FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Algorithms for Automatic Pricing in the Accommodation Services Sector
Student:	Michal Bacigál
Supervisor:	Ing. Mgr. Ladislava Smítková Janků, Ph.D.
Study program:	Informatics
Branch / specialization:	Knowledge Engineering
Department:	Department of Applied Mathematics
Validity:	until the end of summer semester 2021/2022

Instructions

The topic of the work is the design of an algorithm for automatic pricing for accommodation services. Hotel accommodation prices change continuously throughout the year depending on many factors, such as seasons, holidays, sporting and cultural events, hotel occupancy, etc.

1. Carry out a search of the current state of use of mathematical models and AI methods in pricing in the area of accommodation services and outside this area.
2. Design your own algorithm to recommend setting the price for accommodation.
3. Create a SW module for automatic pricing.

In the algorithms, do not consider the influence of measures taken by the government of the Czech Republic, consider the situation before the adoption of measures against COVID-19. The literature will be provided by the supervisor.

Electronically approved by Ing. Karel Klouda, Ph.D. on 30 January 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor Thesis

Algorithms for Automatic Pricing in the Accommodation Services Sector

Michal Bacigál

Department of Applied Mathematics
Supervisor: Ing. Mgr. Ladislava Smítková Janků, Ph.D.

June 27, 2021

Acknowledgements

I would like to express my gratitude to my supervisor, Ing. Mgr. Ladislava Smítková Janků, Ph.D., for her guidance, and to my family and friends who have been a part of my journey and supported me over the past few years.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on June 27, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Bacigál, Michal. *Algorithms for Automatic Pricing in the Accommodation Services Sector*. Bachelor Thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Táto bakalárska práca stavia na súčasných prácach, ktoré sa zaoberajú tvorbou hotelových simulátorov pomocou simulačnej optimalizácie a metód Monte Carlo. Literárna rešerš je zameraná na doterajšie využitie matematických metód a umelej inteligencie na riešenie problémov výnosového manažmentu. Na základe analýzy silných a slabých stránok existujúcich riešení prezentujeme odporúčania, ako súčasné riešenie vylepšiť. V práci prezentujeme nový spôsob modelovania pravdepodobnosti prijatia, na základe ktorého si každý zákazník dynamicky vyberá „cenové bariéry“ (ceny s 0 % a 100 % pravdepodobnosťou prijatia), na rozdiel od súčasného riešenia, v ktorom majú zákazníci fixný pohľad na cenu. Ďalej diskutujeme možnosti širšieho využitia cenových multiplikátorov a ich aplikáciu v úprave akceptačnej krivky. Naše experimenty ukázali, že sa navrhnutý model chová realisticky a je schopný dosiahnutia lepších výsledkov prostredníctvom optimalizácie. V testoch sme namerali zvýšenie výnosov troch simulovaných hotelov s kapacitou 10, 25 a 75 izieb o 3.43, 8.04 a 20.29 %. Toto porozovanie naznačnuje vhodnosť modelu ako alternatívu k existujúcemu riešeniu pre účely ďalšieho výskumu.

Klíčová slova umelá inteligencia, dynamická cenotvorba, evolučné algoritmy, metódy Monte Carlo, výnosový manažment, simulačná optimalizácia

Abstract

This bachelor thesis builds upon contemporary works that deal with creating hotel simulators using simulation-based optimization and Monte Carlo methods. The literature review summarizes the contemporary use of mathematical and artificial intelligence based method in solving revenue management problems. An analysis of strengths and weaknesses of existing hotel simulators is performed, based on which we present suggestions for improvement of the current solution. A novel approach to modelling customer demand is presented, which is based on the concept of each customer choosing a distinct set of 'price walls' (prices with 0 % and 100 % probability of acceptance), rather than having a fixed perception of price, as implemented in previous works. Additionally, we discuss the possibilities of using additional types of price multipliers, as well as their utilization in altering the shape of the demand function. Our experiments have shown that the model behaves realistically and is capable of improving through optimization, achieving an increase of 3.43, 8.04 and 20.29 % in total revenue of three simulated hotels with the capacity of 10, 25 and 75 rooms. This observation suggests the viability of the model as an alternative to the existing solution for further research.

Keywords artificial intelligence, dynamic pricing, evolutionary algorithms, Monte Carlo methods, revenue management, simulation-based optimization

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Motivation

Over the past few decades, tourism has seen a continuous increase in popularity and has since become one of the fastest-growing industries globally, as well as one with a crucial impact on the global economy. It has consistently accounted for approximately one-tenth of the world's gross domestic product [2], with some countries heavily dependent on income from this industry [3]. Additionally, through its combined impact, tourism now generates one in ten jobs, having employed around 330 million people in 2019 [4].

The increasing trend is captured by Figure 1.1. Apart from 2009, when tourism reclined as a result of the Great Recession, there has been a constant growth rate in terms of both international tourist arrivals and tourism exports (the combined cost of international passenger receipts and passenger transport). Just over the last decade, an increase of more than 50 % was registered in both of these metrics.

In 2020, tourism was significantly affected by the global pandemic, resulting in a 74 % decrease in international arrivals [5] that brought tourism levels down to values recorded in 1988. According to UNWTO, it could take as long as four years to return to pre-pandemic levels [6]. Yet, in the midst of an era of accessible transport and technological abundance, tourism has never been in a better position to grow. The digitization of processes in the tourism sector, known as *e-tourism*, paved a way for collection of massive amounts of data generated by the industry, usually in the form of internet searches, accommodation and transport reservations or reviews on social media platforms. As a result of ever-increasing computing power and the accessibility of cloud services, the vast infrastructure required to handle such a large amount of data that was once only financially viable for large businesses has now become an option for everyone, causing a shift in the business sector's dynamics and increasing the competitiveness of smaller entrepreneurs. If properly collected and analyzed, this data can reveal valuable information that provides

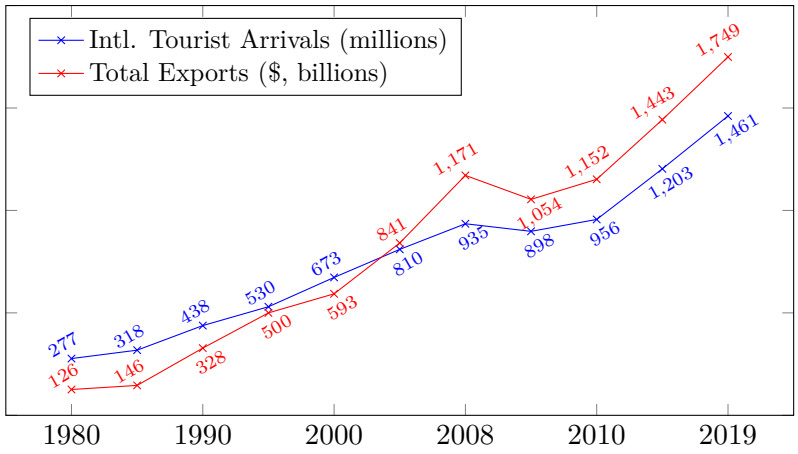

Figure 1.1: International arrivals and tourism exports (1980 – 2019) [1]

businesses with a significant competitive edge. This advantage allows them to optimize resources and offer higher quality services, increasing their chances to attract new customers that can eventually grow into loyal clients.

According to *Bughin et al.* [7], there are four key areas in which big data and artificial intelligence create value for service providers and customers:

- forecasting demand, optimizing R&D and improvement of sourcing,
- increasing production of goods and services of higher quality at a lower price,
- promoting to right customers at the right time and price,
- providing rich and personal user experiences.

Incoincidentally, the abovementioned points are key interests of an area called revenue management. Since its conception in the 1970s, it has spread beyond the airline industry and found its application in numerous business lines, including the hospitality sector [8]. As the popularity of travel grew over time, so did the competition on the market, and with around 700,000 hotels worldwide today [9], the proper use of effective hotel revenue management practices seems more crucial than ever.

Building on the original definitions presented by Kimes [10, p. 15] and Kimes & Wirtz [11, p. 125], Ivanov [12, p. 8] defined hotel revenue management as *"the constellation of tools and actions dedicated toward the achievement of an optimal level of the hotel's net revenues and gross operating profit by offering the right product to the right customers via the right distribution channel at the right time at the right price with the right communication"*.

Much like airline tickets, hotels' products and services have an "expiration date" past which they lose all their value and cannot be consumed on a future

occasion. Each room that the hotel fails to occupy on any date represents a loss of profit that they can only offset in the future, and continuous failure to do so may result in extensive financial problems. This is a significant problem during low demand seasons when revenue managers have to strategize to attract customers in order to cover the fixed costs associated with running the hotel.

Additionally, they have to consider that the amount of products they can sell is limited. On most occasions, especially with rooms, expansion of capacity is not an option, so it is crucial to choose an appropriate pricing strategy to control customer demand. While lowering the price to fill the capacity in advance might seem appealing at first, it results in potential loss of profit that last-minute customers usually bring in. Consequently, the rooms' revenue might not be high enough to turn a desirable profit or even cover the costs of providing the service. On the other side of the spectrum, keeping the price too high may discourage more price-sensitive customers, leaving a high amount of rooms unoccupied, essentially resulting in the same fate.

Out of all revenue management tools, pricing techniques are the only ones that directly affect the hotel revenue [12, p. 12]. Knowing when and how to bend the pricing rules in order to maximize profit is a highly valued skill whose mastering is becoming increasingly more difficult, if not impossible, without the use of intelligent revenue management systems.

The underlying issue is that hotel revenue management relies heavily on data – both internal (generated by the hotel and its customers - reservations, cancellations, purchases) and external (information about demand, competition prices, economic and social factors, weather, ongoing public events, etc.). As the amount of data continues to rise exponentially, it is becoming challenging to make sense of and find the complex connections between different factors without the use of revenue management systems, which is where the power of artificial intelligence methods comes into play. At the same time, the lack of data poses a similar problem. In order to be able to make accurate predictions and models, we need a vast amount of data at our disposal. The assumption of the availability of sufficient data is, however, a bold prediction that may not always hold true. In certain situations, hotel managers are limited to their own experience and observations. In such cases, one must resort to alternative methods capable of simulating hotel processes using a limited amount of input data available at hand.

The original focus of this thesis was to analyze the influences of various internal and external factors on the price of hotels in Prague. However, due to the lack of freely accessible unbiased data, the focus has shifted to exploring the latter situation, where one has to deal with scarcity of historical data. The considerable lack of practical research in this area leaves us room to explore possibilities to improve existing solutions and contribute to the pool of knowledge on this topic.

Bachelor Thesis Structure

This bachelor thesis is structured as follows. *Chapter 2* presents a summary of contemporary knowledge and works in the area of dynamic pricing using artificial intelligence methods and simulation-based optimization, mentions the advantages of utilizing artificial intelligence methods in solving problems in the tourism sector and includes an analysis of components used in works focused on hotel simulation, suggesting several modifications that can be subject for further research. *Chapter 3* lays down the theoretical foundations recommended for proper understanding of methods and practices used throughout this bachelor thesis. *Chapter 4* is dedicated to the analysis of the problem at hand and provides an in-depth overview of our parametric model and its components, identifying the similarities and modifications in relation to previous hotel simulators. *Chapter 5* provides an overview of used technologies and general implementation details of the model and explains how the hotel simulation process is executed. Finally, *Chapter 6* justifies the system's feasibility for further research by demonstrating the functionalities on a series of experiments where we test our model's performance.

Goals

The main objective of this bachelor thesis was to design an algorithm for automatic pricing of rooms in accommodation establishments. This objective included the following subtasks:

- performing research of contemporary application of artificial intelligence methods for solving problems in the tourism sector,
- analyzing the existing implementations of hotel simulators, identifying their strengths and weaknesses and suggesting modifications for their improvement,
- designing an algorithm for real-time pricing of accommodation services and evaluating its feasibility for further research in a series of relevant experiments.

Approach

Due to the lack of freely accessible data and the pandemic-related bias of the few datasets available at hand, we resorted to using synthetically created data using Monte Carlo simulation methods. Using a parametric approach similar to [13], we create an accurate interpretation of a hotel's events based on the experience of hotel managers. First, each reservation is individually priced according to the current state of the hotel and and additional contextual factors

using a series of user-defined multipliers. The multipliers are designed in a way that allows them to take the shape of various functions (linear, n-piecewise linear or constant) and change according to the current situation. Each of the reservations is then passed to the acceptance probability model, which attempts to realistically simulate customers' willingness-to-pay and calculates the probability of acceptance. The adjustable parameters of the multipliers are then optimized using the CMA-ES algorithm, which attempts to find a configuration that maximizes total revenue.

1.1 Chapter Summary

In this chapter, we:

- explained the importance of tourism to the global economy and summarized recent trends in the industry,
- explained the basics of hotel revenue management and its dependence on big data,
- foreshadowed the structure of this bachelor thesis,
- presented the main and partial objectives,
- summarized the chosen approach.

Chapter 2

Literature Review

In this chapter, we summarize contemporary knowledge regarding dynamic pricing, the use of artificial intelligence methods for solving problems in the tourism sector, and utilization of simulation-based optimization in modelling hotel operations. Firstly, we present the definition of dynamic pricing in the context of this bachelor thesis. Secondly, we provide a short overview of history of hotels' view of the importance of designing strategic pricing policies. Thirdly, we summarize the use of mathematical models and artificial intelligence methods in demand forecasting, demand modelling and price optimization, and mention some notable works that contributed to this research field. Finally, we provide an overview of contemporary works focused on using simulation-based optimization models for building accurate hotel simulators, evaluate their strengths and weaknesses and point out suggestions for improvements and future research.

2.1 What is Dynamic Pricing?

Dynamic pricing is a pricing strategy where the cost of a product is adjusted in real-time based on the current state of the market and other selected contextual factors that affect the product's value. In the context of accommodation services, these factors can be split into two main groups and characterized as follows:

- **internal factors**, which are directly connected to or influenceable by the hotel, such as *product quality and differentiation, reputation, marketing strategy* and *fixed and variable running costs*,
- **external factors**, which influence the hotel's operations but cannot be influenced or changed, such as *current demand levels, customers' price elasticity and perception of value, economic and political situation* and *weather conditions.*

Considering the complexity of the search space of possible pricing strategies, it is easy to understand why creating an optimal policy is such a daunting task, and why it has been of such interest to revenue managers and researches since the conception of the idea.

2.2 A Brief History of Approach to Pricing

In the early days of revenue management, it was seldom for hotels to put a lot of effort into developing various pricing strategies, finding optimal pricing policies or trying to increase occupancy during low demand periods. Instead, they resorted to a simple system of opening and closing room rates based on demand levels. According to *Kimes* [14], it was not until the early 1990s that hotels began seeking a more sophisticated approach. The rise of online travel agencies at the dawn of the 2000s brought transparency into the world of hotel rates, which had until then been mostly available on demand only. In the present, the spread of *e-tourism* has allowed travellers to compare rates of various agents almost instantly, which has put hotels under a lot of pressure to stay on par with their competitors. This has led to development of a wide variety of marketing strategies, which have been thoroughly described by Ivanov [12]. In this bachelor thesis, we will focus on developments in the area of dynamic pricing.

2.3 Use of Artificial Intelligence in Tourism

Over the years, artificial intelligence methods have proven to be very effective at solving a wide variety of problems, including ones in tourism and hospitality. Although they are still somewhat novel in this field, at least when compared to their predecessors, they have been a target of extensive research over the past two decades. They have been successfully applied in demand forecasting, where they performed on par or superior to traditional mathematical and econometric models [15, 16]. An interesting approach was chosen by *Chen* [17], who combined traditional mathematical models with back-propagation neural networks and support vector regression. The author concluded that it improved the accuracy of the vanilla models, arguing that the improvements are owed to the ability of AI methods to capture non-linearity in the data.

Another successful application of artificial intelligence methods was their use in demand modelling and price optimization. In a notable work by *Shakya et al.* [18], the authors presented an universal framework for optimal pricing of various products and services consisting of a neural network based module for modelling demand and a pricing optimization engine based on evolutionary algorithms. As an input, the neural network takes in the purchase history and evaluates the effect of various factors on demand, building a so-called demand model. This output is then passed to the the optimization

module, which attempts to find an optimal pricing strategy that maximizes revenue.

An implementation of this framework was later presented in [19]. In the first part, the authors proposed using a set of N back-propagation neural networks, each with:

1 an input layer consisting of one bias node and N input nodes, each representing a price at a given period of time,

2 a hidden layer consisting of one bias node and *(N+1)/2* hidden nodes,

3 a single-node output layer representing the amount of sales at a given period of time,

4 a bias node in bot input and hidden layers.

In the second part, they tested four various evolutionary algorithms to solve the optimization problem. The authors first trained the NN-based model, along with three widely used mathematical demand models: linear, non-linear and multinomial logit models, on a total of 45 datasets. These datasets were generated using the three mathematical models, for each of which three parameter sets were created to simulate different scenarios, and finally, five instances of each were generated. As for the performance of demand models, three key observations were made: (1) the best performance was achieved by the model that fitted the nature of a given dataset, i.e. linear model outperformed other models on the dataset generated by a linear model; (2) the NN-based model consistently scored second for all dataset types; (3) the NN-based model scored the lowest average RMSE error over all datasets. These observations are strong arguments for the use of artificial intelligence methods, as it implies its superiority in real-life scenarios in which one cannot always make assumptions about the linearity of the data.

2.4 Simulation-Based Optimization

While the previous works were based on the premise that sufficient historical data is available, there are situations in which this statement does not hold true. In such cases, the best solution would seem to be to resort to the use of estimations based on personal experience. According to contemporary literature, a seemingly efficient way to approach such problems is through a method called *"simulation-based optimization"*, which combines the advantages of a discrete event simulator with an optimization module that can explore the solution space and find a near-optimal configuration of system parameters [20]. *Mariello et al.* [13] claim that one can build a simulator of hotel processes using Monte Carlo simulation methods, and that such model can reach an approximate maximization of revenue while decreasing the computational complexity

of looking for an exact solution. The approximate layout of such model seems to be shared among several papers focusing on building Monte Carlo-based hotel simulators, and we will discuss approaches to the design of the base components in the following subsections.

2.4.1 Event Simulation

Arguably the most essential part of building an accurate (hotel) simulation is the generation of events that make up the hotel sales process. In the simplest case, we consider there to be two opposing key events: reservations and cancellations. As the amount and distribution of these events throughout a calendar year is highly specific to each establishment, it is crucial to adjust the values accordingly to keep the simulation realistic. If historical data is available, it can be used to either estimate the values directly or forecast future arrival rates through time-series analysis or utilization of artificial intelligence methods. In the opposite case, a customary approach is to opt for Monte Carlo simulation methods, which work by performing repeated random sampling using estimations provided by hotel managers. By the law of big numbers, the average values of samples provided by Monte Carlo methods should converge to the average values supplied as input. One of the first works to use these methods was [21], which focused on forecasting arrivals in future periods based on estimations derived from historical data. The authors chose to model events as a series of Bernoulli trials with probabilities of event occurence estimated from available past observations. They disregarded the use of the Poisson process, arguing that the nature of collected data may lead to rounding inaccuracies, as opposed to *Mariello et al.* [13], who justified their use of a non-homogenous Poisson process to model events with the claim that binomial distribution converges to a Poisson one with an indefinitely increasing amount of trials, as per the Poisson limit theorem. The essential thought behind the generation process, however, remains the same. In both cases, events were generated using an estimated amount of event occurrences on a given date and a distribution function that specified the distribution of the amount over a specific range preceding the date. A somewhat simpler approach was chosen by *Brunato and Battiti* [22], who chose to draw the reservation creation date from an exponential distribution. Apart from the reservation creation and arrival dates, it is customary to generate two additional parameters specifying the length and size of a reservation. Both the amount of overnight stays and booked rooms usually follow an exponential distribution, as seen when estimated from data [21], and have been modelled either as such [22], or using the Beta distribution [13].

2.4.2 Pricing Strategies

The most common approach to pricing is through the use of a multiplier-based technique proposed in [23]. The technique is based on multiplying a (seasonal)

reference price with a collection of multipliers with an average value of 1. This is constraint ensures that the average daily rate does not deviate significantly from the price set by the hotel, but still allows it to change its value so that it has considerable effect on customer demand. The value of multipliers is set dynamically in response to specific hotel and reservation parameters, namely the current capacity, time-to-arrival, length-of-stay and number of rooms booked within a reservation. An advantage of this approach is that the number of parameters to be optimized is limited to six (in the case of [23]), with the other being calculated mathematically so as to keep the average around 1, effectively reducing the optimization complexity. To ensure that the price stays within reasonable bounds, [13] modified the pricing function to keep the price within 60 % from the specified reference price. The superiority of multiplier-based pricing was demonstrated in [22], where multiple exact and heuristic policies were evaluated on a set of differently sized simulated hotels. The authors have concluded that if a proper optimization technique is used, factored pricing can provide a considerable increase in earned revenue. What we consider as the main strengths of this method is that it is both easy to comprehend and implement. It simulates the way prices of products are adjusted in real situations and can be easily expanded to include additional factors the hotel wants to apply a premium or discount to. In this bachelor thesis, we will try to explore the possibility of adding additional multipliers in conjunction with slight modifications to the demand model. In particular, we want to explore the possibility of applying premium on weekends, holidays and days when special events take place. To achieve this, we implemented a functionality that allows the user to easily add customized multipliers that take the shape of linear, n-piecewise linear or constant functions.

2.4.3 Demand Modelling

One of the biggest challenges involved with building a hotel simulator is modelling demand. When historical data is not available, it is very difficult to estimate the price elasticity of customers. Furthermore, customers tend to have different budgets and perceptions of product value of. Therefore, a common approach is to use a model where the probability of acceptance is calculated based on the price difference between the actual price of a reservation and a pre-defined "median price" with a 50 % acceptance probability. In [22], the probability is calculated by applying the sigmoid function on the price difference divided by a slope parameter that simulates different values of price elasticity. The model proposed in [13] uses a similar approach, replacing the sigmoid with a cumulative distribution function of the standard normal distribution. The value of the slope is selected so that the probability of acceptance reaches zero when there is a 50 % price increase, and vice versa. Our model presents a slightly modified approach to modelling demand by considering the following ideas, which will be thoroughly explained later in the work.

2.4.3.1 Simulating various budgetary conditions

One of the flaws in contemporary models is that they make a bold assumption that every customer perceives prices in the same manner. The way the price acceptance models are designed implies that potential customers consider the *reference price* P_R as given and ignore possible budgetary conditions a customer may have. In our model, we attempt to simulate this feature by having each customer select their *proposed price* P_P from a range of values around P_R (e.g. 0.85 to 1.15 times P_R). This value indicates the price they are willing to pay for the product (or the value they assign to it), and one at which they will accept an offer with 100 % probability. Additionally, each customer selects their *threshold price* P_T as a certain multiple of their P_P, which defines the price with 0 % probability of acceptance. As the *actual price* P_A exceeds P_P and approaches P_T, the acceptance probability will decrease according to Equation 2.1.

$$pbt_{accept} = 1 - (\Delta_P)^{n^k} \tag{2.1}$$

where

- $\Delta_P = \frac{P_{actual} - P_{target}}{P_{threshold} - P_{target}} \in [0, 1]$
- n is the parameter that controls the slope of the function
- k is the slope magnifier, which increases the effect of changes to n on the slope of the function

A more detailed discussion of this implementation will be presented in Chapter 4.5.

2.4.3.2 Simulating changes in price elasticity over time

Another oversimplifying assumption is that price elasticity is constant and does not change over time. While this may be true with certain customers, it is customary for one to adjust their expectations of price as the time-to-arrival changes. As an example, a customer may be more price-sensitive when booking several weeks in advance, but will slowly lower their expectations as the arrival window closes in. If a booking is made very close to the check-in date, there is a higher probability of a customer's price elasticity to be low, as it is common practice for hotels to increase their price as time-to-arrival decreases.

2.4.3.3 Simulating expectancy of price discounts

In a world saturated with deals and discounts, it is understandable that a customer may have certain expectations of discounts in specific situations. The terms 'first-minute' and 'last-minute' have reached the core of travellers'

dictionary and are often more of a rule than exception. Customers may also expect group discounts when travelling with friends or family or lower prices when booking a longer amount of nights.

2.4.3.4 Simulating tolerance of price premiums

On the other side of the spectrum, it is common practice for hotels to increase prices during holidays and special events so as to maximize revenue. In some cases, prices can increase several times over. A knowledgeable customer understands that prices may rise during these events and will adjust his acceptance accordingly.

2.4.4 Revenue Optimization

Optimization is a crucial part of the model, as it is used to find optimal values of parameters used in the price multipliers. While the selection variety of evolutionary algorithm is wide, most works have chosen to use CMA-ES [24] for optimization purposes. A notable advantage of this method is, as pointed out in [23], that the only input parameters required are the boundary values of parameters. While *Brunato and Battiti* [22] argued that this technique showed underwhelming performance, mostly performing worse than all methods except for fixed pricing and therefore cannot be used in practical situations, *Mariello et al.* [13] and *Bayoumi et al.* [23] managed to reach positive results, increasing revenue by at least 12.8 % and 13.39 %, respectively. As far as other evolutionary algorithms are concerned, [22] compared the performance of CMA-ES with affine and inertial shakers, reactive optimizers presented in [25] and [26], respectively. However, research of other algorithms and their performance in hotel simulators is yet to be explored.

2.5 Chapter Summary

In this chapter, we:

- presented the definition of dynamic pricing in the context of accommodation service providers,
- provided a brief historical overview of hotels' perception of importance of creating strategic pricing policies,
- summarized the use of mathematical and artificial intelligence methods in demand forecasting, demand modelling and price optimization,
- thoroughly summarized the application of simulation-based optimization for modelling hotel processes, discussed the strengths and weaknesses of existing solutions and contributed own suggestions for further research and implementation.

Chapter 3

Theoretical Foundations

In the following chapter, we lay down the minimal theoretical foundations required for proper understanding of a selection of essential methods and concepts that this thesis builds upon. More specifically, we discuss the following: using Monte Carlo methods for simulation and general summary of evolutionary algorithms.

3.1 Monte Carlo Simulation

When working with mathematical models, we often want to find an exact analytical solution to the problem at hand. However, finding such a solution might not always be the best approach, such as when

- the complexity of the problem makes it computationally too difficult or impossible to solve in a reasonable amount of time,
- the input variables show signs of randomness and uncertainty,
- the requirements are satisfied by an approximate solution.

In such cases, it is viable to choose an approach that yields numerical solutions. Although they are not as accurate as their analytical counterparts, this disadvantage is offset by the reduction in time, cost and computing power required to find one.

Although there is no consensual definition of **Monte Carlo methods** [27], the term usually refers a collection of methods used for obtaining numerical solutions by the means of repeated random sampling. Its name was inspired by the eponymous Monte Carlo, an administrative area in Monaco, which is known for its gambling establishments.

Let us demonstrate this on a simple case of estimating the value of π using a technique illustrated in Figure 3.1.

Consider a machine that draws random points on a piece of paper. The paper contains a drawing of a circle (with a radius of $r = 1\text{cm}$) and a square (with a side of $a = 1\text{cm}$).

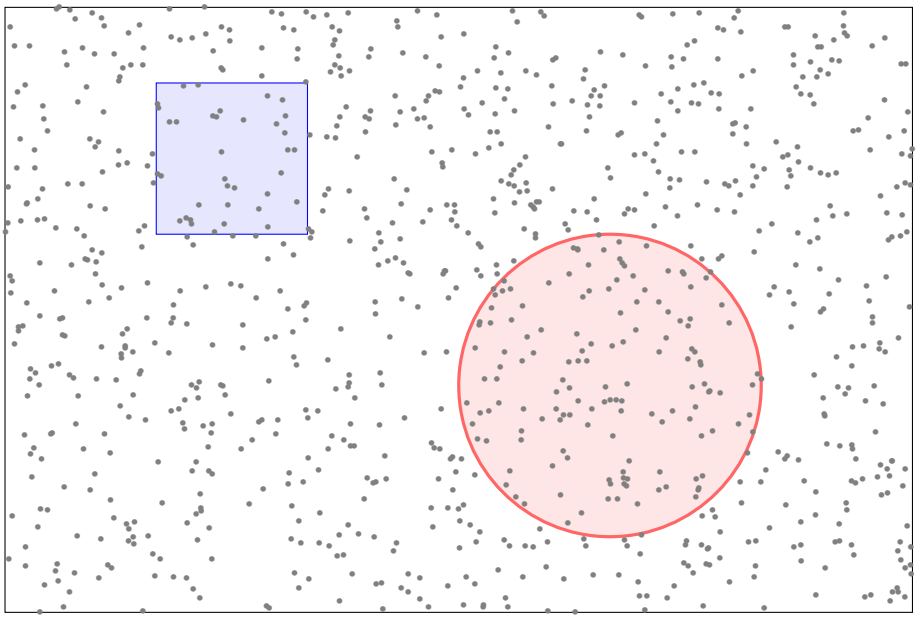

Figure 3.1: Experiment layout for estimation of π via Monte Carlo methods

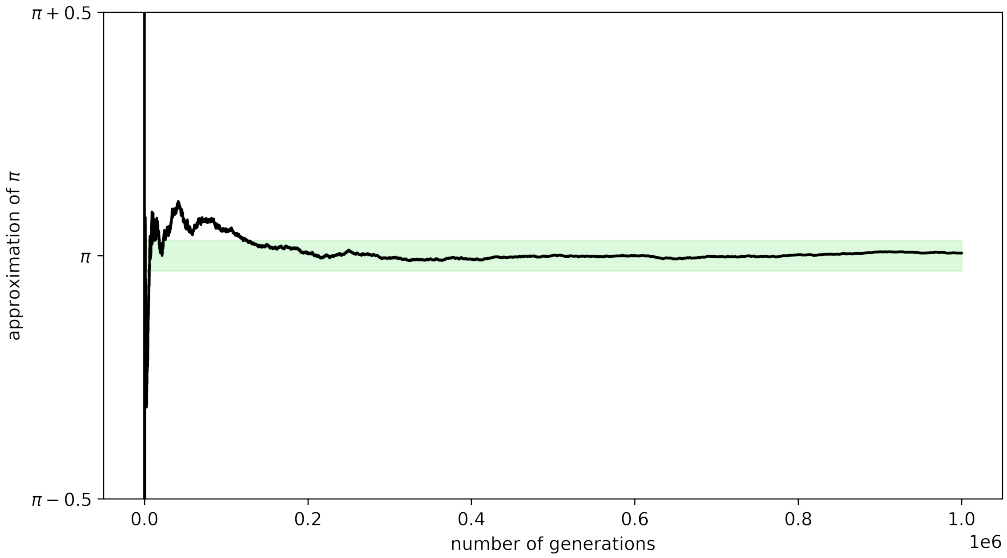

Figure 3.2: Estimation progress of π using the method from Figure 3.1

As the amount of dots on the paper increases, so does the amount that

falls into either of the shapes. At some point, there will no more space on the paper left (we will have approached positive infinity), and the ratio of dots that "fell" into the shapes should be equal to the ratio of their area. Figure 3.2 shows the estimated value by generation (namely the ratio of circle area and square area). The yellow section shows the values that fall into 1% from the actual value of π.

Although this is one of the simplest ideas to explain the principles of random sampling on, it present a rough picture of how Monte Carlo methods. Instead of working specific inputs, they draw input values from a probability distribution .For each set of input values, we receive a corresponding set of output values. By collecting and averaging a sufficient amount of samples, we should receive a considerably approximate estimation of the observed variable [28].

3.2 Evolutionary Algorithms

It is not uncommon for computer algorithms to take inspiration from the way nature works, with evolutionary algorithms being no exception [29]. As their name suggests, their modus operandi mimics the key principles of the theory of evolution by Charles Darwin.

Instead of working with a single solution at a time, as is the case of artificial neural networks, evolutionary algorithms work with a collection of solutions known as **a population**. A single solution within a population is called **an individual**. During each iteration, known as **a generation**, a set of **genetic operators** is applied on the population and its individuals to ensure advancement and add variation to the existing solutions. The quality of an individual is evaluated using **a fitness function**. This process repeat until a certain number of generations has passed or **a termination condition** is met.

3.2.1 Genetic Operators

Throughout the evolution process, various functions or modifiers called **genetic operators** are applied to the population and its individuals in order to alter their properties and prevent premature convergence to a local minimum or a suboptimal solution. The most common operators used in evolutionary algorithms include **initialization**, **selection**, **crossover** and **mutation**.

3.2.1.1 Initialization

The process of initialization in the context of evolutionary algorithms includes the creation of **an initial population** of a predefined size. The simplest, yet the most common approach is to generate a random set of individual solutions from the entire solution space. Depending on the nature of the problem, a

heuristic approach can be used in order to generate the initial population in a subset of the solution space where optimal solutions are more likely to be found

3.2.1.2 Selection

The function of the selection operator is to evaluate which individuals from a population are to be chosen for later breeding. Some common approaches include **roulette-wheel selection**, in which each individual has a selection change proportional to its normalized fitness value; **tournament selection**, in which random sets of individuals are selected to compete against each other and **elitism selection**, in which a small portion of best individuals are selected for further reproduction.

3.2.1.3 Crossover

The crossover operator is used for creating new individuals by combining the genetic payload of two parents. Depending on the settings, one or more offspring can be generated. In case of a binary representation the genotype, the crossover can be performed using ***k*-point crossover**, in which a chromosome is split into $(k+1)$ subsections which are then exchanged between parents in an alternating fashion; or **uniform crossover**, in which each bit is selected randomly from either of the parents.

3.2.1.4 Mutation

Mutation is used for making minor changes to an existing solution in order to maintain diversity between generations. Assuming a binary representation, the simplest way to perform mutation is by using **bit-flip mutation**, which takes n random bits and inverts them.

3.3 Chapter Summary

In this section, we:

- explained the basic principle of Monte Carlo methods,
- explained the characteristics of evolutionary algorithms.

Chapter 4

Analysis and Design

The following chapter will be dedicated to the the analysis of the problem of hotel simulation and price optimization, as well as the design of the proposed model. Firstly, a short, general description of core components of the model is provided. Secondly, we will present the terminology used in the model. Thirdly, we will provide a thorough description of the event generation process. Fourthly, we will explain the principles of multiplier-based pricing system. Penultimately, we will discuss and present the modified acceptance probability model proposed in this bachelor thesis. The chapter will be closed with a comparison of our model with the general trend in reference works.

4.1 Core Components

The backbone of the model proposed in this bachelor thesis has been mostly adapted from related contemporary works [13, 22, 23], with specific modifications to the pricing and acceptance modules, which we will discuss later in this chapter. It consists of the following main components:

- **an event generation module** capable of providing realistic simulations of incoming reservations and cancellations using only a minimal amount of input data,
- **a pricing module** for creating tailored dynamic pricing policies using multiplier functions that change their value according to user-defined price determinants,
- **an acceptance probability module** used for modelling customers' response to proposed price offers,
- **an optimization module** that optimizes parameters of the pricing module through evolutionary computation so as to maximize revenue.

4.2 Terminology

Prior to a deeper discussion of the model and its principles, we will dedicate the following section to an introduction to basic terminology used in this chapter. This is mostly to clarify the meaning of abbreviations and eliminate any possible misconceptions about terms.

4.2.1 General Terms

The following definitions are general terms that are broadly used in this thesis, which we consider to be ambiguous or confusing, and therefore define it to clarify their meaning.

Definition 4.2.1. A *hotel* is an accommodation establishment that is subject to the simulation and optimization process. The type of the establishment (hotel, hostel, B&B, etc.) is considered irrelevant for the purposes of this thesis and will therefore be referred to as hotel regardless of its actual type.

Definition 4.2.2. A *hotel room*, or simply *room*, is a single unit of accommodation provided by a hotel. It is considered a resource of limited quantity and the main product of the hotel. For simplification purposes, we assume all rooms to be of the same, indistinguishable type, although it is often different in practice.

Definition 4.2.3. A *hotel manager* is a person in a managerial position that is knowledgeable in the way the hotel operates and can provide information required for a proper setup of the model, such as expected amount of reservations and cancellations throughout a typical year.

Definition 4.2.4. A *customer* is a person that expresses interest in purchasing the hotel's products by placing a reservation request, regardless of whether the request is finalized or not.

Definition 4.2.5. The *system* or *model* are terms that refer to the simulation-based optimization model implemented in this thesis.

4.2.2 Timeframes

Throughout this thesis, we will often refer to certain timeframes called *windows*. In this subsection, we explain their specific meanings and clarify the constraints that apply among them.

Definition 4.2.6. The *reservation window RW* is a range of dates on which a reservation can be created, effectively forming the domain of possible values of R_{cdate} (see Definition 4.2.12).

Definition 4.2.7. The *arrival window AW* is a range of possible arrival dates of reservations, effectively forming the domain of possible values of $R_{arrival}$ (see Definition 4.2.12).

Definition 4.2.8. The *booking window BW* is the maximum number of days prior to any check-in date that a reservation can be created on.

There are several constrains that apply between the timeframes that must be met so that the system functions properly.

Definition 4.2.9. Let $X \in \{RW, AW, OW\}$. Then X_{start} and X_{end} are the first and last dates in X, respectively.

- (C0) $X_{start} \leq X_{end}$

The constraint (C0) is trivial and can be deduced from the definition of a date range.

- (C1) $RW_{start} \leq AW_{start}$

According to (C1), the arrival window must not start prior to the reservation window. This is due to the fact that reservations cannot be generated for a date in the past. If (C1) is violated, it will not be possible to generate reservations with arrival dates prior to RW_{start}, resulting in possibly unwanted behaviour. Note that there are no constraints for the relationship between RW_{end} and AW_{end}, meaning that AW may or may not be a subset of RW, as long as the abovementioned constraint is satified.

- (C2) $AW_{start} - BH \leq RW_{end}$

The constraint (C2) serves to ensure non-emptiness of the set of generated events. Since $AW_{start} - BH$ represents the first day on which an event can be generated, setting it after the last day of the reservation window would mean that events cannot be generated on any day of RW, as there would be a gap between the last day of RH and the first day a reservation could be made for the first day of AH.

4.2.3 Hotel Parameters

Definition 4.2.10. Each hotel has a limited amount of rooms that can be sold a single night, defined by the parameter H_{max_rooms}. The amount of currently free rooms, should some of them be occupied, is defined by H_{free_rooms}.

Definition 4.2.11. The maximum length and group size of a reservation is defined by H_{max_length} and H_{max_size}, respectively.

4.2.4 Reservation Life Cycle

Throught its lifetime, a reservation can take on several states. To clarify the meaning of each one of them, we present the following diagram:

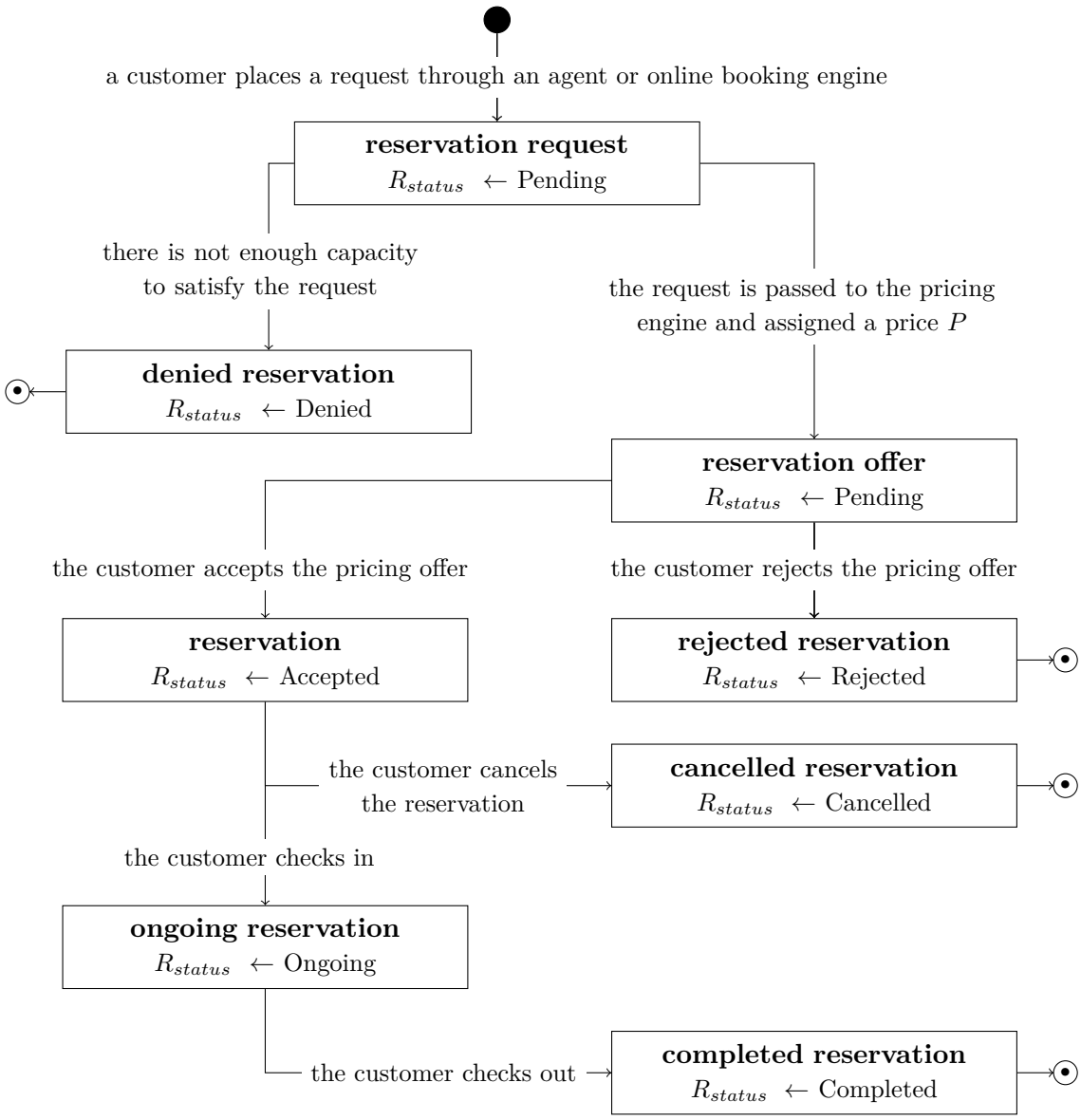

Figure 4.1: Schema of the reservation life cycle

4.2.5 Events

Definition 4.2.12. A *reservation* R is an event defined as follows:

$$R = (R_{status}, R_{cdate}, R_{arrival}, R_{length}, R_{size}, R_{price})$$

where

- the *reservation status*, R_{status}, specifies the stage of the reservation process that R is currently in,
- the *reservation creation date*, $R_{cdate} \in RW$, specifies the date when the reservation request was placed,
- the *reservation arrival date*, $R_{arrival} \in AW$, specifies the date when the customer is scheduled to arrive,
- the *reservation length-of-stay*, $R_{length} \in [1, H_{max\_length}]$, specifies the amount of overnight stays,
- the *reservation group size*, $R_{size} \in [1, H_{max\_size}]$, specifies the amount of rooms to be booked within this reservation,
- the *reservation price*, $R_{price} \in \mathbb{R}^+$, specifies the end price for the customer.

Definition 4.2.13. A *cancellation* C is an event defined as follows:

$$C = (C_{date}, R)$$

where

- C_{date} is the date when the cancellation occured,
- R is the reservation that was cancelled.

4.3 Event Generation

There are two key types of events involved in the hotel simulation process: **reservations** and **cancellations**. The frequency and distribution of these events are unique for each hotel and may change over time. Proper knowledge of these patterns is a crucial prerequisite for successful revenue management [12]. In most cases, a hotel can use its historical data to analyze patterns in the past and predict future trends in arrivals. However, availability of a sufficient amount of data can not be always achieved, such as in the early stages of a hotel's existence or when the required infrastructure is too costly. In such cases, the best approach would be to rely on the experience of the

hotel manager, who should be able to provide sufficient estimations of arrivals with enough granularity to build a fairly accurate model.

The process of event generation used in this bachelor thesis has been adapted from [13] and [22]. Reservation requests are generated using Monte Carlo simulation methods. Considering the lack of arrival data, we need to provide the model with the estimated amount of requests generated on each day of the reservation window. If historical data is available at hand, calculation of $R_{:\to a}$ is a trivial task. In our case, this value needs to be estimated in the following way:

$$\mathbb{E}[R_{:\to a}] = \frac{\mathbb{E}[R_{:\to a}^{accept}] + \mathbb{E}[R_{:\to a}^{cancel}]}{\mathbb{E}[pbt_{accept}(R)]} \tag{4.1}$$

where

- $\mathbb{E}[R_{:\to a}]$ is the expected amount of reservations with an arrival date a,
- $\mathbb{E}[R_{:\to a}^{accept}]$ / $\mathbb{E}[R_{:\to a}^{cancel}]$ are the expected amounts of accepted / cancelled reservations with an arrival date a,
- $\mathbb{E}[pbt_{accept}(R)]$ is the average expected probability that a reservation will be accepted (or the ratio of accepted reservations).

Using the values above, we can proceed to actual event generation. For each day $d \in RW$, we need to evaluate $R_{r\to(r+i)},\ i \in \{0, 1, ..., BW\}$ using the following formula:

$$R_{r\to(r+i)} = Poisson\left(\mathbb{E}[R_{r\to(r+i)}] * \mathbb{Q}_{\alpha_{accept}}(i, BW)\right) \tag{4.2}$$

where $\mathbb{Q}_{\alpha_{accept}}(i, BW) = \left(\left(\frac{BW+1-i}{BW+1}\right)^{\alpha_{accept}} - \left(\frac{BW-i}{BW+1}\right)^{\alpha_{accept}}\right)$ is a quantifier function used to control the distribution of events over the booking window, as per [13, 30]. For better understanding of its functionality, we provide its visualization in Figure 4.2.

The function in Equation 4.2 yields a number n which specifies the amount of reservations generated on day d with an associated arrival day $d + i$. We generate these n reservations using the following algorithm:

1 The reservation creation date, R_{cdate}, is set to d.

2 The reservation arrival (check-in) date, $R_{arrival}$, is set to $d + i$.

3 The reservation length, R_{length}, is drawn from the exponential distribution with the rate parameter λ set to the average amount of nights per reservation.

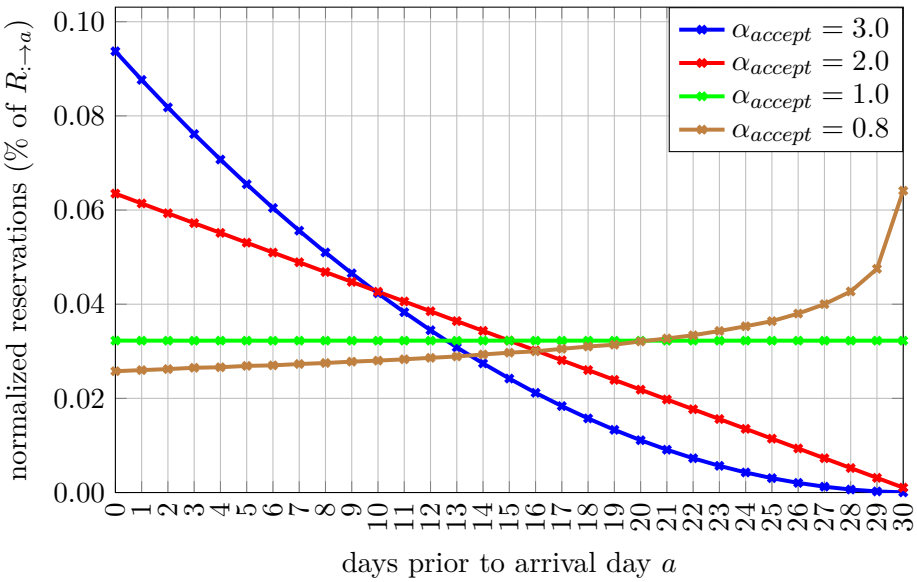

Figure 4.2: Values of $\mathbb{Q}_\alpha$ for $BW = 30$ and various values of α_{accept}

4 The number of rooms, R_{rooms}, is calculated using Equation 4.3.

$$R_{rooms} = \begin{cases} 1 & r < \gamma \\ c & otherwise \end{cases} \tag{4.3}$$

where

- r is a random number, $r \in [0, 1]$,
- γ is a threshold parameter, $\gamma \in [0, 1]$,
- c is drawn from the exponential distribution with the rate parameter λ set to the average amount of booked rooms per reservation.

5 The reservation status, R_{status}, is set to *Pending*.

Upon iterating over the entire reservation window, we will have obtained a full collection of reservation requests that will be passed to the pricing module. As far as cancellations are concerned, they are generated on every day of the reservation process in a way analogical to reservation requests. In order to calculate the amount of cancellations to generate, we must first estimate the ratio of cancelled reservations to all reservations in the following way:

$$\Omega_a = \frac{\mathbb{E}[R^{cancel}_{:\to a}]}{\mathbb{E}[R^{accept}_{:\to a}] + \mathbb{E}[R^{cancel}_{:\to a}]} \tag{4.4}$$

In analogy to Equation 4.2, we can calculate the amount of cancellation generated on each day $d \in RW$ associated with each of the possible arrivals days $d+i, i \in [0, R_{TTA}]$ as follows:

$$C_{r\to(r+i)} = \Omega_{r+i} * \mathbb{Q}_{\alpha_{cancel}}(i, R_{TTA}) \tag{4.5}$$

where

- Ω_{r+i} is the value calculated in Equation 4.4,
- R_{TTA} is the reservation time-to-arrival, calculated as:

$$R_{TTA} = R_{arrival} - R_{cdate},$$

- $\mathbb{Q}_{\alpha_{cancel}}(i, R_{TTA}) = \left(\left(\frac{R_{TTA}+1-i}{R_{TTA}+1}\right)^{\alpha_{cancel}} - \left(\frac{R_{TTA}-i}{R_{TTA}+1}\right)^{\alpha_{cancel}}\right)$.

Equation 4.5 yields a value from the interval $[0, 1]$, which defines the probability that a reservation is cancelled on a given day.

4.4 Dynamic Pricing

When a user submits a reservation request, it is sent to the pricing module, which looks at the current state of the hotel and factors that affect the price and makes one of the following decisions:

- if for any of the relevant nights, H_{free_rooms} happens to be smaller than R_{size}, the hotel rejects the request,
- if there is sufficient capacity to accommodate the request, the price is calculated according to Equation 4.6 and passed to the acceptance probability model

$$P_A = \left(\sum_{i=1}^{l}(\bar{P}_R^i \times \prod_{j=1}^{m} M_j^N)\right) \times \prod_{k=1}^{n}(M_k^T) \tag{4.6}$$

where

- P_A is the **actual price** of a reservation that will be proposed to the customer,
- l is the **number of nights** ($= R_{length}$),
- $\bar{P}_R^i$ is the **reference unit price** for night i out of l,
- m is the **amount of nightly multipliers**,
- M_N is a **nighly multiplier** (see Definition 4.4.2),

- n is the **amount of total multipliers**,
- M_T is a **total multiplier** (see Definition 4.4.3).

The pricing system used in this bachelor thesis utilizes the concept of multipliers presented in [23].

Definition 4.4.1. A *multiplier* M is a function used to apply slight alterations (≈ 1) to the reference unit price P'_R depending on the current state of its single dependent variable x_M, which is usually a variable taken from the hotel registry, the reservation R itself, or the value of a relevant factor affecting the price.

Definition 4.4.2. A *nightly multiplier* M_N is a multiplier that is applied to each night independently. Nightly multipliers are evaluated in the inner cycle of the price evaluation process (see Equation 4.6).

Definition 4.4.3. A *total multiplier* M_T is a multiplier that is applied to the reservation as a whole. Total multipliers are evaluated in the outer cycle of the price evaluation process (see Equation 4.6).

Note 4.4.1. The reasoning behind the split of multipliers into two categories is that some multipliers need to be evaluated on a nightly basis. As an example, if we supply the pricing module with a multiplier that applies a premium on overnight stays on Fridays and Saturdays, we cannot apply the multiplier on the reservation as a whole, as it would increase the price of nights that the premium does not apply to.

Definition 4.4.4. A linear multiplier M_L is a multiplier that takes on the shape of a linear function with the following signature:

$$M_L = (x_{M_L}, [X_1, X_2], [P_1, P_2]) \tag{4.7}$$

where

- x_{M_L} is the **dependent variable**, e.g. R_{TTA}, H_{free_rooms},
- X_1, X_2 are **boundary values of** x_{M_L}, usually set to 0 and $max(x_{M_L})$,
- P_1, P_2 are **boundaries of the multiplier value**, usually set so that their average is ≈ 1.

The evaluation of a linear multiplier function is done by performing a least squares polynomial regression using two provided data points. When the multiplier is instantiated, the input values will be converted to a well-known signature of a linear function:

$$M_L(x) = ax + b \tag{4.8}$$

An example of a linear multiplier that provides up to a 30 % discount when there is a scarcity of rooms (Equation 4.9) can be seen in Figure 4.3.

$$M_1 = (H_{free\_rooms}, [0, 100], [1.3, 0.7]) \tag{4.9}$$

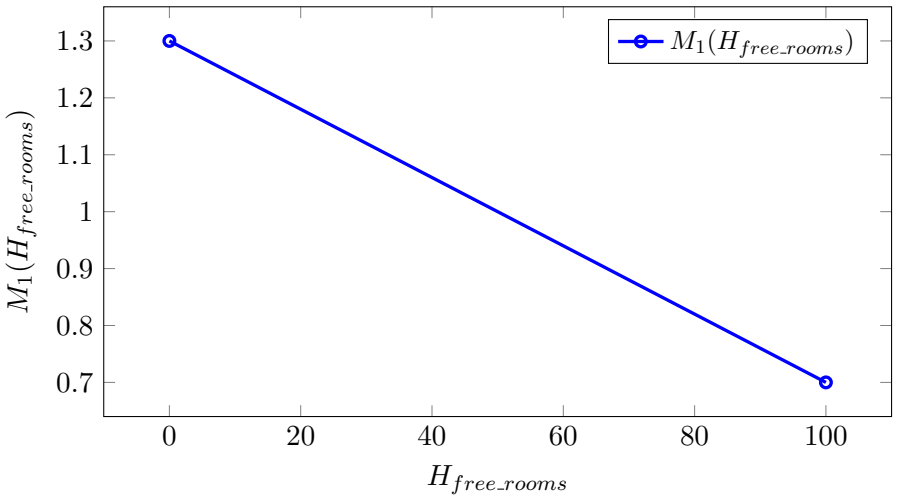

Figure 4.3: Example of a linear multiplier based on the value of H_{free_rooms}

Definition 4.4.5. An n-piecewise linear multiplier $M_{PL(n)}$ is an extension of the linear multiplier that is composed of several segments, with each being defined as a standalone linear function. The signature of the multiplier is as follows:

$$M_{PL(n)} = (x_{M_{PL(n)}}, X^{n+1}, P^{n+1}) \tag{4.10}$$

where

- $x_{M_{PL(n)}}$ is the **dependent variable**, e.g. R_{TTA}, H_{free_rooms},
- X^{n+1} is a (n+1)-element **container of boundary values of** $x_{M_{PL(n)}}$,
- P^{n+1} is a (n+1)-element **container of boundary values of the multiplier**.

Note 4.4.2. Trivially, $M_{PL(1)}$ is equal to M_L.

The evaluation of the n-piecewise linear multiplier function is analogical to regular linear multipliers. Each segment is extracted and evaluated separately. The signature of the resulting n-piecewise linear function goes as follows:

$$M_{PL} = \begin{cases} a_0 x + b_0 & \text{if } X_0 \leq x \leq X_1 \\ \dots & \\ a_{n-1} x + b_{n-1}, & \text{if } X_{n-2} < x \leq X_{n-1} \end{cases} \tag{4.11}$$

An example of a 3-piecewise linear multiplier that provides customers who book early with up to a 25 % discount that gradually decreases up to 3 days prior to arrival, then drops an additional 10 % to attract last-minute customers (Equation 4.12) can be seen in Figure 4.4.

$$M_2 = (R_{TTA}, [0, 3, 30], [0.9, 1, 0.75]) \tag{4.12}$$

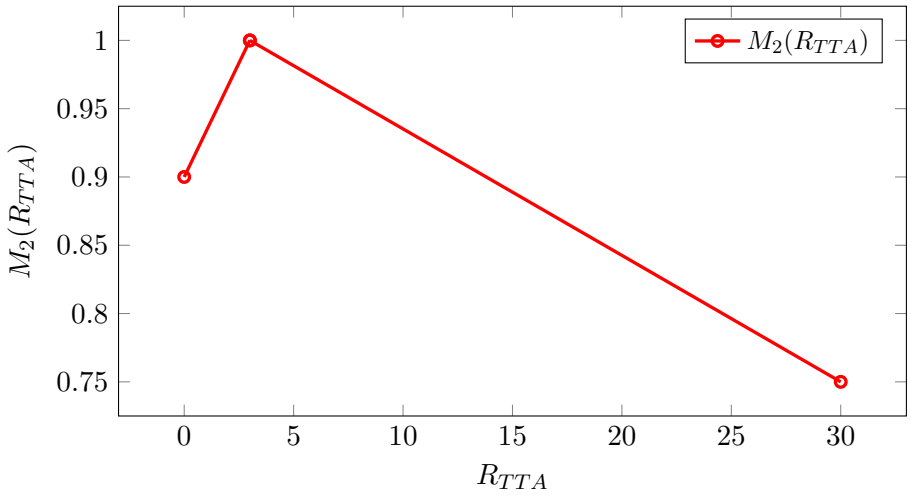

Figure 4.4: Example of a 3-piecewise linear multiplier based on R_{TTA}

Definition 4.4.6. A constant multiplier M_C is a multiplier used for specifying fixed premiums or discounts based on evaluation of a certain condition.

$$M_C = (f_{M_C}, C) \tag{4.13}$$

where

- f_{M_C} is a function that evaluatues a certain condition, e.g. a certain subset of weekdays or dates in a calendar year,
- C is the constant value of the multiplier.

4.5 Acceptance Probability Modelling

Once a reservation request has been assigned a price by the pricing module, it is passed to the acceptance probability module, which calculates the customer's willingness to accept a given offer at a proposed price. The probability calculation process is performed using the following steps:

1 For each customer, a *preferred price* P_P is calculated as follows:

$$P_P \sim \mathcal{N}(\frac{(\alpha+\beta)\times P_R}{2}, \frac{(\beta-\alpha)\times P_R}{3}) \tag{4.14}$$

where

- $\frac{(\alpha+\beta)\times P_R}{2}$ is the **mean** μ of $\mathcal{N}$,
- $\frac{(\beta-\alpha)\times P_R}{3}$ is the **standard deviation** σ of $\mathcal{N}$,
- P_R is the **reference price** for the specific reservation,
- α, β are **multipliers** of P_R that define the possible domain of P_P,

with the following additional constraints

$$0 \leq \alpha \leq \beta \ \wedge \ min(P_P) = \alpha \times P_R \ \wedge \ max(P_P) = \beta \times P_R.$$

P_P can be interpreted as either the customer's perceived value of the offered product, or their budget for accommodation spendings. Effectively, it specifies the price at and under which the probability of acceptance remains fixed at 100 %. Let us define this fact for further reference:

$$pbt_{accept}(R, P_A) = 1.0, \quad \text{if } P_A \leq P_P \tag{4.15}$$

2 Analogically to Step 1, each customer is assigned a *threshold price* P_T in the following way:

$$P_T = \begin{cases} P_P \times (1 + [(\gamma_{max} - 1) \times f_S(X, \delta)]), & \text{if } \gamma_{max} > 1 \\ P_P + \epsilon, & \text{if } \gamma_{max} = 1 \end{cases} \tag{4.16}$$

where

- P_P is the **preferred price** for the reservation (see Step 1),
- γ_{max} is the upper bound of the **multiplier** specifying the lowest multiple of P_P that the customer is no longer willing to tolerate,
- $f_S(X, \delta) = (1 - (-x + 1)^\delta)$ is the **scaling function**,
- $X \sim \mathcal{U}(0, 1)$ is a **random variable** with an uniform distribution,
- δ is the **slope** of the **scaling function** f_S,
- ϵ is the **fractional monetary unit** of the used currency.

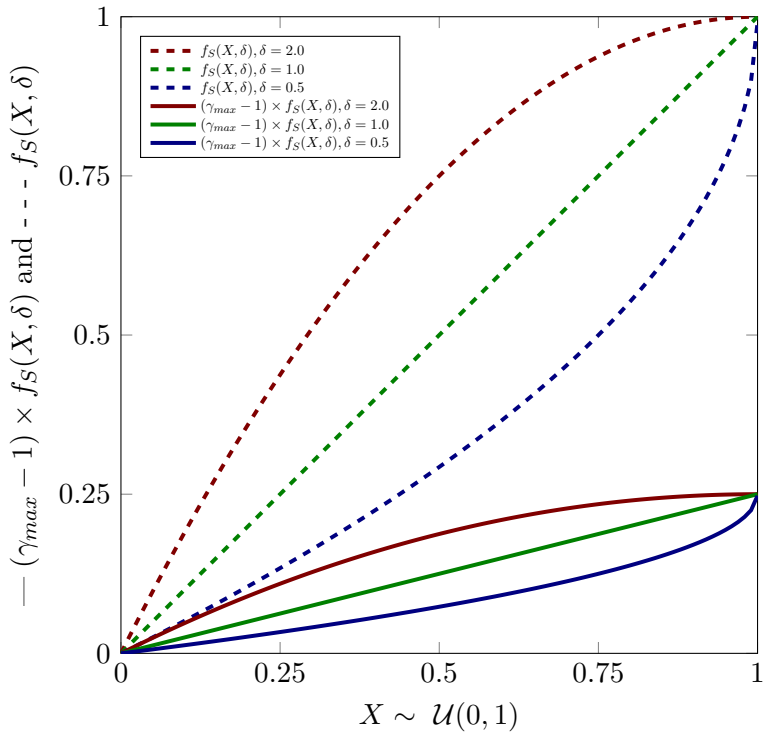

Figure 4.5: Values of $f_S(X,\delta)$, $f_S(X,\delta) \times (\gamma_{max}-1)$ based on γ_{max}, δ

P_T can be interpreted as the exclusive outer bound of perceived value. In other words, it represents the lowest price the customer is no longer willing to pay. Any reservation with this price or higher will be automatically rejected, as its probability value is equal to 0 %.

To further clarify the reasoning for choosing the approach described by Equation 4.16, we will first look at the second case, in which $\gamma_{max} = 1$. Had this exception not been handled, setting γ_{max} to 1 would result in P_T being equal to P_P, which would mean that the customer is not willing to pay more than their preferred price ($p = P_P = P_T$). This would, however, cause the price p to have both 0 and 100 % chance of being accepted. For this reason, we chose to set P_T to be one fractional monetary unit ϵ (e.g. 0,01€ for Euros) higher than P_P. As a practical example, setting P_P to 50.00€ and γ_{max} to 1 would result in P_T being equal to 50.01€. Although this is not very common in practice, it ensures that the model functions properly should this edge case happen. If $\gamma_{max} > 1$, the P_T can be rewritten as the sum of P_P and the product of three factors, namely $P_P, (1 - \gamma_{max})$ and $f_S(X,\delta)$. The purpose of the last two factors is to calculate the specific value of γ for a given customer. Note that the value γ_{max} specifies the *maximum* possible value

of γ, however, each customer is assigned a certain value from the range $[\gamma, \gamma_{max}]$ to keep the simulation realistic and allow for parametrization. As shown in Figure 4.5, by multiplying $f_S(X, \delta)$ (dashed lines) with γ_{max}, we receive functions that accept a random variable $X \sim \mathcal{U}(0, 1)$ and return a specific value of γ (full lines). Using the parameter δ, we can alter the shape of these functions, modifying the rate at which the value γ increases with the value of X. In analogy to Equation 4.15, we define the probability in relation to P_T as follows:

$$pbt_{accept}(R, P_A) = 0.0, \quad \text{if } P_A \geq P_T \tag{4.17}$$

3 At this point, we have specified both P_P and P_T, and thus know the upper and lower bounds of a customer's acceptance model. What remains is to specify the rate at which the probability decreases as the actual price P_A increases from P_P to P_T. To model this relationship, we will utilize a modified version of the scaling function $f_S(x, \delta)$ from Step 2. Let us redefine the function signature with distinct parameters to avoid ambiguity:

$$pbt_{accept}(R, P_A) = f_{S'}(\Delta_P, n, k) \tag{4.18}$$

where

- $f_{S'}(\Delta_P, \zeta, \eta) = 1 - (\Delta_P)^{\zeta^\eta}$ is the modified version of the **scaling function** from Step 2,
- $\Delta_P = \frac{P_A - P_P}{P_T - P_P} \in [0, 1]$ is the **price delta ratio**, which specifies the fraction of the "path" from P_P to P_T that P_A "lies" on,
- ζ is the **slope** of the **scaling function** f_S,
- η is the **slope magnifier**, which increases the effect of changes to n on the slope of the function.

For better visualization, Figure 4.6 depicts the shape of the scaling function for various values of ζ, η and Δ_P. In a similar fashion to the previous step, we can modify the rate at which the probability of acceptance of the price P_A increases with Δ_P through changes to ζ and η. While alternations to ζ have the same effect as *delta* in f_S, modifications to η will alter the magnitude of the effects that ζ has on the shape of the function (see blue and red lines for comparison).

Finally, we can define the last relationship required for forming the whole pricing function:

$$\begin{aligned} pbt_{accept}(R, P_A) &= f_{S'}(\Delta_P, \zeta, \eta) \\ &= 1 - (-(\frac{P_A - P_P}{P_T - P_P}) + 1)^{\zeta^\eta}, \quad \text{if } P_T \leq P_A \leq P_T \end{aligned} \tag{4.19}$$

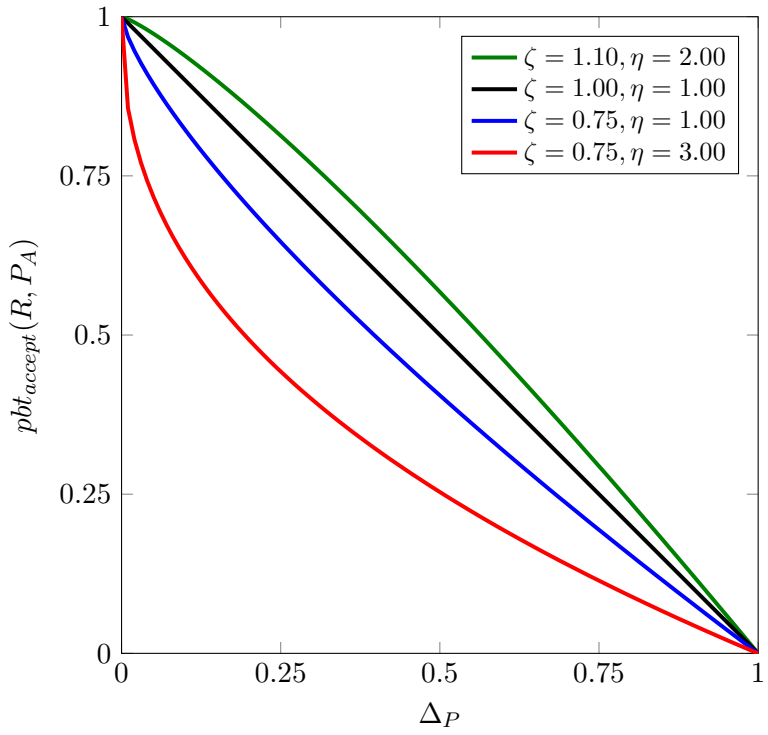

Figure 4.6: Shape of $f_{S'}(\Delta_P, \zeta, \eta)$ for different values of Δ_P, ζ, η

By combining all three relevant equations (4.15, 4.17, 4.19), we can define the acceptance function $pbt_{accept}(R, P_A)$ as:

$$pbt_{accept}(R, P_A) = \begin{cases} 1.00, & \text{if } P_A \leq P_P \\ 1 - (-(\frac{P_A - P_P}{P_T - P_P}) + 1)^{\zeta^\eta}, & \text{if } P_P < P_A < P_T \\ 0.00, & \text{if } P_A \geq P_T \end{cases} \quad (4.20)$$

where ζ, η are parameters as defined below Equation 4.18.

4.5.1 Using ζ, η for simulating variant price elasticity

While the acceptance model functions properly with fixed values of ζ and η, an opportunity arises for simulation of changes to customers' price elasticity depending on selected contextual factors, a few examples of which include:

- **time-to-arrival:** increasing price elasticity when bookings are made longer in advance to simulate customers' expectancy of lower prices (e.g. first-minute discounts), and decreasing it for last-minute bookings to simulate customers' acknowledgment of common hotel practices to increase the price as the arrival day closes in,

- **holidays and events:** decreasing price elasticity when a subset of booked nights falls on a date with higher demand (national holidays, conferences, sports events), to simulate customers' understanding of demand-and-supply curve,

- **group bookings:** increasing price elasticity when a booking is made for more than room, to simulate the customers' expectancy of a group discount

The simulation of variant elasticity can executed using the price multipliers that are already utilized in the pricing module. A multiplier evaluates the value of a factor passed in as input and returns a value that is used to multiply the price (from the hotel's point of view). To apply it on the customer, one can simply apply the multiplier on ζ, which alters the acceptance curve in a desired direction, effectively shifting the probability value for each Δ_P. A slight disadvantage is that due to the nature of $f_{S'}$, the adjustments made to probability values are of exponential nature (provided that $\zeta \neq 1$), rather than linear. This means, for example, that a 5 % decrease of ζ does not result in a 5 % decrease in probability for each Δ_P. Nevertheless, it is a problem that can be solved by multiplying the value of $f_{S'}(\Delta_P, \zeta, \eta)$ with the multiplier, rather than applying its value to ζ itself. This leaves room for future re-evaluation of the model setup so that adjustments to price elasticity can be made more comprehensible to the end user.

4.5.2 Modelling various customer segments

The layout of the acceptance model allows for simulation of various customer segments, such as leisure or business travellers. Since patterns of these segments can be roughly generalized, it is possible to estimate a price elasticity model for each of them.

4.6 Comparison to Previous Models

The main differences between our proposed model and previous works lie the pricing and acceptance modules. While all of the works used the multiplier-based pricing system, none have attempted to apply premiums or discount on any additional factors beyond the original four first presented in [23]. Our model implements the possibility of adding various types of linear, n-piecewise linear or constant multipliers, that are customizable based on the user's needs. As far as the acceptance model is concerned, previous works have used a model that is based on choosing a price with a 50 % probability of acceptance. As the price changes, the probability of acceptance changes in a sigmoidal fashion with the rate specified by a slope parameter. While it has proven to be an

accurate estimation, we think that it does not reflect the way customers perceive value, and have therefore presented an alternative which allows creation of a wider range of acceptance models.

4.7 Chapter Summary

In this chapter, we:

- presented the basic module structure and relevant terminology,
- explained the process of event generation for both reservations and cancellations,
- explained the multiplier-based pricing system and different types of multipliers,
- laid down the foundations of our novel acceptance module,
- summarized the differences between our model and previous works.

Chapter 5

Implementation

This chapter deals with the implementation details of the model. Firstly, we summarize the technologies and libraries that we used to implement the model. Secondly, a schema of the model is presented and the structure of the code is described. Thirdly, we explain the required structure of input data and ways it can be supplied to the model. Fourthly, we describe the algorithm used for simulating hotel processes. Finally, we describe the calculation of metrics based on results from the simulation process.

5.1 Used Technologies and Libraries

Python

Python is a dynamically typed programming language originally designed by Guido van Rossum. In spite of being over 30 years old, it is currently one of the most popular languages [31, 32] for both general development and data science. Our motivation behind the use of Python was the general accessibility of a wide variety of libraries for working with mathematical models and data through public domain via the The Python Package Index (PyPI), namely:

- **NumPy** [33] and **SciPy** [34], libraries targeted at scientific computation, which provide a variety of methods spanning from basic algebraic operations to complex operations on multi-dimensional arrays. In our work, we mostly utilized their implementations of statistical probability distributions, but they have been transitively used through other libraries.

- **Matplotlib** [35] and **Bokeh** [36], data visualization libraries that support creation of static, dynamic and interactive plots. We used them to visualize the results of simulations and experiments.

- **cma-es** [37], a ready-made implementation of the CMA-ES algorithm, which we used for optimization of pricing policy parameters.

JupyterLab and Jupyter Notebooks

JupyterLab is a web-based user interface for working with data and code within an integrated development environment [38]. Using the Jupyter Notebook App, it allows the user to create and edit interactive documents with markdown cells, executable live code and visualizations.

5.2 Model Structure

The structure of the model follows the one presented in [13] and can be depicted as in the following diagram:

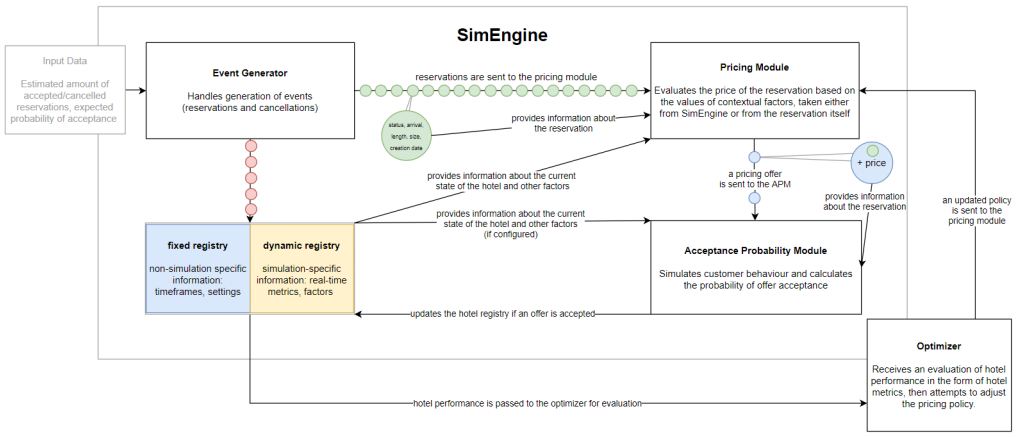

Figure 5.1: Diagram of the model's working process

5.2.1 Code Structure

The code of the software module is structured in the following way:

- **acceptance.py** contains definitions of two customer acceptance models:
  - **ReferenceAcceptanceModel**, an implementation of the model presented in previous works [13, 22],
  - **ProposedAcceptanceModel**, an implementation of our proposed model as defined in Chapter 4.5.
- **daterange.py** contains the **DateRangeList** class, a wrapper for the *date_range()* method of *pandas* for easier generation of datetime lists,

- **engine.py** containts the **SimEngine** class, which serves as the system controller; each instance represents an instance of a simulator; it stores the system configuration, contains wrapper methods to ease working with the model, and is responsible for running simulations,
- **events.py** contains methods for generation of events and cancellations, as well as the **Reservation** class (with an enum **ReservationStatus** for representing R_{status}),
- **inputs.py** contains classes that help facilitate data input (see 5.3),
- **multipliers.py** contains implementations of multiplier functions, namely:
  - **LinearMultiplier**, an implementation of the multiplier defined in Definition 4.4.4,
  - **PiecewiseLinearMultiplier**, an implementation of the multiplier defined in Definition 4.4.5,
  - several implementations of constant multipliers, as defined in Definition 4.4.6, namely:
    - **SpecificDateMultiplier**, a variant of the constant multiplier which applies a constant on a list of specific dates bound to a concrete year (e.g. March 21 - 23, 2016),
    - **PeriodicDateMultiplier**, a variant of the constant multiplier which applies a constant on a list of dates not bound to a specific year (e.g. March 21 - 23),
    - **WeekdayMultiplier**, a variant of the constant multiplier which applies a constant on a list of specific weekdays specified by their index (e.g. Friday, Saturday = [4, 5]),
- **pricing.py** contains the **PricingModule** class, which contains various method for price evaluations and working with multipliers,
- **report.py** contains an improvised data structure used for storing metrics.

5.3 Handling Input Data

The main prerequisite for proper functioning of the model is to supply it with input data. We designed our model to be able to process data in three distinct ways: via **a source file**, as **input parameters** using a collection of *InputDataEntry* objects, or by **random generation** (mostly for debugging purposes).

The classes that facilitate data provision are implemented in **inputs.py**. Here follows a brief description of the classes:

- ***InputData*** A wrapper class for input data which processes a chosen data source and generates a dataset during its instantiation. The dataset can then be retrieved using the class's *data* property.

- ***InputDataEntry*** A custom container class capable of generating variably-sized dataset segments based on its four input parameters (the average value, negative and positive deviations and time period). It is used to provide data points when passing input values as function parameters.

5.3.1 Loading input data from a source file

In order for the *InputData* class to recognize and correctly parse the contents of an input file, it must be written in one of two supported formats illustrated in Figure 5.2.

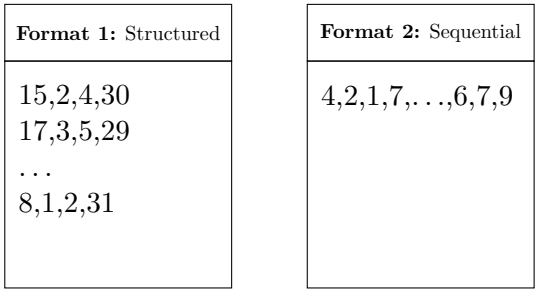

Figure 5.2: Example of input data files in structured and sequential formats

An input file with the structured format consists of n lines, each with four comma-separated integer values (corresponding to the average, negative deviation, positive deviation and period). The constructor of *InputData* parses the file line by line, splitting the line based on the comma delimiter and uses the parsed values to generate random data points within the specified bounds. An input file with the sequential format contains a single line with comma-separated integer values, each corresponding to a specific calendar year. These values are then saved as data points.

Note 5.3.1. While the principles of parsing these two formats are similar, they have distinct use cases. The sequential format is used when we want to process already existing data points, whereas the structured format is used to generate data based on compressed knowledge of average values and deviations.

The contents of the structured file above can be interpreted as follows: the first 30 days of the calendar year, there is an average amount of 15 events, a minimum amount of (15 - 2) events and a maximum amount of (15 + 1) events; the next 29 days of the calendar year, there is an average amount of 17 events ... (and so on).

The contents of the sequential file above can be interpreted as follows: on the first day of the calendar year, there are exactly 4 events; on the second day, there are 2 events ... (and so on).

5.3.2 Passing input data as function parameters

If the input data is not available in the form of a file, one can supply it by passing it to *InputData* as a parameter. More specifically, the data is passed as a tuple of *InputDataEntry* objects. On its instantiation, *InputData* it will sequentially retrieve the data segment from each of the tuple objects and joins them together to form the final dataset.

Note 5.3.2. In essence, the parameters passed to InputDataEntry are identical to the comma-separated values in the structured file format from the previous section. The generation of data based on these four values is identical, they only differ in the way they are supplied (read from a file vs. passed as input parameters wrapped in an *InputDataEntry* object).

5.3.3 Generating random datasets

For purposes of testing or debugging, it is also possible to generate random datasets. Although there is no advanced logic behind the generation, the user can adjust the values using three parameters: a deseasonaled average avg, a 12-element collection m of integer values representing monthly seasonal multiplier and a standard deviation value d. The generation process goes as follows: each day, the yearly average avg is multiplied by a corresponding monthly index m_i, then a random value is generated from the interval ($avg*m_i$ - d, $avg * m_i + d$).

5.4 Hotel Simulation Process

A single iteration of the hotel simulation process begins with the generation of a unique set reservations and concludes with the evaluation of performance using various hotel metrics. The implementation of this process in our work is described by the following algorithm:

1 First, we generate a unique set of reservations R.

   1 $R \leftarrow$ generate_reservations()

2 For each day d of the reservation and arrival windows ($RW \cup AW$):

2.1 If $d \geq AW_{start}$, process checkouts on day d:

```
checkouts ← 0 // number of checkouts
foreach r in R do
    if r_departure = d and r_status = Ongoing then
        r_status ← Completed
        checkouts ← checkouts + 1
    end
end
|R_ongoing| ← |R_ongoing| - c
|R_completed| ← |R_completed| + c
```

2.2 If $d \in RW$, evaluate reservation requests created on day d.
Let $R^d = \{r \in R \mid r_{cdate} = d\}$, then for each $r \in R^d$:

2.2.1. Evaluate the possibility to accommodate the request:

```
// Let r_dates = {d | r_arrival ≤ d < r_departure}
// Let F_d be the amount of free rooms on day d
minimum_capacity ← ∞
foreach d' in r_dates do
    if F_d' < minimum_capacity then
        minimum_capacity ← F_d'
    end
    if minimum_capacity < r_size then
        r_status ← Denied
        |R_denied| ← |R_denied| + 1
        continue // skip steps 2.2.2 - 2.2.3
    end
end
```

2.2.2. Calculate the price for the request:

```
r_price ← calculate_price() // see Chapter 4.3
```

2.2.3. Evaluate the acceptance of the proposed price:

```
accepted ← evaluate_acceptance(r_price)
if accepted = true then
    r_status ← Accepted
    |R_accepted| ← |R_accepted| + 1
    foreach d' in r_dates do
        F_d' ← F_d' − r_size
    end
else
    r_status ← Rejected
    |R_rejected| ← |R_rejected| + 1
end
```

2.3 Generate possible cancellations on day d:

```
// Let C_d be a set of r cancelled on day d
C_d ← generate_cancellations()
cancellations ← 0 // number of cancellations
foreach r ∈ C_d do
    cancellations ← cancellations + 1
    r_status ← Cancelled
    foreach d' in r_dates do
        F_d' ← F_d' + r_size
    end
end
|R_accepted| ← |R_accepted| − cancellations
|R_rejected| ← |R_rejected| + cancellations
```

2.4 Process arriving guests:

```
checkins ← 0 // number of checkins
foreach r in R do
    if r_arrival = d and r_status = Accepted then
        r_status ← Ongoing
        checkins ← checkins + 1
    end
end
|R_accepted| ← |R_accepted| - checkins
|R_ongoing| ← |R_ongoing| + checkins
```

2.5 Evaluate metrics for day d using techniques discussed in Section 5.5.

3 Summarize the performance of the hotel over the simulation period by calculating the average values of collected metrics over all n runs. If only one simulation run was executed, then the result was already calculated in Step 2.5.

5.5 Evaluating Performance

The metrics used for evaluation of hotel performance used in this bachelor thesis are:

- **total revenue up to day TR_d / daily revenue on day i DR_i**, calculated as:

$$TR_d = \sum_{i=1}^{d} DR_i = \sum_{i=1}^{d} \sum_{j=1}^{r} (R_j)'_{price} * (R_j)_{size} \tag{5.1}$$

where

  - R'_{price} is the average unit price ($\frac{R_{price}}{R_{length}}$),
  - r is the number of ongoing reservations on a given day i.

- **RevPAR**, calculated as $\frac{DR}{H_{max\_rooms}}$
- **ADR**, calculated as $\frac{DR}{H_{max\_rooms} - H_{free\_rooms}}$
- **occupancy**, calculated as $\frac{H_{max\_rooms} - H_{free\_rooms}}{H_{max\_rooms}}$.

Additionally on each day d, we collect:

- the average values of these metrics, calculated as the average of values collected up to day d, included,
- the amount of reservations in every possible state ($R_{accepted}, R_{denied} \ldots$).

5.6 Chapter Summary

In this chapter, we:

- summarized used technologies and libraries used in the implementation process,
- presented a schema of the model and described the code structure,
- explained the ways to supply input data and its required format,
- provided a detailed explanation of the algorithm used for simulating hotel processes,
- explained the process of calculating hotel metrics based on output data of the model.

Chapter 6

Experiments

In the following chapter, we describe the experiments used to evaluate the performance of our model, as well as its feasibility for future research. We begin with a description of the arrival dataset generation process. Secondly, we specify the details of the experimental setup. Thirdly, we summarize and visualize the results.

6.1 Configuration

6.1.1 Arrival Dataset

For the purposes of our experiments, we designed a synthetic dataset that aims to imitate arrivals and cancellations in a typical seaside destination. To generate a specific dataset, we first had to create an estimation model by specifying the minimum, average and maximum amounts of reservations and cancellations throughout a calendar year.

The resulting model is visualized in Figure 6.1 below.

Using these estimates, we generated the specific input dataset displayed in Figure 6.2. The dataset is characterized by several key components:

- a **high demand** (summer) **season**, spanning from mid-June to mid-September, with an average of over 25 requests per day,
- a **low demand** (winter) **season**, spanning from late November to late February, with less than 10 requests per day,
- a **medium demand season**, which includes dates outside of low and high seasons, with an average of 10 to 25 requests per day,
- **holidays and events**, represented by multi-day and single-day spikes in arrivals number, respectively.

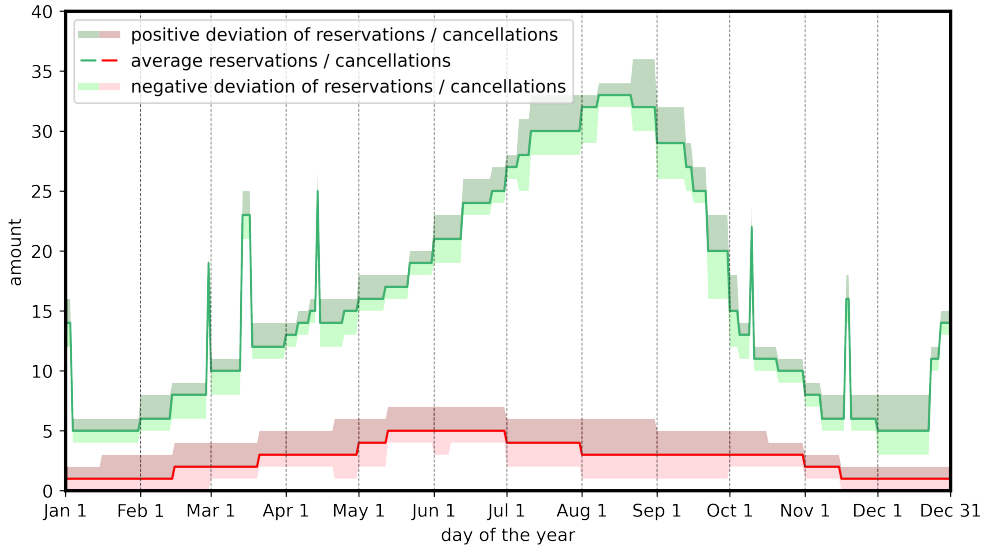

Figure 6.1: Estimates of request amounts used for input dataset generation

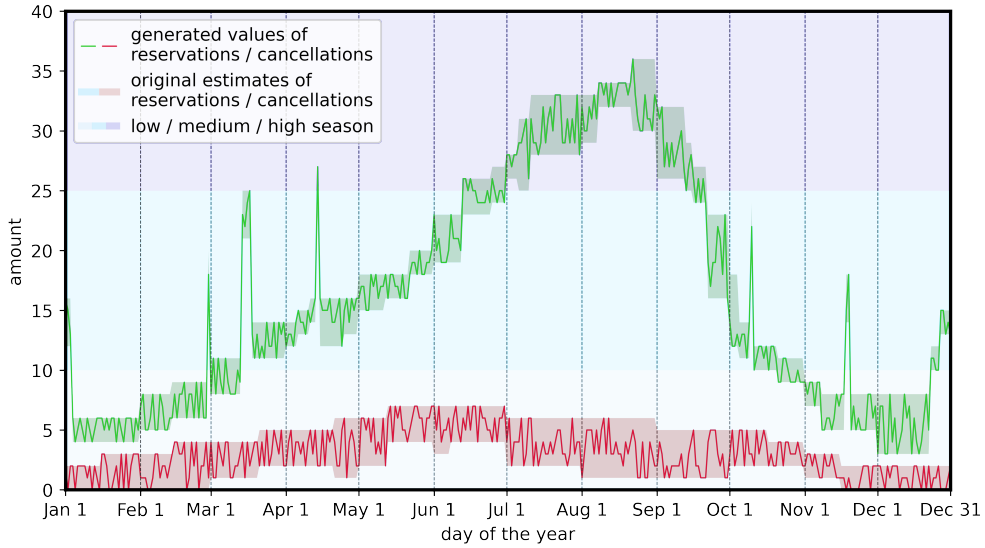

Figure 6.2: Generated requests with highlighted seasons and original estimates

6.1.2 Experimental Setup

Each of the experiments will be carried out in a set of three fictional hotels: H_{10}, H_{25} and H_{75}, with subscripts specifying the amount of rooms available at each establishment. These amounts were selected so that we can observe the effects of various policies under different capacity constraints. In our case, H_{10} is expected to be almost always fully occupied, H_{25} should be able to

satisfy all requests except during high season, and H_{75} should only run of out capacity in isolated cases. Similarly to [13], we model both reservations and cancellations so that 40 % of each happens on the arrival day, with the rest being distributed in a monotonically decreasing manner as displayed in Figure 6.3. The reservation and arrival windows overlap completely and span over the entire year of 2020. Bookings can be made up to 31 days in advance, can last for up to 14 days and include a maximum of 5 rooms per reservation. The probability of a group booking is set to 10 %.

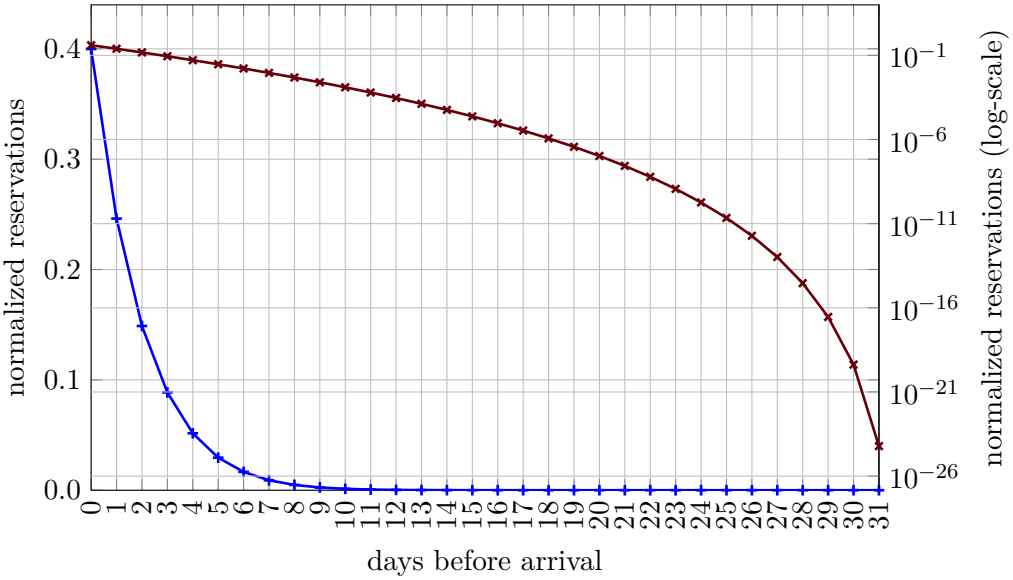

Figure 6.3: Occurence distribution of events in experiments

If not specified otherwise, the reference unit price P'_R will be set to 100. By default, customers select P_P from within up to 20 % of the actual price ($\alpha, \beta = 0.2$), and are willing to pay a maximum of 30 % ($\gamma = 1.3$) on top of P_P. They are slightly opposed to paying extra ($\delta = 0.8$), with just over half of the customers not willing to pay more than 10 % over their desired price. The additional parameters are selected as follows: $\epsilon = 0.01, \zeta = 0.8, \eta = 3.0$. Using these settings, the corresponding probabilities at $\Delta_P = 0.25, \Delta_P = 0.50$ and $\Delta_P = 0.75$ are approximately 50.8, 29.9 and 13.7 %, respectively.

6.2 Report

The following section is dedicated to summarization of conducted experiments. In all experiments, the average of 100 simulation runs was used and visualized. The optimization process was run on up to 100 generations, with each

generation taking the average result of 25 runs. We found this to present a balance between the length of the optimization process and the quality of results. With an average of ≈ 3.5 second per simulation, the optimization process took an average of 2.5 hours to complete.

Experiment 1

Measuring baseline model performance under default conditions

Objective

The objective of this experiment is to evaluate the performance of the model under general conditions as described in Subsection 6.1.2. The results from this experiment can then be used as a baseline for further comparison.

Pricing Policy

Fixed pricing (no multipliers), $P'_R = 100$.

Acceptance Model

Default, as described in Subsection 6.1.2.

Results

Let us begin our analysis with analyzing total revenue. According to Figure 6.4, the hotels managed to earn $TR(H_{10}) = 326700, TR(H_{25}) = 665400$ and $TR(H_{75}) = 896000$. The figure shows the evolution of total revenue growth over time. One can observe that the shape of the blue line (H_{10}) is similar to the one of a linear function. A similar pattern can be observed with the green line around high season. From our knowledge of the input dataset and limited capabilities of H_{10} and H_{25} to satisfy requests during high season, we would expect this to be an indicator of converge to their maximum capacities. H_{75}, which is known to deny a request only in exceptional cases, seems to exhibit unrestrained growth. Let us confirm this on a graph of daily revenue, shown in Figure 6.5.

A quick look at the figure confirms as our suspicions, as it can be observed that H_{10} was close to full occupancy throughout the whole year, H_{25} was reaching higher values between days 130 - 270 (roughly the same date indices as the ones in between which the green line in Figure 6.4 maintained linear shape), and H_{75} never reached its maximum capacity. Although this looks very positively for H_{75}, high total or daily revenue may be misleading. For this reason, we will now analyze revenue per available room (RevPAR), which is considered one of the most important hotel metrics, as it reflects the hotel's ability to maintain high revenue at high occupancy levels [12].

Figure 6.6 shows the evolution of RevPAR (semi-transparent lines) and its average (regular lines) over the arrival window. While the graph of H_{10}'s daily revenue per available room looks rather hectic, it maintained to hold a stable RevPAR throughout the whole period, reaching final average of 89.13.

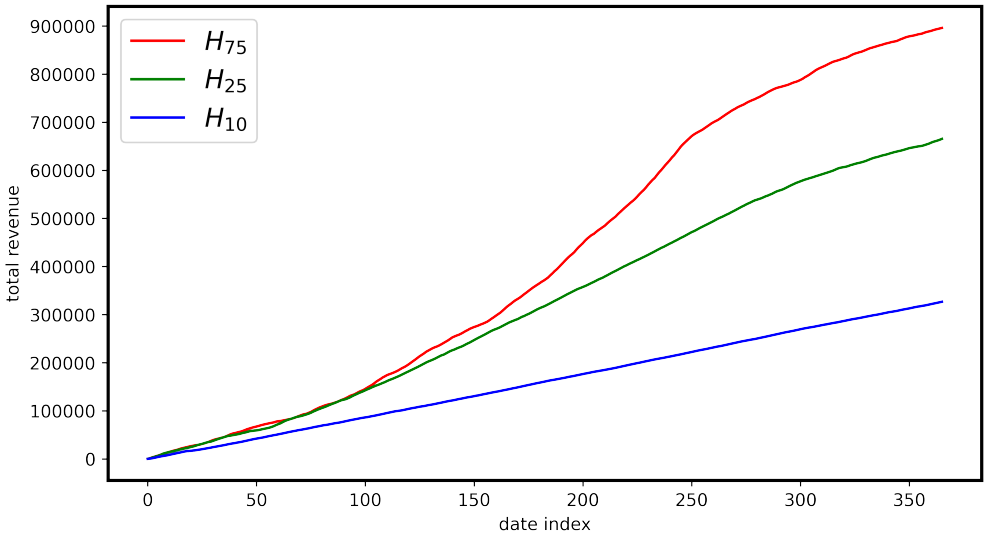

Figure 6.4: Measured values of total revenue (Experiment 1)

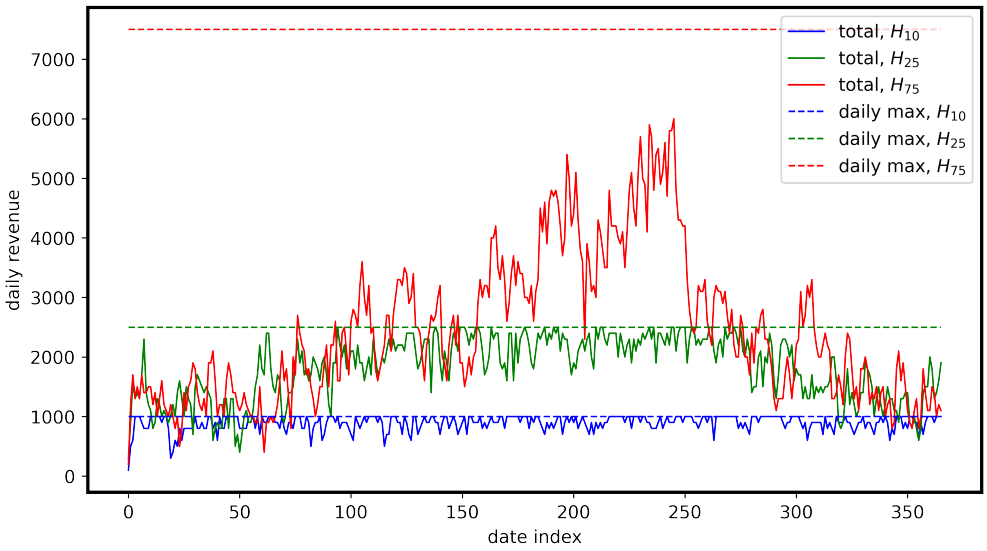

Figure 6.5: Measured values of daily revenue (Experiment 1)

H_{25} performed slightly worse in the beginning, but managed to increase its RevPAR during high season and finished with an average of 72.72. At last, H_{25} performed rather poorly, failing to occupy a vast majority of its rooms, reaching an underwhelming average of just 32.64. As a conclusion, we can state that there is a lot of potential for improvement through optimizing the pricing policy, especially in the case of the two larger hotels, which have around 37% and twice as many rooms to sell, respectively. The objective of H_{10}, which already reaches almost full occupancy, will have to rely on extra revenue

brought in by improved pricing rather than the sold amount of rooms, alone.

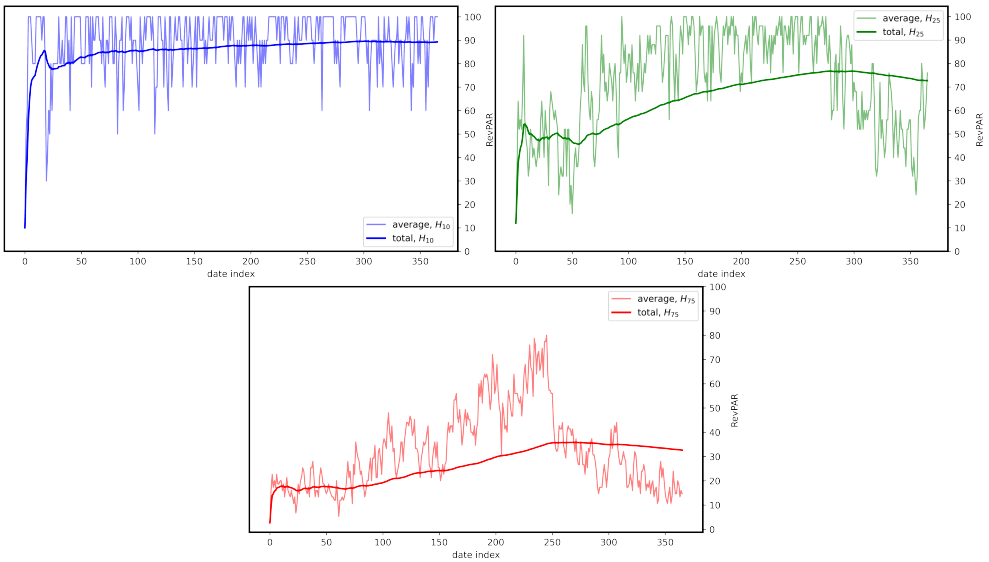

Figure 6.6: Measured values of RevPAR (Experiment 1)

Finally, we observe Figure 6.7, which shows us the normalized reservation amount depending on their status. The colors, when read from top to bottom, correspond to states in the following order: completed, ongoing, accepted, cancelled, rejected, denied. Looking at the graph for H_{10}, we can immediately observe rapid growth of denied reservations, caused by its restricted capacity. At its peak, their share reaches almost two thirds, converging to a final 59.71 %. The ratio of rejected reservations was 15.69 %, or 38.96 % when denied reservations are neglected. In the case of H_{25}, we do not begin to notice denial of reservations before high season, confirming its ability to handle low and mid season demand. The respective amounts, in the order mentioned above, reached 22.37, 30.20 and 38.91 %, respectively. Finally, we can see that there are no noticable denials in the case of H_{25} (in fact, the actual value is 1/100000th of a percent), and the ratios remain generally stable. The only relevant value, the ratio of rejected reservations, converged to 38.73 %. This indicated that the average rejection rate, in the case of our configuration, is $\approx$ 39 %.

The results collected and depicted in the previous graphs provide us with a good estimation of how our model performs in default situations. It allows us to compare the performance of enhanced methods and evaluate how much they have contributed to each of the relevant metrics. Let us summarize the observed values for better readability and ease of future comparison.

Experiment 2

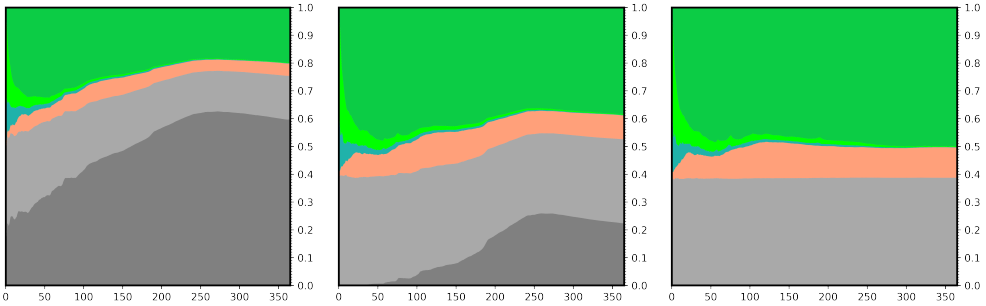

Figure 6.7: Normalized reservations based on states (Experiment 1)

Hotel	TR	$\overline{RevPAR}$	$\overline{ADR}$	$\overline{occupancy}$	% DR	% RRWOD
H_{10}	326700	89.13	100	89.13 %	59.71 %	38.96 %
H_{25}	665400	72.72	100	72.72 %	22.37 %	38.91 %
H_{75}	896000	32.64	100	32.64 %	>0.000001 %	38.73 %
DR = denied reservations, RRWOD = rejected reservations without denials						

Table 6.1: Summary of average hotel performance under default conditions

Measuring model performance with basic optimization

Objective

The objective of this experiment is to evaluate the magnitude of improvements reached by implementing price multipliers a letting an optimization engine optimize their boundaries and values by comparing the results of this experiment to the results of Experiment 1.

Pricing Policy

$P'_R = 100$, 4 multipliers: time, capacity, length and group (as per [23])

Acceptance Model

Default, as described in Subsection 6.1.2.

Results

Once again, we commence our analysis with evaluation of earned total revenue. Figure 6.8 shows the comparison of total revenue earned in both experiments. Looking at the graph, we can immediately notice marginal improvement in revenue collected by H_{25} and H_{25}, and a minor increase in the case of H_{10}. As we previously mentioned, there was a lot of room for improvement in the case of the two larger hotels, which has reflected in the results of this experiments. The hotels managed to increase their total revenue by 3.43, 8.04 and 20.29 %, respectively.

Figure 6.9 shows the ratio of daily revenue values between the current and previous experiment. Despite being fairly grainy, it can be observed that the

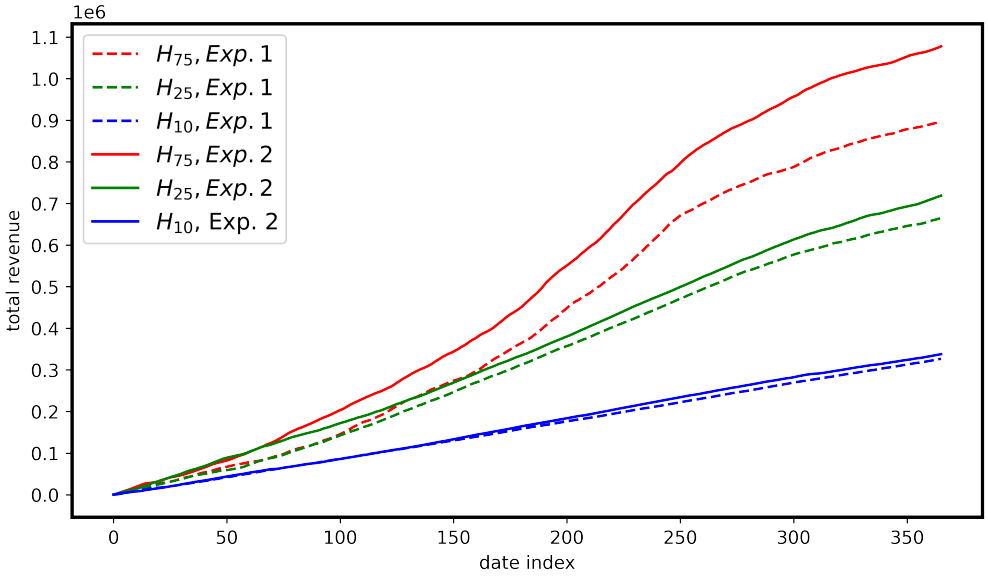

Figure 6.8: Measured values of total revenue (Experiments 1 vs. 2)

ratio remains mostly above the value of 1 (the black line), meaning that the optimized revenue is larger than the original value. The average ratios are 1.065, 1.167 and 1.318, respectively.

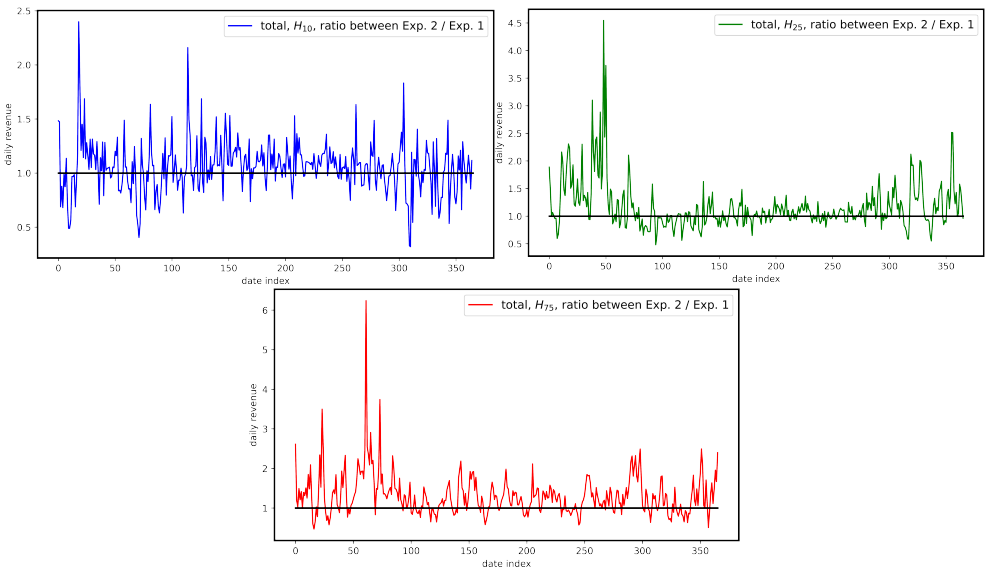

Figure 6.9: Ratio of daily revenue between Experiments 2 and 1

As far as RevPAR is concerned, we can observe improvements in all three hotels, roughly proposional to their size. As depicted in Figure 6.10, the increases in RevPAR go as follows: 89.13 to 92.33 for H_{10}, 72.72 to 78.57

for H_{25}, and 32.64 to 39.27 in case of H_{75}. Now that the daily rate is no longer fixed, observing the values of ADR and occupancy become relevant to understand what changes led to an increase of RevPAR. Since RevPAR is a multiplication of average daily rate and occupancy, this increase must necessarily mean either of the following:

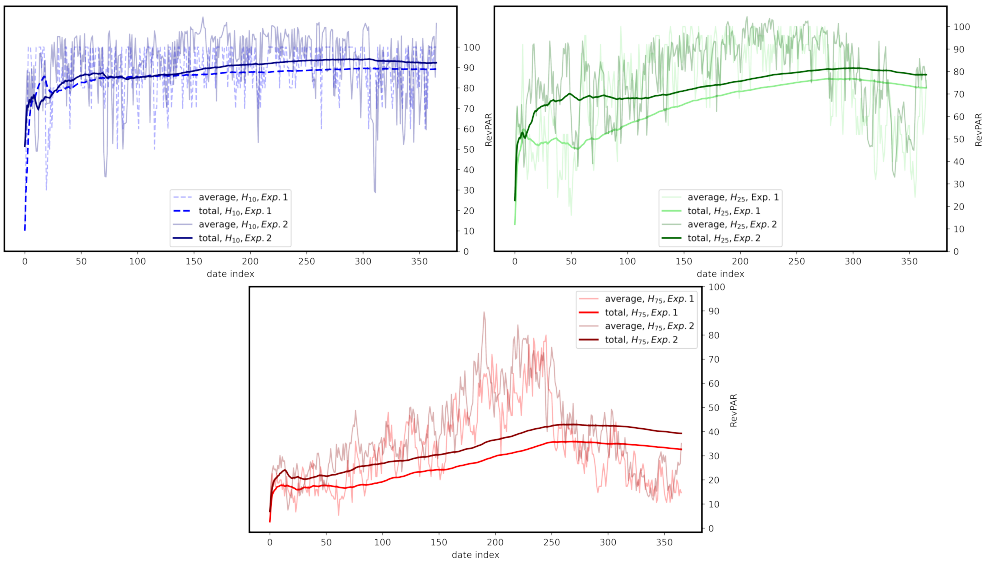

Figure 6.10: Measured values of RevPAR (Experiments 1 vs. 2)

- both ADR and occupancy have increased,
- either of the two increased, while the other kept its value,
- one of the values increased and the other decreases, but the ratio of these changes remained positive.

Looking at Figure 6.11, we immediately notice that H_{25} and H_{75} have decreased their daily rate, indicating that they sacrificed the cost for higher occupancy. This is logical, as they had an abundancy of free rooms they could sell. H_{10}, which was already nearly occupied, had to opt for an opposite approach, and increased its daily rate above 100 to make the most of its restricted situation.

This observation is confirmed by Figure 6.12, which shows that H_{25} and H_{75} have indeed gone the path of occupying more rooms, while H_{10}'s occupancy slightly decreases. The last thing that remains is to examine the ratio of reservations in order to evaluate the quality of the chosen strategy by each hotel.

Figure 6.13 shows the normalized reservations of Experiment 1 (top) compared to the values in Experiment 2 (bottom). The order of the hotels remains unchanged (H_{10}, H_{25}, H_{75}).

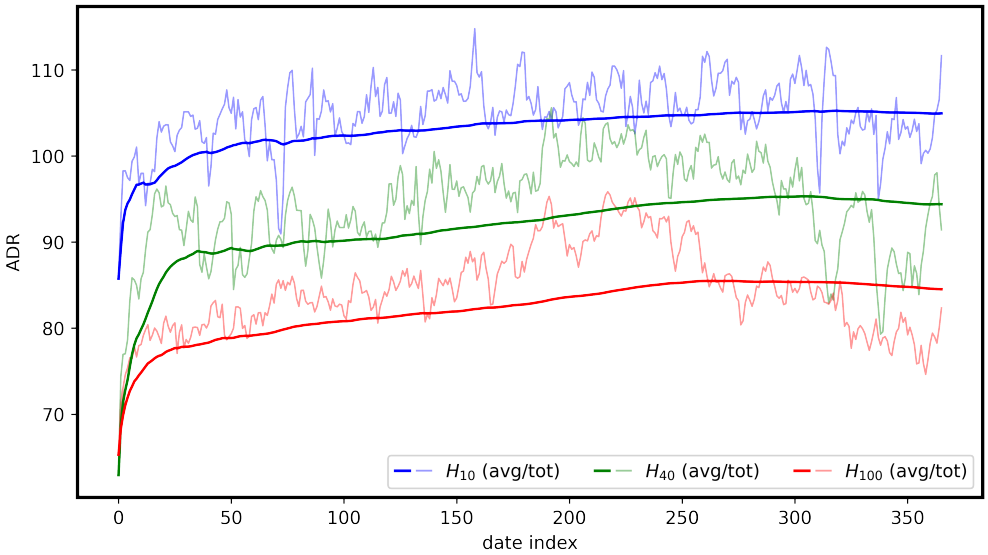

Figure 6.11: Measured values of average daily rate (Experiment 2)

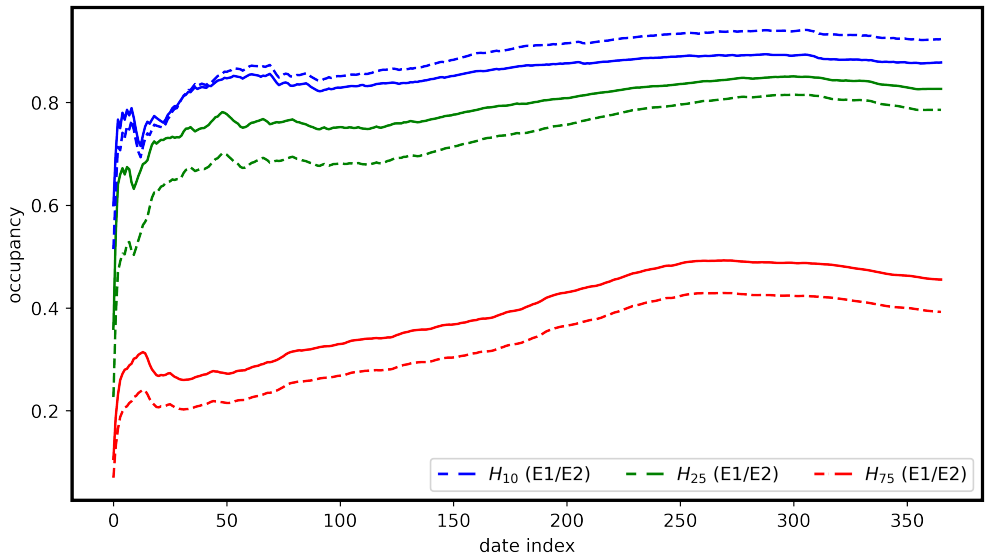

Figure 6.12: Measured values of occupancy (Experiment 1 vs. 2)

In the case of H_{10}, we registered a significant decrease of denied reservations from the previous 59.71 % to 38.89 %. On the other hand, the ratio of rejected reservations has increased from 38.96 % to 64.20 % (without denials). This is most probably caused by the hotel's approach to increasing revenue. The increase in daily hotel rate results in a decrease in occupancy, meaning that fewer reservations are denied by the hotel. At the same time, due to a higher price, more customers reject the pricing offer. H_{25} managed

to decrease the ratio of denied reservations from 22.37 % to 17.97 %, despite increasing its occupancy throughout the year. The rejection rate (without denials) remained roughly the same, only decreasing by ≈ 1.7 %. Finally, the largest difference can be seen in H_{75}, which decreased the amount of rejection from 38.73 % to 16.47 %, which allowed the ratio of completed reservation to almost double.

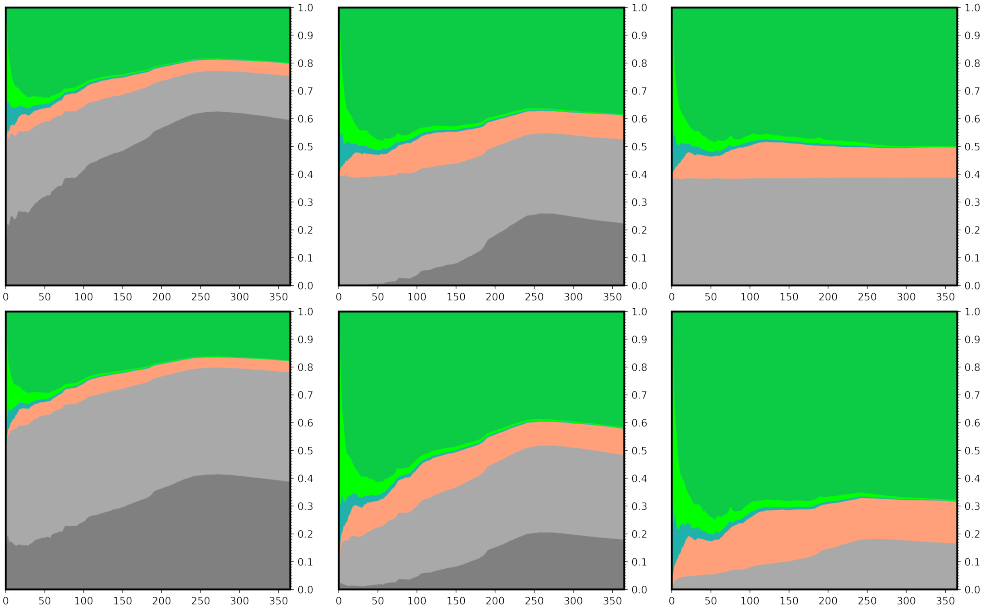

Figure 6.13: Normalized reservations based on states (Experiments 1 vs. 2)

For purposed of summarization, let us recreate the Table 6.1 from Experiment 1 and replace the values with the percentage change recorded, in order to summarize the results of our comparison.

Hotel	TR	$\overline{RevPAR}$	$\overline{ADR}$	$\overline{occupancy}$	% DR	% RRWOD
H_{10}	+3.43 %		+4.95 %	-1.50 %	-34.87 %	+64.81 %
H_{25}	+8.04 %		-5.61 %	+13.66 %	-19.65 %	-4.32 %
H_{75}	+20.29 %		-15.48 %	+39.52 %	>3590.25* %	-57.47 %
DR = denied reservations, RRWOD = rejected reservations without denials						
* = this looks like a large increase, but in fact equals less than 0.04 %						

Table 6.2: Percentage change in values compared to Experiment 1

6.3 Chapter Summary

In this section, we:

- explained the process of generating an experimental dataset,
- presented the details of the experimental configuration,
- conducted experiments that tested the performance of our model and summarized the observations.

Conclusion

This bachelor thesis deals with the application of mathematical and artificial intelligence methods in dynamic pricing of accommodation services. In the first chapter, I outlined the motivation for this research and established the following objectives:

- designing an algorithm capable of real-time pricing of hotel rooms through simulation of hotel processes,
- performing research of applications of mathematical and artificial intelligence methods in solving revenue management problems
- performing an analysis of existing solutions, and pointing out their strong and weak elements,
- suggesting improvements of the current solution based on a previous analysis,
- evaluating the viability of the suggested algorithm by conducting relevant experiments.

Firstly, I performed a literature review in which I analyzed relevant articles so as to gather information about the latest advancements in solving the issue at hand. The literature review focused on research of contemporary applications of both mathematical and artificial intelligence methods for solving various revenue management problems. Additionally, a critical analysis of existing hotel simulators was performed. Based on a thorough evaluation of previous solutions, I identified several possibilities for improvement. Namely, I put forward two enhancements for the use of price multipliers: for one, I suggested research of their applicability on a wider range of user-defined factors, for another, I proposed their utilization in scaling of the customer acceptance model. The principal enhancement advocated in this bachelor thesis was the proposed acceptance customer model, based on a novel approach

that attempts to model customer behaviour in a more natural and realistic way through use of a concept of choosing distinct price boundaries that symbolize preferred and unfavourable prices, as opposed to the generic solution presented in previous works. The model's feasibility for future research was evaluated on a set of two experiments, which compared the model's performance before and after optimization of price multiplier values. To measure the magnitude of improvement under different capacity constraints, I expanded the simulation to include three different-sized hotels with distinct capabilities of satisfying customer demand during different levels of season. The results of the experiments have shown that the model is capable of improving through optimization, achieving increases of 3.43, 8.04 and 20.29 % in total revenue. Additionally, I measured an increase of 13.66 and 39.52 % in occupancy for the two larger establishments, while the smallest hotel managed to increase its revenue without having to resort to making any compromises regarding the average daily rate.

Both the main goal and subtasks of this bachelor this are considered fulfilled. I successfully performed a research of contemporary applications of artificial intelligence in hotel revenue management, and dynamic pricing. An analysis of existing solutions was performed, and a novel approach was proposed based on observations of strengths and weaknesses of previous solutions. Finally, experiments were carried out to evaluate the model's potential, and the results are considered to have laid enough foundation for future research.

I consider my achieved results as a sign of the model's potential and motivation for further experimentation. I am confident that through further research, the proposed idea can improve the quality of hotel simulators and replace the limiting implementation that currently dominates. Due to the limiting scope of bachelor theses, I chose not to pursue a more in-depth analysis of the proposed changes. However, I plan to further explore this topic and my proposed improvements as part of *Výzkumné léto na FIT (VýLET)*, a research program organized by CTU, and potentially in the form of a master thesis.

The possibilities of further research include:

- proper analysis of the proposed model, including its evaluation on real hotel data, which was not available at the time of writing this thesis,
- exploration of possibilities of simulating multiple types of rooms, customers or hotels co-existing in a single environment,
- further improvements to the customer acceptance model based on research of papers that analyze real-life customer behaviour patterns.

Bibliography

[1] UNWTO. Global and Regional Tourism Performace. 2020, available: https://www.unwto.org/global-and-regional-tourism-performance (accessed on 2021-03-20).

[2] WTTC. Share of GDP generated by the travel and tourism industry worldwide from 2000 to 2019. 2020, available: https://www.statista.com/statistics/1099933/travel-and-tourism-share-of-gdp (accessed on 2021-03-20).

[3] Knoema. Contribution of travel and tourism to GDP as a share of GDP (%). 2020, https://knoema.com/atlas/topics/Tourism/Travel-and-Tourism-Total-Contribution-to-GDP/Contribution-of-travel-and-tourism-to-GDP-percent-of-GDP.

[4] WTTC. Economic Impact Report 2020. 2020. Available from: https://wttc.org/Research/Economic-Impact

[5] UNWTO. 2020: Worst Year in Tourism Industry with 1 Billon Fewer International Arrivals. 2021. Available from: https://www.unwto.org/news/2020-worst-year-in-tourism-history-with-1-billion-fewer-international-arrivals

[6] UNWTO. Impact Assessment of the COVID-19 Outbreak on International Tourism. 2020. Available from: https://www.unwto.org/impact-assessment-of-the-covid-19-outbreak-on-international-tourism

[7] Bughin, J.; Hazan, E.; et al. Artificial intelligence: The next digital frontier? 2017. Available from: https://apo.org.au/node/210501

[8] Perrault, R.; Shoham, Y.; et al. Artificial Intelligence Index 2019 Annual Report, 2019. *Human-Centered Artificial Intelligence Institute, Stanford,*

California: Stanford University, December, 2019. Available from: https://euagenda.eu/upload/publications/untitled-283856-ea.pdf

[9] HMS, O. How many hotels are there in the world? 2019. Available from: https://medium.com/@info_77326/how-many-hotels-are-there-in-the-world-8ee983295b91

[10] Kimes, S. E. The basics of yield management. *Cornell Hotel and Restaurant Administration Quarterly*, volume 30, no. 3, 1989: pp. 14–19. Available from: https://doi.org/10.1177/001088048903000309

[11] Kimes, S. E.; Wirtz, J. Has revenue management become acceptable? Findings from an international study on the perceived fairness of rate fences. *Journal of service research*, volume 6, no. 2, 2003: pp. 125–135. Available from: https://doi.org/10.1177/1094670503257038

[12] Ivanov, S. *Hotel revenue management: From theory to practice*. Zangador, 2014. Available from: https://doi.org/10.13140/RG.2.1.2714.8960

[13] Mariello, A.; Dalcastagné, M.; et al. HotelSimu: Simulation-Based Optimization for Hotel Dynamic Pricing. In *International Conference on Learning and Intelligent Optimization*, Springer, 2020, pp. 341–355.

[14] Kimes, S. E. The evolution of hotel revenue management. *Journal of Revenue and Pricing Management*, volume 15, no. 3, 2016: pp. 247–251.

[15] Al Shehhi, M.; Karathanasopoulos, A.; et al. Forecasting hotel prices in selected Middle East and North Africa region (MENA) cities with new forecasting tools. *Theoretical Economics Letters*, volume 8, no. 09, 2018: p. 1623.

[16] Al Shehhi, M.; Karathanasopoulos, A. Forecasting hotel room prices in selected GCC cities using deep learning. *Journal of Hospitality and Tourism Management*, volume 42, 2020: pp. 40–50.

[17] Chen, K.-Y. Combining linear and nonlinear model in forecasting tourism demand. *Expert Systems with Applications*, volume 38, no. 8, 2011: pp. 10368–10376. Available from: https://doi.org/10.1016/j.eswa.2011.02.049

[18] Shakya, S.; Chin, C. M.; et al. An AI-based system for pricing diverse products and services. In *Research and Development in Intelligent Systems XXVI*, Springer, 2010, pp. 393–406.

[19] Shakya, S.; Kern, M.; et al. Neural network demand models and evolutionary optimisers for dynamic pricing. *Knowledge-Based Systems*, volume 29, 2012: pp. 44–53, doi:10.1016/j.knosys.2011.06.023. Available from: https://doi.org/10.1016/j.knosys.2011.06.023

[20] Figueira, G.; Almada-Lobo, B. Hybrid simulation–optimization methods: A taxonomy and discussion. *Simulation Modelling Practice and Theory*, volume 46, 2014: pp. 118–134.

[21] Zakhary, A.; Atiya, A. F.; et al. Forecasting hotel arrivals and occupancy using Monte Carlo simulation. *Journal of Revenue and Pricing Management*, volume 10, no. 4, 2011: pp. 344–366.

[22] Brunato, M.; Battiti, R. Combining intelligent heuristics with simulators in hotel revenue management. *Annals of mathematics and artificial intelligence*, volume 88, no. 1, 2020: pp. 71–90.

[23] Bayoumi, A. E.-M.; Saleh, M.; et al. Dynamic pricing for hotel revenue management using price multipliers. *Journal of Revenue and Pricing Management*, volume 12, no. 3, 2013: pp. 271–285.

[24] Hansen, N. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[25] Brunato, M.; Battiti, R. The reactive affine shaker: a building block for minimizing functions of continuous variables. 2006.

[26] Brunelli, R.; Tecchiolli, G. P. Stochastic minimization with adaptive memory. *Journal of Computational and Applied Mathematics*, volume 57, no. 3, 1995: pp. 329–343.

[27] Harrison, R. L. Introduction to monte carlo simulation. In *AIP conference proceedings*, volume 1204, American Institute of Physics, 2010, pp. 17–21.

[28] Raychaudhuri, S. Introduction to monte carlo simulation. In *2008 Winter simulation conference*, IEEE, 2008, pp. 91–100.

[29] Yu, X.; Gen, M. *Introduction to evolutionary algorithms*. Springer Science & Business Media, 2010.

[30] Yager, R. R. Quantifier guided aggregation using OWA operators. *International Journal of Intelligent Systems*, volume 11, no. 1, 1996: pp. 49–73.

[31] PYPL PopularitY of Programming Language Index. Accessed: 2021-06-24. Available from: `https://pypl.github.io/PYPL.html`

[32] The TIOBE Programming Community Index. Accessed: 2021-06-24. Available from: `https://www.tiobe.com/tiobe-index/`

[33] Harris, C. R.; Millman, K. J.; et al. Array programming with NumPy. *Nature*, volume 585, 2020: p. 357–362, doi:10.1038/s41586-020-2649-2.

[34] Virtanen, P.; Gommers, R.; et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, volume 17, 2020: pp. 261–272, doi:10.1038/s41592-019-0686-2.

[35] Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, volume 9, no. 3, 2007: pp. 90–95, doi:10.1109/MCSE.2007.55.

[36] Bokeh Development Team. *Bokeh: Python library for interactive visualization*. 2018, accessed: 2021-06-24. Available from: `https://bokeh.pydata.org/en/latest/`

[37] Strock, R. A Tensorflow v2 Python Implementation of Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Accessed: 2021-06-24. Available from: `https://pypi.org/project/cma-es/`

[38] Jupyter, P. JupyterLab: Documentation. Accessed: 2021-06-24. Available from: `https://jupyterlab.readthedocs.io/en/latest/`

Appendix A

Acronyms

ADR average daily rate

AI artificial intelligence

BB bed breakfast

CMA-ES Covariance Matrix Adaptation Evolution Strategy

NN neural network

RevPAR revenue per available room

RMSE root-mean-square error

RD research development

UNWTO United Nations World Tourism Organization

Appendix B

Enclosed Media Contents

- `README.md` ..... instructions about the structure of the media
- `src` ..... files related to the software module
  - `code` ..... software module codebase
    - `input` ..... input data from experiments
  - `measurements` ..... raw measurement from the experiments
- `text` ..... files related to the thesis text
  - `BP_Bacigal_Michal_2021.pdf` ..... generated PDF of this thesis
  - `thesis_assignment.pdf` ..... PDF of the thesis assignment
    - `src` ..... source files for the thesis text
      - `assets` ..... graphs and schemes from the text
      - `sections` ..... separated tex chapters of the thesis