016). *Optimal premium as a function of the deductible: customer analysis and portfolio characteristics.* Risks 4 (4), 42.

[9] Wüthrich, M. V. (2020). *Non-Life Insurance: Mathematics and Statistics.* Department of Mathematics, ETH Zurich. Available at SSRN: https://ssrn.com/abstract=2319328