Kaunas University of Technology

Faculty of Informatics

Machine Learning Algorithm Application in Trip Planning

Master's Final Degree Project

Grantas Gadliauskas

Project author

Doc. Dr. Andrius Kriščiūnas

Supervisor

Kaunas, 2022

Kaunas University of Technology

Faculty of Informatics

Machine Learning Algorithm Application in Trip Planning

Master's Final Degree Project

Software Engineering (6211BX011)

Grantas Gadliauskas

Project author

Doc. Dr. Andrius Kriščiūnas

Supervisor

Doc. Dr. Agnė Paulauskaitė-Tarasevičienė

Reviewer

Kaunas, 2022

Kaunas University of Technology

Faculty of Informatics

Grantas Gadliauskas

Machine Learning Algorithm Application in Trip Planning

Declaration of Academic Integrity

I confirm the following:

1. I have prepared the final degree project independently and honestly without any violations of the copyrights or other rights of others, following the provisions of the Law on Copyrights and Related Rights of the Republic of Lithuania, the Regulations on the Management and Transfer of Intellectual Property of Kaunas University of Technology (hereinafter – University) and the ethical requirements stipulated by the Code of Academic Ethics of the University;

2. All the data and research results provided in the final degree project are correct and obtained legally; none of the parts of this project are plagiarised from any printed or electronic sources; all the quotations and references provided in the text of the final degree project are indicated in the list of references;

3. I have not paid anyone any monetary funds for the final degree project or the parts thereof unless required by the law;

4. I understand that in the case of any discovery of the fact of dishonesty or violation of any rights of others, the academic penalties will be imposed on me under the procedure applied at the University; I will be expelled from the University and my final degree project can be submitted to the Office of the Ombudsperson for Academic Ethics and Procedures in the examination of a possible violation of academic ethics.

Grantas Gadliauskas

Confirmed electronically

Kaunas University of Technology

Faculty of Informatics

Task of the Master's final degree project

Topic of the project: Machine learning algorithm application in trip planning.

Requirements and conditions (title can be clarified, if needed):

Supervisor

(position, name, surname, signature of the supervisor) (date)

Gadliauskas Grantas. Machine learning algorithm application in trip planning. Master's Final Degree Project / supervisor lect. Andrius Kriščiūnas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Software Engineering.

Keywords: travelling salesman problem, flight search, combinatorial optimization, neural network.

Kaunas, 2022. 55 p.

Summary

This paper describes a software system module for a travel trip planning solution and the research conducted into how machine learning can be applied to optimally generate trip deals. The solution is a unique one, since it requires no direct input from the user to generate the trip results and is solely driven by the created algorithm. It allows for its end users to save time and hassle of manually searching for flights between multiple destinations by offering attractive prepared trip offers.

The research part of the work explores how machine learning can be applied in efficiently solving a variation of the Travelling Salesman Problem (TSP) in the context of air travel tourism. Large number of cities create too many trip route combinations to be efficiently evaluated in real time. The method proposed uses a feedforward neural network to narrow down the number of trip route combinations, while a more traditional algorithm based on dynamic programming is then able to select the best trip offers. It was shown that the method can be applied in practice to achieve almost real-time generation of best possible trip offers while evaluating a large amount of real-world flight data.

Gadliauskas Grantas. Mašininio mokymosi algoritmų taikymas kelionių planavime. Magistro baigiamasis projektas / vadovas lekt. Andrius Kriščiūnas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Programų sistemos.

Reikšminiai žodžiai: keliaujančio pirklio uždavinys, skrydžių paieška, kombinatorinė optimizacija, neuroninis tinklas.

Kaunas, 2022. 55 p.

Santrauka

Šiame darbe aprašomas programų sistemos modulis, skirtas kelionių platformos sistemai, bei tyrimas siekiantis išnaudoti mašininio mokymosi galimybes sudaryti patrauklius kelionių rinkinius. Kuriamas sprendimas unikalus tuo, jog kelionių planai potencialiems klientams, norintiems vienos kelionės metu aplankyti daugiau nei vieną šalį, būtų generuojamos ne rinkoje įprastu būdu, kai vartotojas pasirenka konkrečias kryptis, o pateikiamos pagal sukurto algoritmo rezultatus. Toks sprendimas leistų galutiniams vartotojams sutaupyti laiką, reikalingą tinkamų skrydžių tarp skirtingų vietovių suradimui, o kartu pateiktų visą aibę galimų kelionių rinkinių.

Darbo tyrimo ir eksperimentų dalis siekia ištirti, kaip mašininis mokymasis gali būti panaudotas išspręsti keliaujančio pirklio uždavinį (*angl. travelling salesman problem; TSP*) oro kelionių kontekste. Dėl didelio kiekio miestų yra sunku efektyviai patikrinti visas kelių kombinacijas. Siūlomas metodas naudoja dirbtinį neuroninį tinklą susiaurinti kombinacijų aibę, kas leidžia dinaminiu programavimu grįstam algoritmui pasirinkti geriausius kelionių pasiūlymus. Tyrime parodoma, kad šis metodas gali būti pritaikomas praktikoje generuoti kelionių pasiūlymus realiu laiku įvertinant didelį kiekį tikrų skrydžių duomenų.

Table of contents

List of figures

List of tables

List of abbreviations and terms

Abbreviations:

ML – machine learning;

TSP – travelling salesman problem;

DP – dynamic programming;

FNN – feed-forward neural network;

CNN – convolutional neural network;

OSM PIS – OpenStreetMap place importance score;

MAE – mean absolute error.

Terms:

Artificial neural network (neural network) - computational model that consists of several processing elements that receive inputs and deliver outputs, inspired by the biological neural networks that constitute animal brains.

Heuristic - any approach to problem solving or self-discovery that employs a practical method that is not guaranteed to be optimal, perfect, or rational, but is nevertheless sufficient for reaching an immediate, short-term goal or approximation.

Introduction

An influx of low-cost airlines in the recent years made a huge impact to the success of the travel industry, which was ultimately shaken by the global pandemic at the start of 2020. While many big-name market players took a huge hit, an exciting environment has opened for new innovative solutions which could help to replenish the state of the industry in the near future.

This project aims to create a travel trip planning solution which leverages aggregated flight data and the latest advancements in the machine learning field. A trip may consist of visiting multiple travel locations spanning multiple countries. The solution is a unique one, since it requires no direct input from the user to generate the trip results and is solely driven by the created algorithm which analyses many travel trip related criteria. The solution allows for its end users to save time and hassle of manually searching for flights between multiple destinations by offering attractive prepared trip offers.

1. Analysis

1.1. Problem Overview

Consider a tourist who wants to visit several different cities in a specific date range in a round trip from his home city. The tourist might also have preferences to which cities one wants to visit or avoid. A list of *N* best possible trip offers then should be provided to the user, based on the real-world flight data. The quality of the trip is determined by its price, but additional metrics could be added.

Since flight data updates very often and the number of possible date ranges is immensely huge it is not practical to pre-calculate all the offers. On the other hand, finding the best offers in real-time is inefficient due to the need to compute the best scored combination of flights for a large amount of possible trip routes.
In the combinatorial optimization domain, the more simplified version of this problem is well known as the Travelling Salesman Problem (TSP) [1]. The applications of TSP and its variants are used globally for logistics, planning, astronomy, and manufacturing. The aim of TSP, given a context of a country, is to find the shortest possible route that visits each city once and returns to the origin city. Since 1976, when Nicos Christofides came up with an algorithm [2] that finds round trips no longer than 50% longer than the best round trip, no significant progress in effectiveness was made [3].
More recent works on the topic include machine learning approaches such as one by Chaitanya K. *et al.* [4] which makes use of neural networks to perform TSP efficiently with hundreds of nodes. For our problem, however, the number of nodes (possible trip flights) will never be more than a few hundred, but the more important issue is the number of trip routes growing polynomially because of the number of different cities.

1.2. Trip design optimization overview

1.2.1. Tourist trip design problem

The Tourist Trip Design Problem (TTDP) is a variant of TSP and refers to a route-planning problem for tourists interested in visiting multiple points of interest (POIs). TTDP implementations usually derive daily tourist tours, for example, ordered visits to POIs, which respect tourist constraints and POIs attributes. The algorithmic approaches for solving TTDP variants represents the most crowded field of research among them. [5]
Two popular TTDP variants exist based on the amount of days the tourist has to stay:

- single tour TTDP variants aim to find a single tour that maximizes the collected profit while respecting certain tourist constraints and POI attributes;
- multiple tour TTDP variants aim to find multiple tours based upon the number of days the tourist's visit will last.

Single tour variants of the TTDP can be modeled using TSPP, a bicriteria generalization of TSP. Specifically, in TSPP a network is given in which nodes are associated with profits and links with travel costs, and the goal is to find a tour (which starts and ends at a specified node -

the depot) over a subset of nodes such that the collected profit is maximized while the travel cost is minimized. [5]

TTDP variants with time windows (TW) considers visits to locations within a predefined time window (this allows modeling opening and closing hours of POIs, for example, flight times). The time-dependent (TD) variants consider time dependency in the estimation of time required to move from one location to another. [5]

1.2.2. Orienteering problem with time windows

The Orienteering Problem with Time Windows (OPTW) can be seen as a model for the Tourist Trip Design Problem (TTDP). [6]
In the OPTW, a collection of locations is given, each with a score that denotes its attractiveness. The time needed for travelling from location *i* to *j*, is known for all locations. The goal of the problem is to maximize the sum of the scores of the selected locations, keeping the total time of the route between these locations under a given time budget or the total distance of the route under a given distance budget. [7]. Time window constraint introduces a start time and an end time for each location, indicating the time when these locations can be visited (Figure 1).

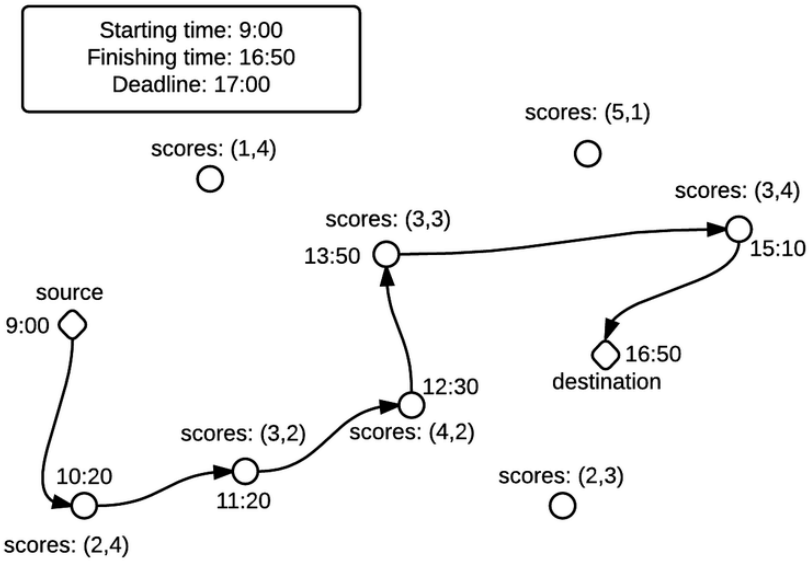

Figure 1. An example of a solution of the multi-objective time-dependent orienteering problem [8]

In the OPTW it is assumed that a route's starting location, its starting time and ending time, and scores of the points of interest are tourist dependent, while the coordinates of the points of interest, their opening and closing times, and duration of visit are not. [6]

1.2.3. Review

Despite its tight relation with the TSP, our problem described in chapter 1.1 concerning commercial flights differs from the alternatives suggested above. The approaches consider that the cost between two cities (nodes) is always constant over time, although assumption is certainly not true for the case of commercial flights, as the tickets price depend not only on the date, but also on the direction of the route. Also, the approaches assume that the waiting period in each city is a static time period, which in our case is not convenient for the traveler.

1.3. Artificial intelligence overview

The field of artificial intelligence (AI) is currently on the rise, although it has been around for more than six decades. The difference between current decades and the previous decades, is that the AI research promise actually materializes when solving real-world problems. [9] The most notable of these problems include speech recognition, computer vision, bio-surveillance, robot or automation control and empirical science experiments [10].

Supervised learning is one of the most popular paradigms of machine learning, where the machine is given a dataset (for example, a set of data points), along with the right answers to a question corresponding to the data points (labels). The learning algorithm is provided with a huge set of data points with answers, for example, a labelled dataset. The algorithm has to learn the key characteristics within each data point in the dataset to determine the answer. So, next time a new data point is provided to the algorithm, based on the key characteristics, the algorithm should be able to predict the outcome/right answer. [9]

1.3.1. Artificial neural networks

Supervised learning falls mostly in two categories: solving of classification and regression problems. Depending on the category, regression analysis is performed:

- Linear regression allows to predict value of a continuous variable to solve regression problems. It assumes that the relationship between the variables can be expressed as a linear function.

- Logistic regression allows to predict whether or not a specific outcome would be achieved. It provides a 0 or 1 prediction, rather than a real value for a continuous variable.

Neural networks are networks of interconnected artificial neurons. Their structure is heavily inspired by the brain's neuron network. A neural network is generally used to create supervised machine learning models. It can be thought of as combining multiple regression models to make a more powerful model.

The feedforward neural network was the first and simplest type of artificial neural network devised [11]. In this network, the information moves in only one direction—forward—from the

input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. [12]

Convolutional neural networks are used for some of the most interesting applications of machine learning, such as image recognition, handwriting reading, interpreting street signs, etc. Convolution is a technique which automates extraction and synthesis of significant features needed to identify the target classes. CNN is usually composed of multiple layers of convolution and pooling combination and then followed by a neural network. [9]

1.3.2. Review

Since artificial intelligence is a very promising field, the work described in this paper puts a big focus on finding a machine learning technique that would prove to be more efficient than the more traditional algorithms. Supervised learning techniques using artificial neural networks will be used in the research of this work, since to generate high quality trips, we want the ML model to be able to describe the quality of the trip (either through regression or classification). ML model can be trained using labeled data, which can be generated using a simpler brute force based algorithm when the speed of generation is not so important.

2. Software system

2.1. Application

2.1.1.1. Demand

2.1.1.2. Project users and clients

The main clients of the solution are travel platforms and agencies. Individual travel guides should the find the solution valuable as well. The final end users are all people who have an interest in travelling and tourism.

2.1.1.3. User problems

Tourists face a difficulty when trying to find a likeable trip. With traditional trip planning platforms, user must select and combine flights into a single trip manually, which usually is not optimal in terms of price and quality.

2.1.1.4. Market research

According to the latest World Travel & Tourism Council (WTTC) data [13], tourism industry accounts for 10.4% of the world's GDP and is one of the largest economic sectors in the world. Tourism industry which in 2018 contributed to 8.8 trillion to the global economy [14] is viewed as one of the most growing industries.

The biggest names in the industry are:

- Kayak – metasearch engine for travel flights. Over 6 billion queries each year. Property of Booking Holdings, the world leader in online travel [15].
- TripAdvisor – travel shopping comparison and user review website. Number 1 in the Travel and Tourism category in the US, revenue of $1.62 billion worldwide and 490 million monthly active users as of 2019 [16].
- Kiwi – flight search website. 100M average daily search queries, €1.3 turnover in 2019 [17].

These companies offer common traditional trip planning solutions and thus could be viewed as potential clients rather than competitors.

2.1.1.5. Information about the clients

Travel platform and agencies seeking to please their own users with attractive trip suggestions. Travel guides and related businesses which do not have enough resources to create qualitative travel offers or want to focus mainly on client support. The clients will be able to access the service by purchasing an application programming interface (API) subscription of the created solution.

2.1.1.6. Product description

The product can be described as an API service which gives its users an ability to query attractive travel trip suggestions. API service is sold through a monthly subscription model or by each successful trip purchase from the end users.
Figure 2 describes how a potential client of the product, a travel agency platform, could leverage the API service to increase its conversions by providing attractive trip suggestions to its end users.

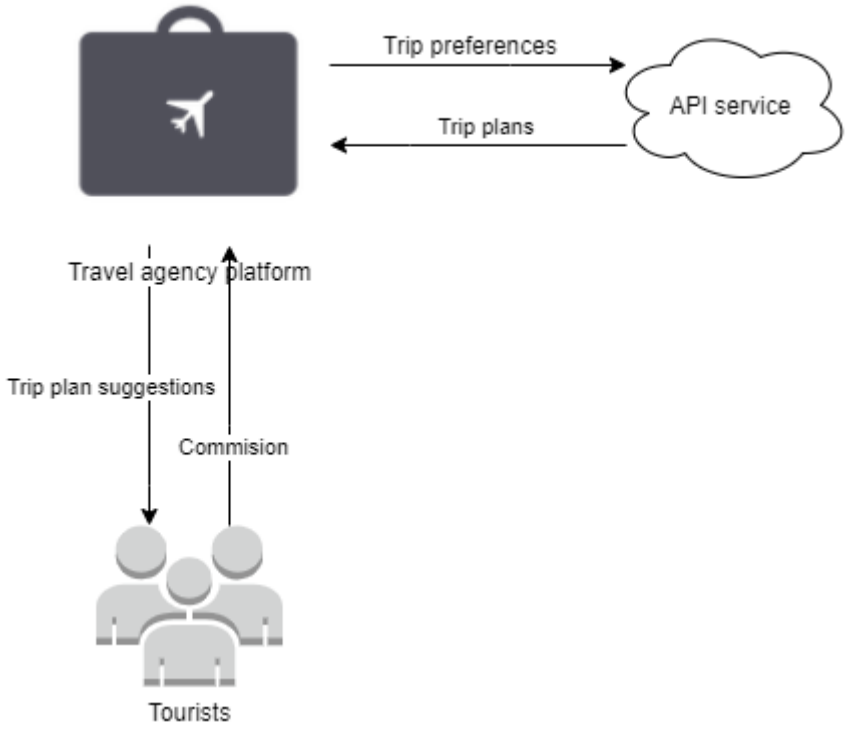

Figure 2. Example of a use case of the product

2.1.2. Quality

Quality criteria of the project is described in Table 1.

Table 1. Quality criteria

No.	Criteria	Justification
1	Trip relevance	Time it takes for the algorithm to generate new trips based on the changed data is less than 24 hours.
2	User satisfaction	At least 70% of the users find generated travel trip plans unique and interesting.

2.1.3. Competition and alternatives

Similar solutions to the product described in this document could not be found online, meaning that the product could be unique and transform the old-fashioned way of planning trips. Market leaders offering such a traditional way of purchasing trips require for the user to provide a lot of information to query for existing flights. Also, they lack the functionality of combining multiple flights into a single cohesive travel experience.

2.2. Requirements

2.2.1. Partner or Collaborative Applications

- Flight data provider API
- Like.travel travel platform (location scoring and user preferences provider)
- Geospatial database

2.2.2. Off-the-Shelf Software

- Free and open source scientific and ML Python packages (such as NumPy, pandas, TensorFlow, PyTorch)
- Kotlin programming language

2.2.3. The Scope of the Product

Scope of the project work done for the master‘s degree project is shaded by a red color below in Figure 3.

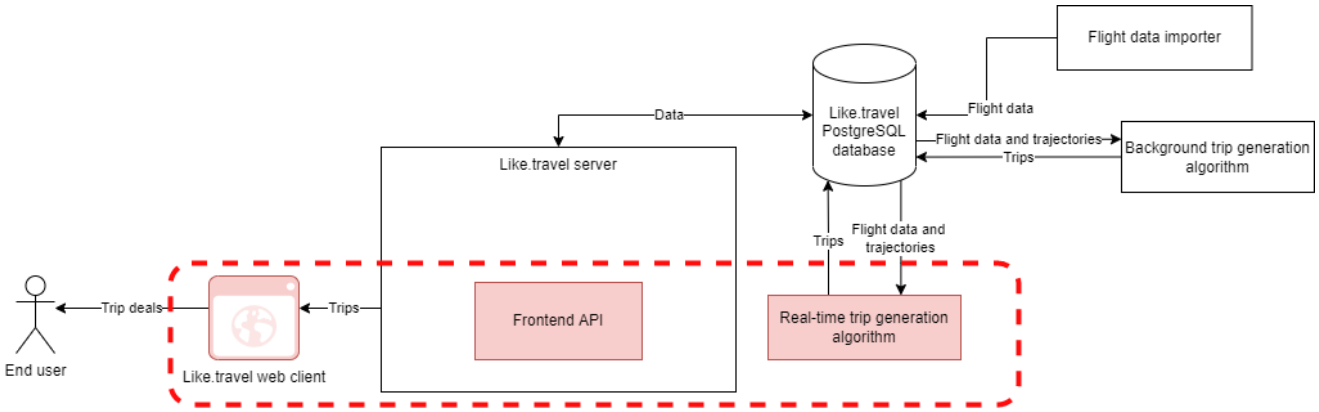

Figure 3. Scope of the project work shaded in red

2.2.3.1. System Boundary

The use cases for the algorithm, the system user and the system admin are presented in Figure 4, Figure 5 and Figure 6. Algorithm use cases include the requirements and constraints that it must take into account when generating the trips.

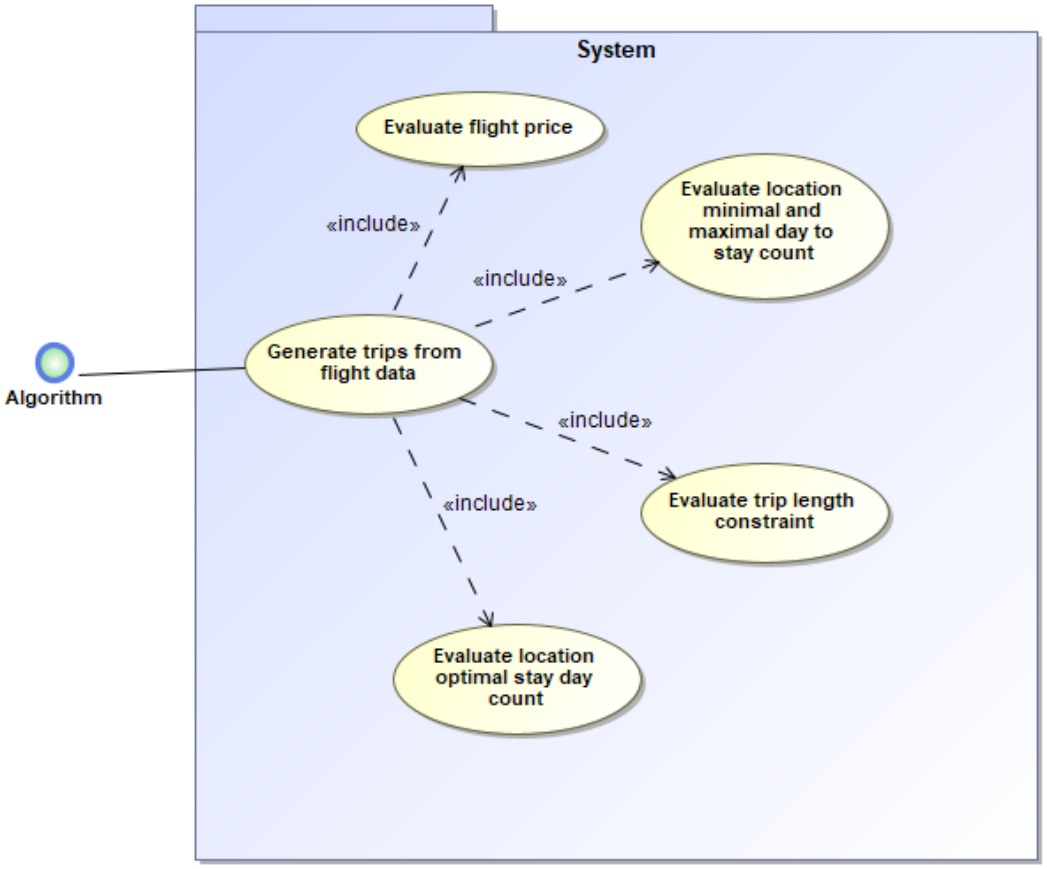

Figure 4. Trip generation algorithm use case diagram

User use cases describe actions that the end user can do while browsing the system.

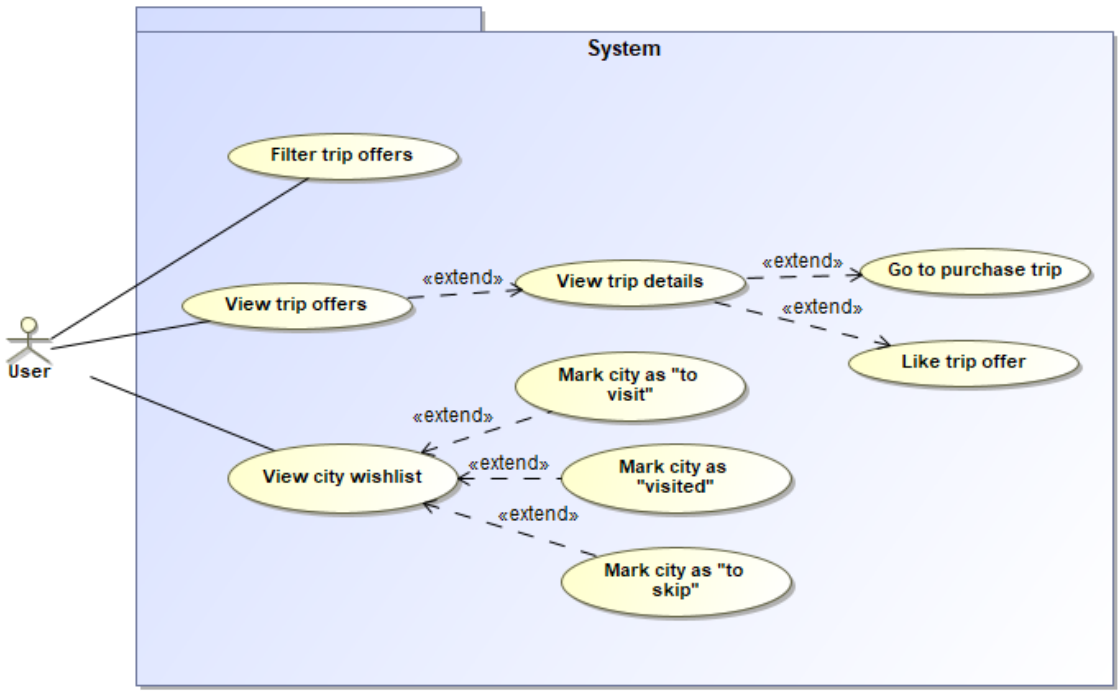

Figure 5. User use case diagram

Admin use cases describe the actions that the system administrator can do.

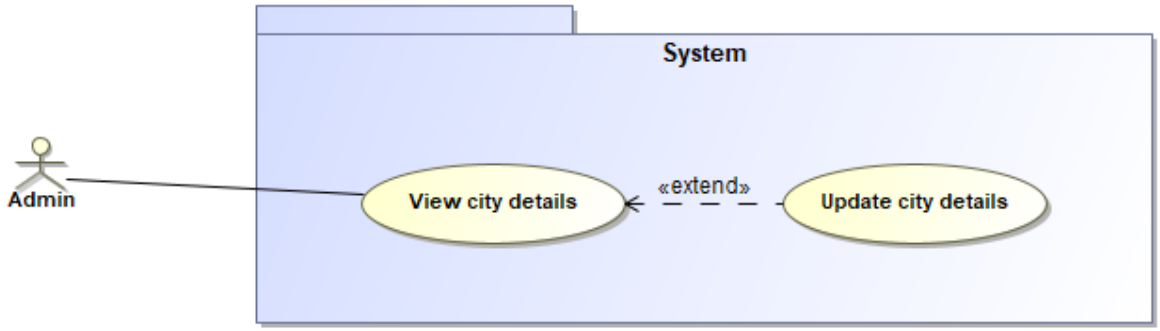

Figure 6. Admin use case diagram

2.2.3.2. Use Case Table

Table 2 describes all the use cases and acceptability criteria for the algorithm, user and the admin user actors.

Table 2. Use case table

No.	Actor	Use case	Description	Acceptability criteria

1	Algorithm	Generate trips	The algorithm should generate qualitative trip deals from given flight data	Algorithm can generate at least a single trip
2	Algorithm	Evaluate flight price	Flights with cheaper prices should be prioritized	Algorithm takes flight price into account
3	Algorithm	Evaluate location minimal and maximal day stay count	The location day stays in generated trips should respect minimal and maximal day constraints	Algorithm respects minimal and maximal day constraints for locations
4	Algorithm	Evaluate trip length constraint	The algorithm should not generate trips longer than the selected trip length	Algorithm respects trip length constraint
5	Algorithm	Evaluate location optimal (preferred) stay day count	The algorithm should prioritize trips with location stays that match their optimal day count	Algorithm respects optimal day constraints for locations
6	User	Filter trip offers	User should be able to filter the trip offers by the starting city, starting month and the date range for how long the trip should last	User is able to filter the trips
7	User	View trip offers	User should be able to view the trip list. The list should show as many trip offers as there is, by implementing infinite scrolling technique	User is able to view trip offers
8	User	View trip details	User can select a particular trip from the trip list and view the trip information in detail.	User is able to view trip details
9	User	Go to purchase trip	User can be redirected to the flight provider page, where he can begin the checkout process for purchasing the flight tickets	User can purchase the trip by being redirected to the flight provider page
10	User	Like trip offer	User can like trip offer, which later could be used to revisit the trip offer	User can like the trip offer
11	User	View city wishlist	User can view the location wishlist showing his trip location preferences	User can view city wishlist
12	User	Mark city as „to visit“	In the wishlist map, user can mark a location which he wants to visit. Trip offers with such locations will be prioritized in the trip list	User can mark location which he wants to visit
13	User	Mark city as „visited“	In the wishlist map, user can mark a location which he has already visited.	User can mark a location which he has already visited
14	User	Mark city as „to skip“	In the wishlist map, user can mark a location which he does not wish to visit. Trip offers with such	User can mark location which he wants to skip

			locations will be less prioritized in the trip list	
15	Admin	View city details	Admin can view the details for every city, such as image URL, assigned airports, min/max/preferred days to stay, amount of trips and others	Admin can view the details for every city
16	Admin	Update city details	Admin can update the city details for a particular city	Admin can update the details for particular city

2.2.3.3. Business Data Model

Entity relation diagram is presented in Figure 7.

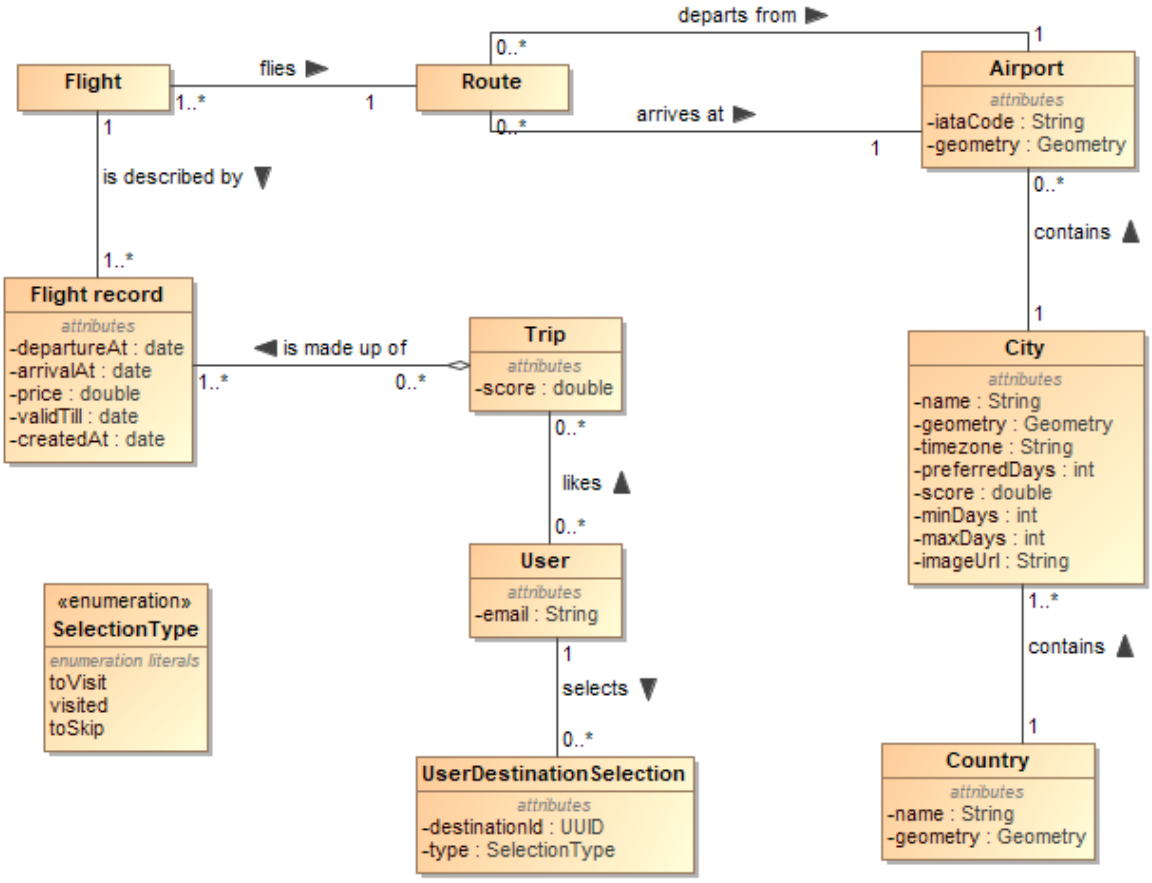

Figure 7. Business entity relation diagram

2.2.4. Non-functional Requirements

2.2.4.1. Look and Feel Requirements

At least 70% of the users find generated travel trip plans unique and interesting.

2.2.4.2. Performance Requirements

The generated trips may lag behind the real-time flight data changes by no more than 24 hours.

2.2.5. Risks

The availability of flight data API is not guaranteed. The APIs may also restrict access to data, which could lead to less information for the algorithm to create quality trips.

2.3. Architecture

2.3.1. Architectural Goals and Constraints

Below are presented the goals and constraints of the architecture:

- Architecture should be clear and concise for new team members to adapt to.
- System should be easily scalable on demand.
- System should be distributed on servers supporting JRE 11.
- System backend services should not be accessible from public.

2.3.2. Overview

System packages related to trips are presented in Figure 8. The project scope does not include *flight*, *cli*, *trip.planner* packages. Not all files from other packages are included in the project scope.

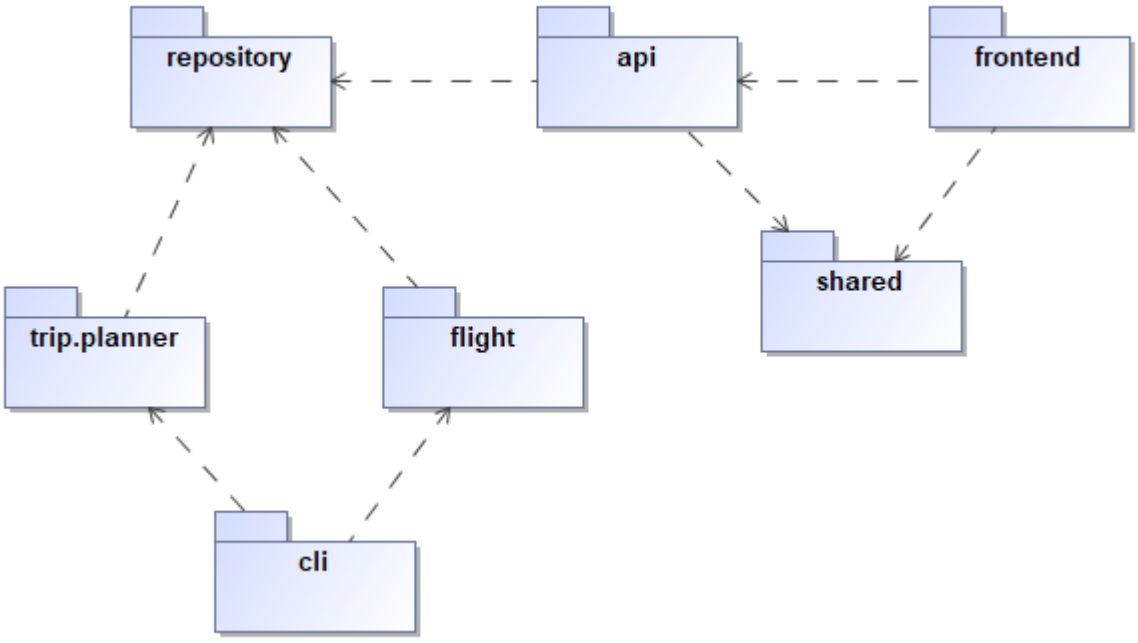

Figure 8. Package diagram

2.3.3. Architecturally Significant Design Packages

– Package “api”

Package “api” is used to control communication between the client and the server. The server provides trip models for the client.

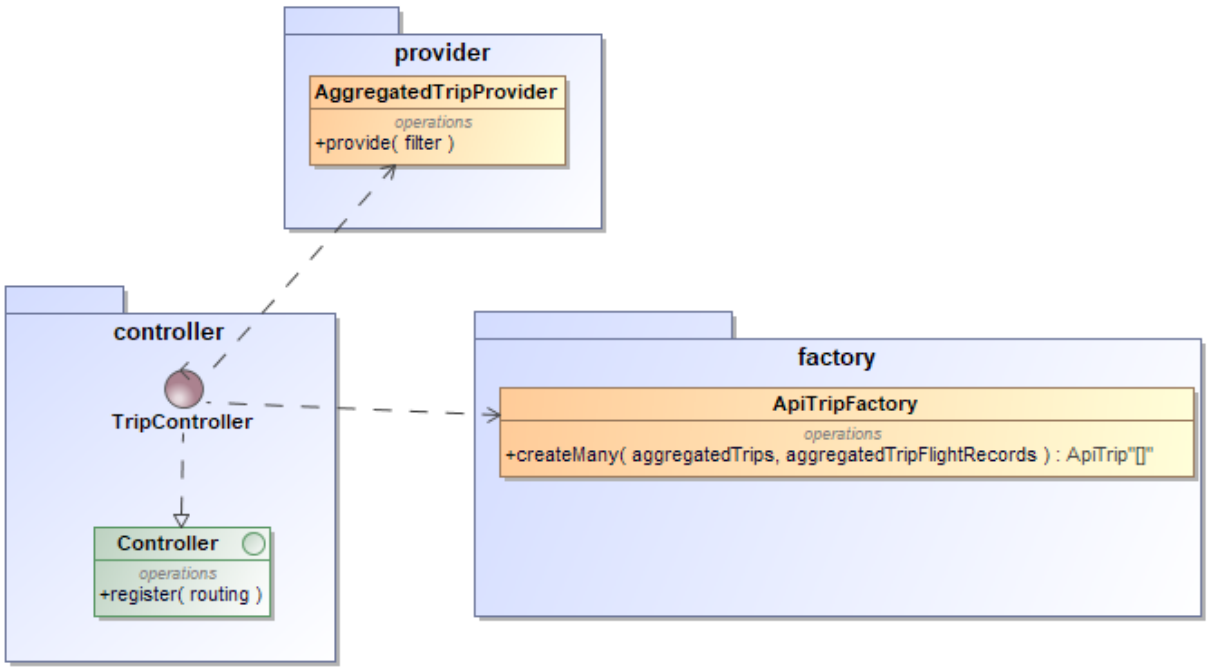

Figure 9. Package "api"

– Package “shared”

Package “shared” is used to hold common data models which are used in communication between the client and the server.

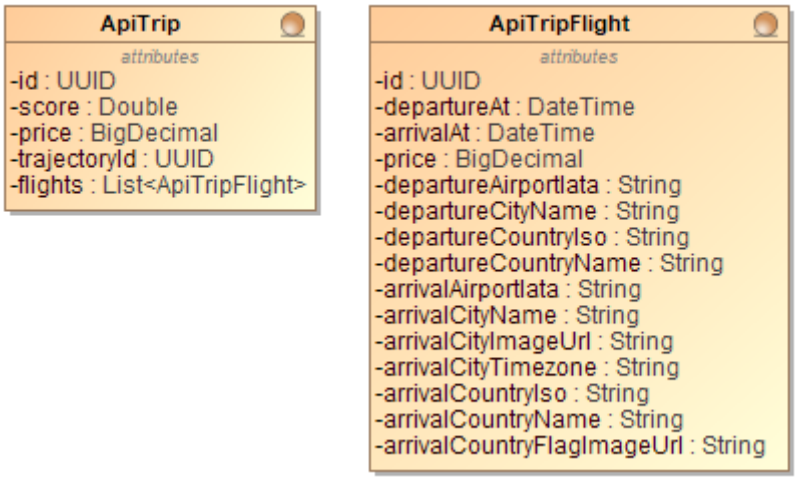

Figure 10. Package "shared"

– Package “frontend”

Frontend package is used to present the trips to the end user.

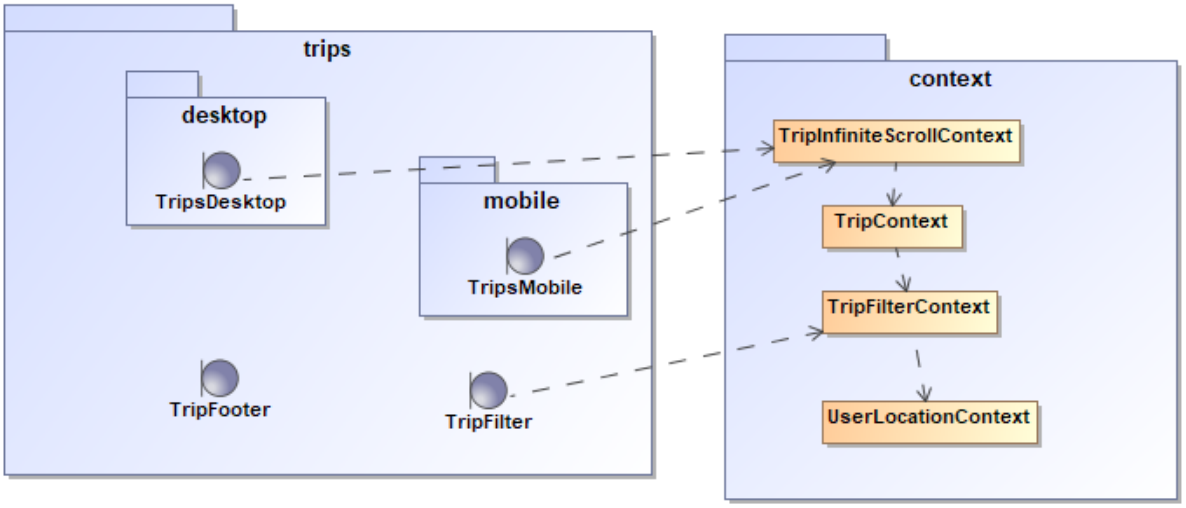

Figure 11. Package "frontend"

2.3.4. Deployment view

Deployment diagram is presented in Figure 12.

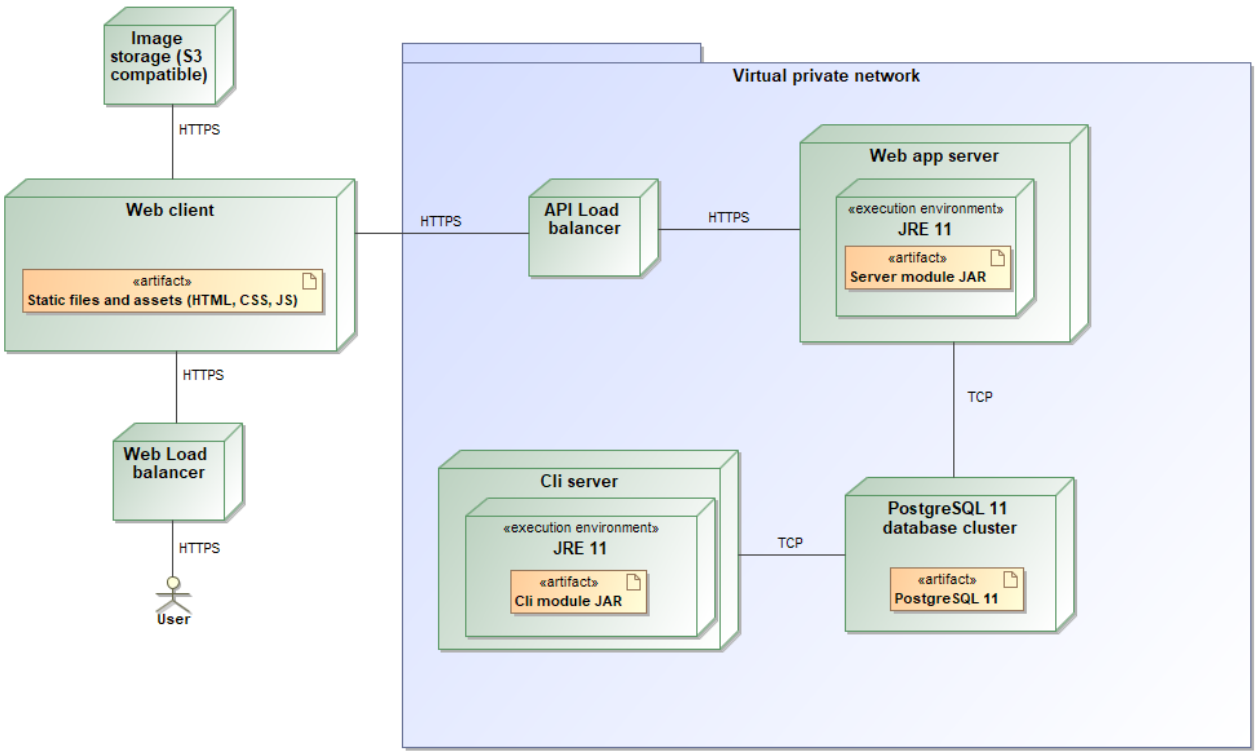

Figure 12. Deployment diagram

2.3.5. Data View

The database model for the system is presented in Figure 13.

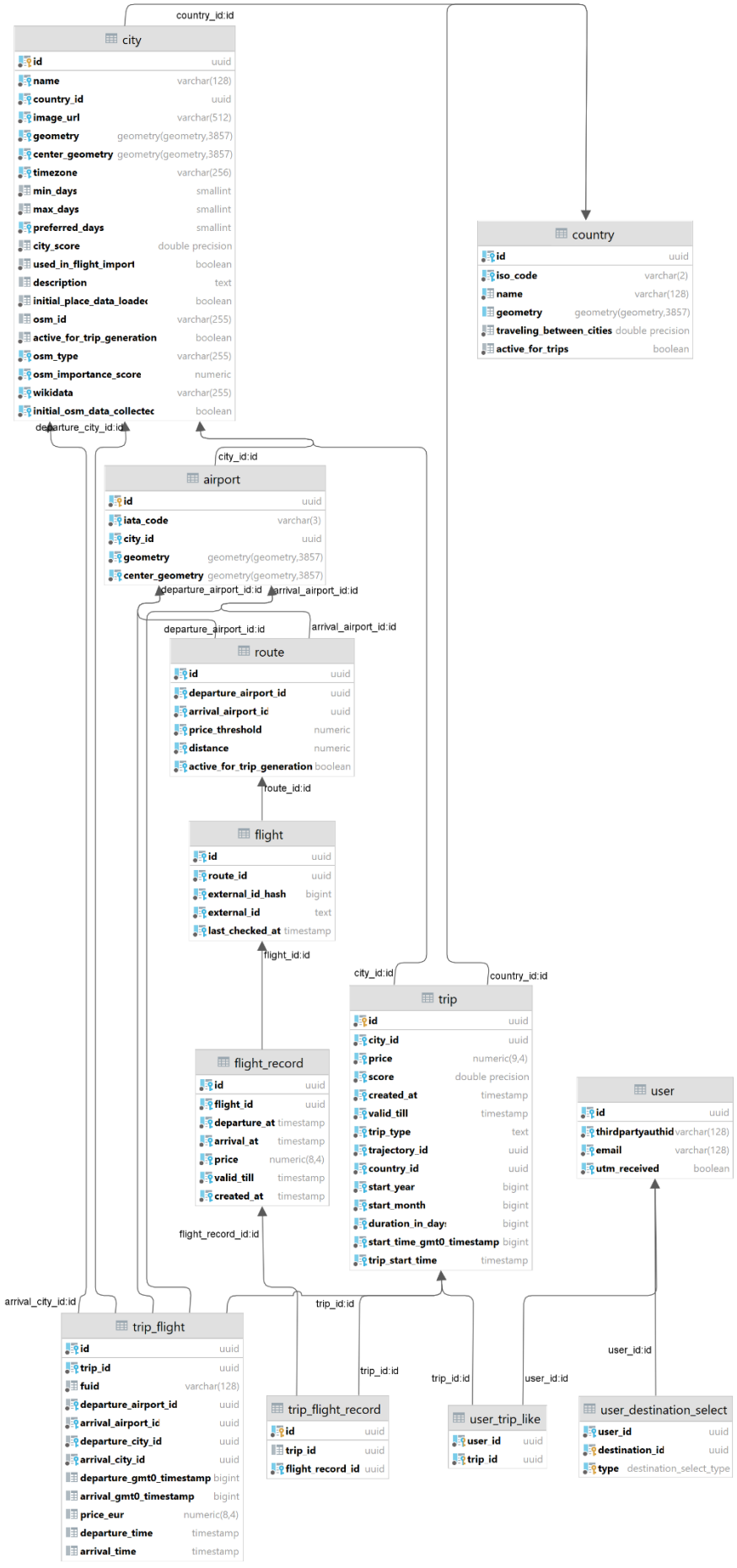
country_id:id
city
id uuid
name varchar(128)
country_id uuid
image_url varchar(512)
geometry geometry(geometry,3857)
center_geometry geometry(geometry,3857)
timezone varchar(256)
min_days smallint
max_days smallint
preferred_days smallint
city_score double precision
used_in_flight_import boolean
description text
initial_place_data_loade boolean
osm_id varchar(255)
active_for_trip_generation boolean
osm_type varchar(255)
osm_importance_score numeric
wikidata varchar(255)
initial_osm_data_collectec boolean
country
id uuid
iso_code varchar(2)
name varchar(128)
geometry geometry(geometry,3857)
traveling_between_cities double precision
active_for_trips boolean
departure_city_id:id
city_id:id
airport
id uuid
iata_code varchar(3)
city_id uuid
geometry geometry(geometry,3857)
center_geometry geometry(geometry,3857)
departure_airport_id:id
arrival_airport_id:id
departure_airport_id:id
arrival_airport_id:id
route
id uuid
departure_airport_id uuid
arrival_airport_id uuid
price_threshold numeric
distance numeric
active_for_trip_generation boolean
route_id:id
flight
id uuid
route_id uuid
external_id_hash bigint
external_id text
last_checked_at timestamp
flight_id:id
city_id:id
country_id:id
trip
id uuid
city_id uuid
price numeric(9,4)
score double precision
created_at timestamp
valid_till timestamp
trip_type text
trajectory_id uuid
country_id uuid
start_year bigint
start_month bigint
duration_in_days bigint
start_time_gmt0_timestamp bigint
trip_start_time timestamp
flight_record
id uuid
flight_id uuid
departure_at timestamp
arrival_at timestamp
price numeric(8,4)
valid_till timestamp
created_at timestamp
user
id uuid
thirdpartyauthid varchar(128)
email varchar(128)
utm_received boolean
flight_record_id:id
arrival_city_id:id
trip_id:id
trip_id:id
trip_id:id
user_id:id
user_id:id
trip_flight
id uuid
trip_id uuid
fuid varchar(128)
departure_airport_id uuid
arrival_airport_id uuid
departure_city_id uuid
arrival_city_id uuid
departure_gmt0_timestamp bigint
arrival_gmt0_timestamp bigint
price_eur numeric(8,4)
departure_time timestamp
arrival_time timestamp
trip_flight_record
id uuid
trip_id uuid
flight_record_id uuid
user_trip_like
user_id uuid
trip_id uuid
user_destination_select
user_id uuid
destination_id uuid
type destination_select_type

Figure 13. Database view

2.3.6. User interface

The user interface screenshots for desktop and mobile screens are presented in the figures below. The trip list views are presented in Figure 14 and Figure 15. The trip filter is visible in the desktop view, while it can be opened by clicking on a search icon in the header in the mobile view.

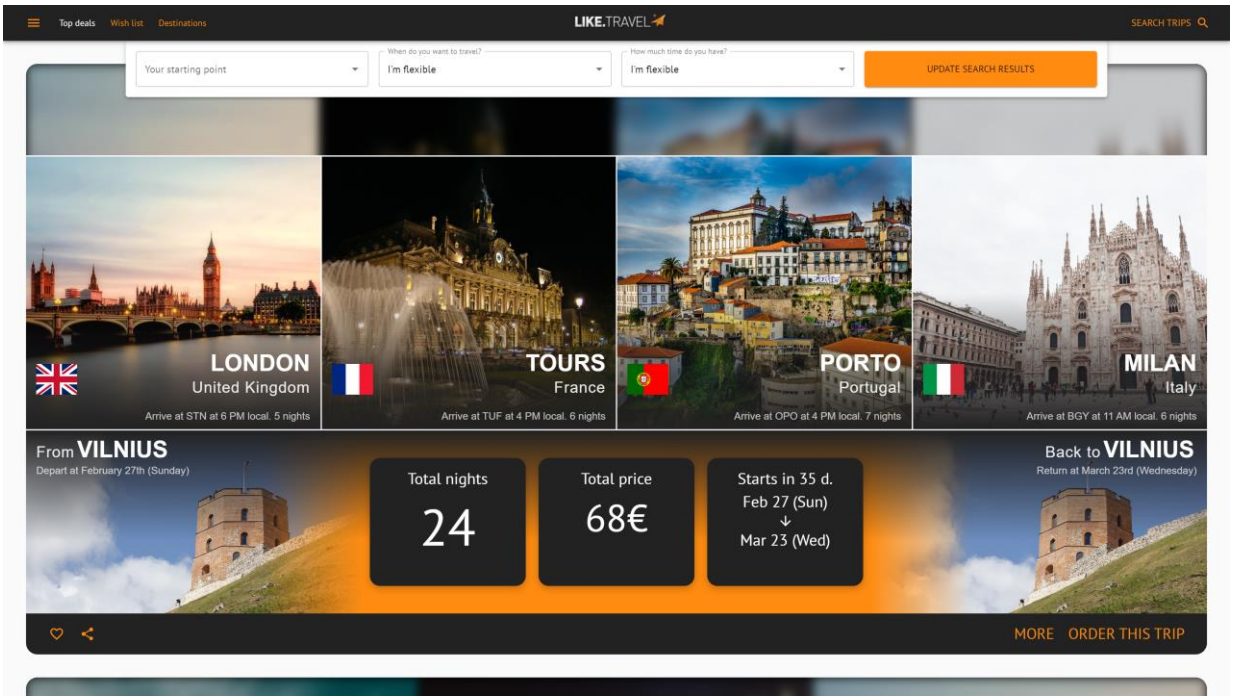

Figure 14. Trip list view for desktop screens

Figure 15. Trip list view for mobile screens

Trip detailed view screens are presented in Figure 16 and Figure 17. Trip purchase button and trip like button are present in the footer.

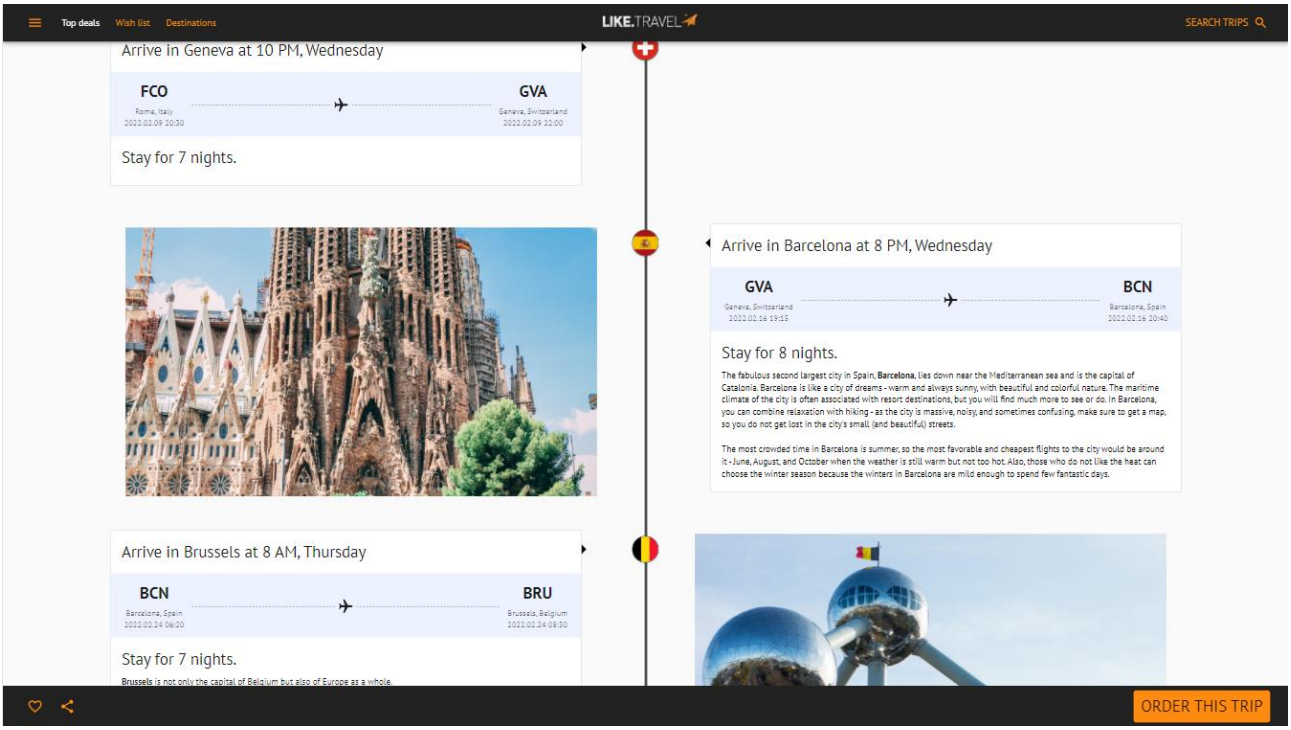

Figure 16. Detailed trip view for desktop screens

Figure 17. Detailed trip view for mobile screens

The wishlist map view screens are presented in Figure 18 and Figure 19. Select forms to select preferred destinations are available for access in both screens.

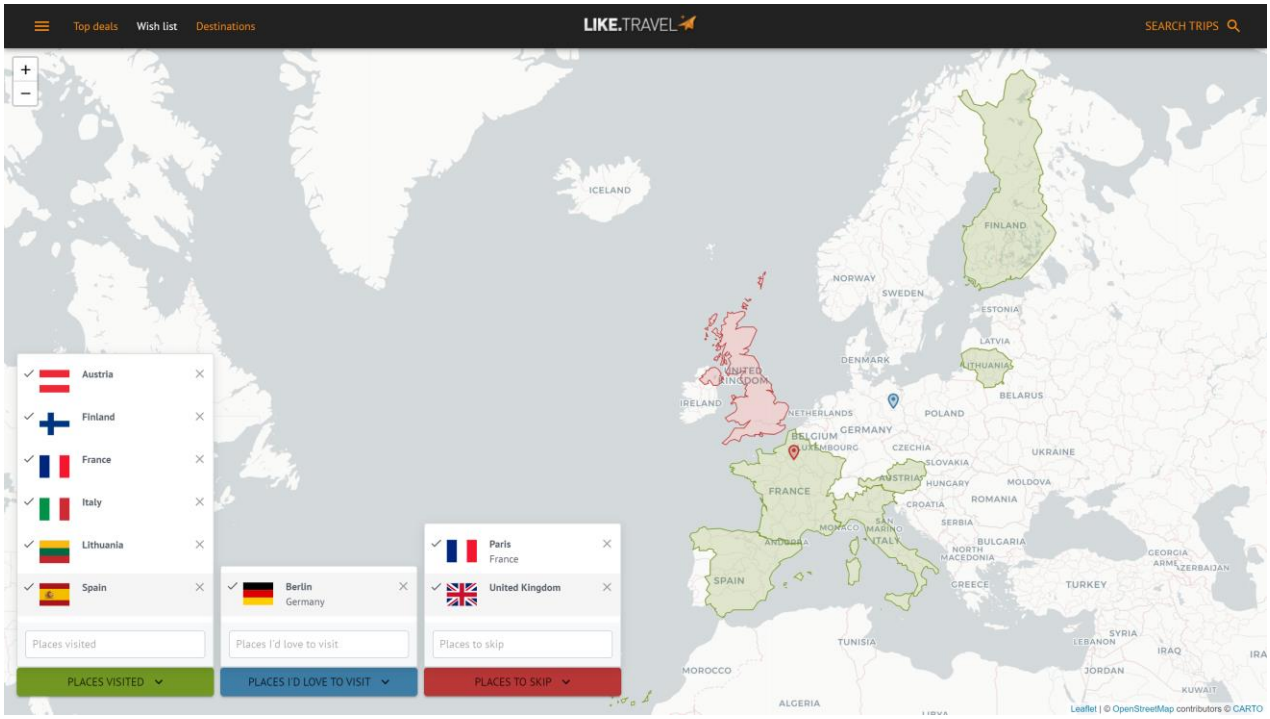

Figure 18. Wishlist map view for desktop screens

Figure 19. Wishlist map view for mobile screens

The administrator views for city detail managing are presented in Figure 20 and Figure 21.

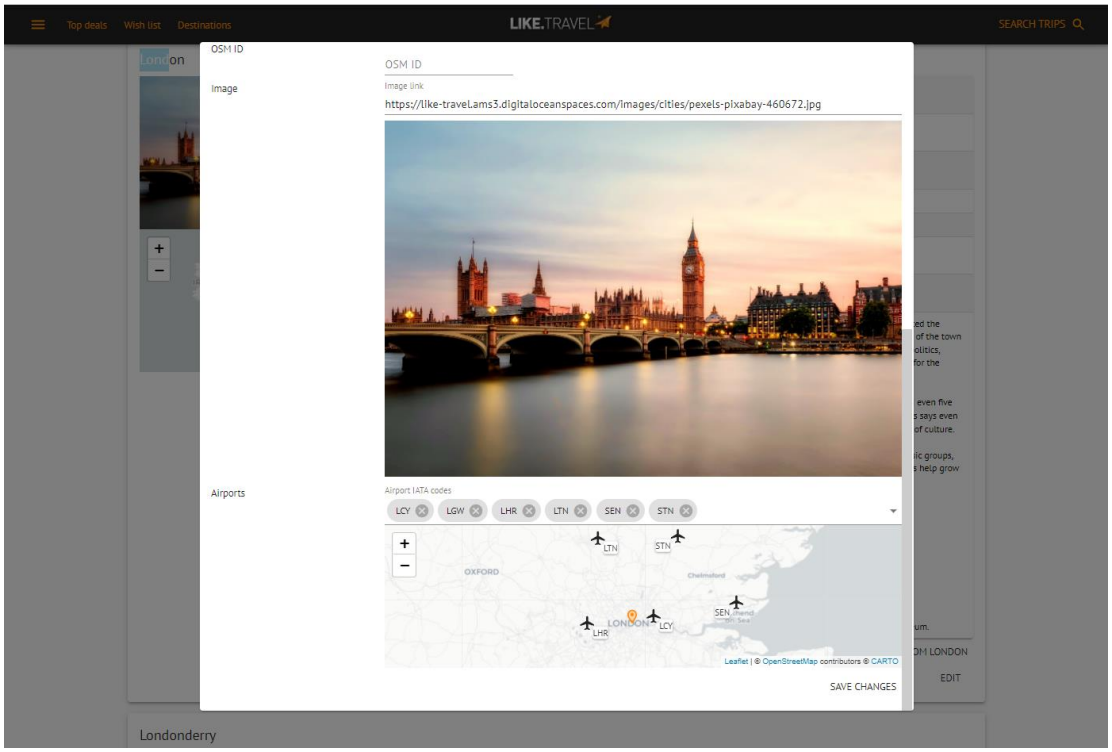

Figure 20. City edit desktop view screen

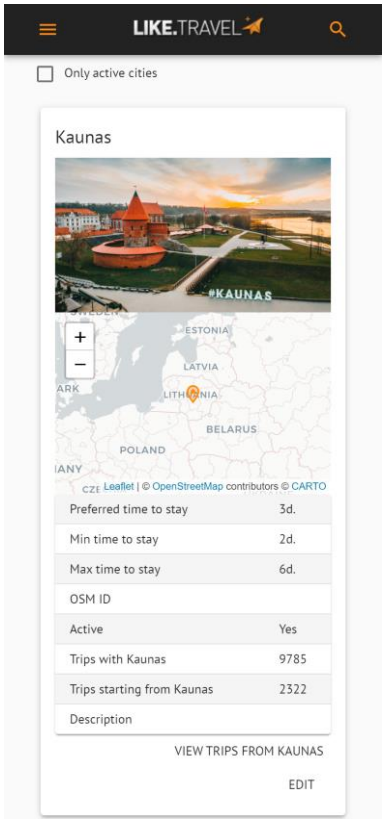

Figure 21. City view mobile view screen

3. Research and experiments

3.1. Problem description

Each trip contains a set of cities *T*, $|T| = N$, with a particular order. A trip starts at the start city t_1 and the last city visited is denoted as t_n. Every trip is a round trip and ends at the start city t_1. The ordered set *T* is referred to as a *trajectory.*

For the experiments, we consider each of the trajectories to consist of 5 cities. Trajectories are generated based on the real-world flight routes which seldom change. Since there could be a total of $N \cdot (N-1) \cdot (N-2) \cdot (N-3) \cdot (N-4)$ trajectories, where *N* is the total number of cities, it would be impractical to consider all the possible combinations. Trajectory amount can be reduced by selecting only the most attractive trajectories – ones with the best trajectory round score and combined OpenStreetMap Place Importance Score (OSM PIS)[1]. Round score is calculated by dividing the total trajectory distance by the minimal possible distance connecting all the cities. Since only 5 cities make the trajectory, calculating the round score is trivial. The generation of trajectories happens in a background process and is not a part of the scope of this research. An example of a trajectory with its adjacent cities connected by a blue line is presented in Figure 22.

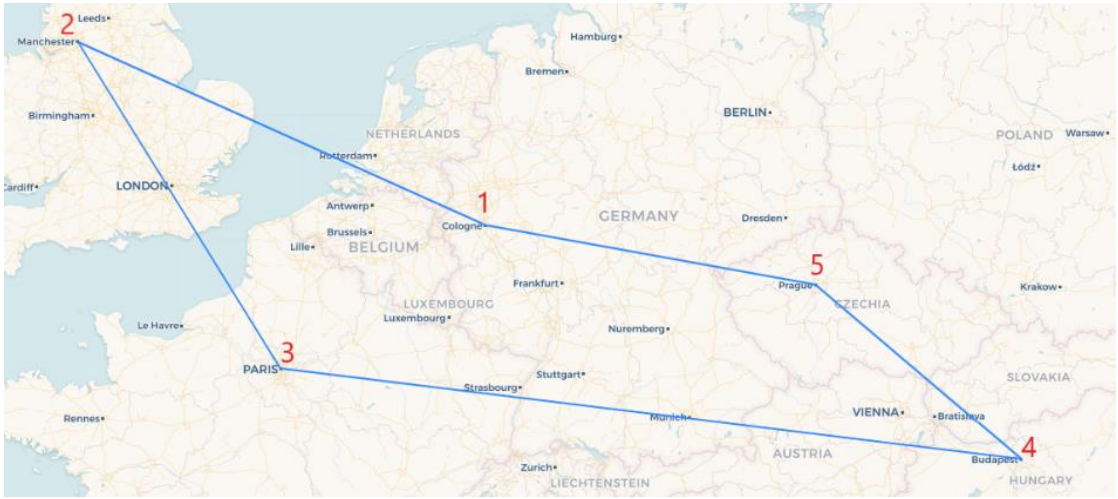

Figure 22. Trajectory graph example

Given the flight prices for a certain period for a certain trajectory, we want to find the best possible trip. The score of the trip equals to the sum of its flight ticket prices. A penalty is added to the trip score if the time spent in a city does not match its preferred number. The lower the score, the better the trip is considered. An example of a best trip in a trajectory is presented in Figure 23. The Y axis represents 6 routes to make a round trip between 6 cities. The X axis represents dates. The blue dots represent an existing flight for a route on a certain day. The number next to the blue dot represents the flight price. The best possible path of the trip is the example marked in red. In the example, it is assumed that the preferred amount of days to stay in each city is 5, and a penalty of 7 is added to the trip score per absolute day offset.

[1] https://lists.openstreetmap.org/pipermail/geocoding/2013-August/000916.html

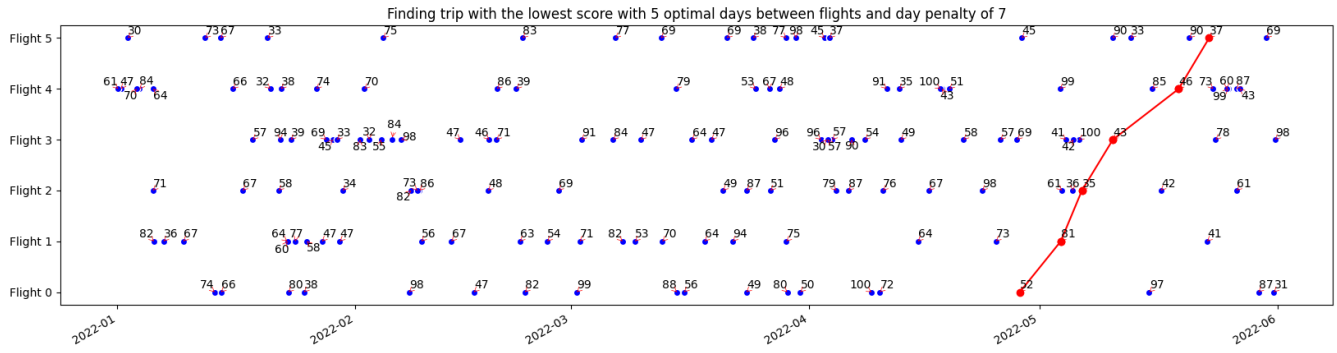

Figure 23. Finding the best trip in a trajectory

A tree search algorithm based on dynamic programming is used to find the actual best possible flight combination for a given trajectory. It works like a brute force tree search algorithm, but with optimizations. Instead of checking each possible flight combination, it stops traversing the flights if the flight for a given day was already traversed and had a better total flight price.
In the research presented below, a few heuristic approaches using machine learning for finding the best trip in a trajectory are tested. Although DP guarantees to find the best possible trip, an approach using ML can be much faster.

3.2. Environment

The computing environment used for the research has the following parameters:

- RAM memory: 32 GB
- CPU: AMD Ryzen 7 3700X 8-Core Processor 3.60 GHz (1 CPU core used)
- OS: Windows 10 64-bit architecture

Python 3.9 programming language is used to program the software. PyTorch machine learning framework is used to create the neural network models.

3.3. Trip classification using CNN

This method uses convolutional neural networks to classify if a good trip can be found in a trajectory. For all trajectory chunks that are classified as containing a qualitative trip, a DP algorithm then can be run to find the exact flights for the best possible trip in that trajectory part.

3.3.1. Method

A heatmap image is generated for each trajectory. An example of a heatmap is presented in Figure 24. Each cell represents an existing flight for a route on a certain day. The shade of the cell depends on the price of the flight. If a flight does not exist for a certain day, the cell is marked as black.

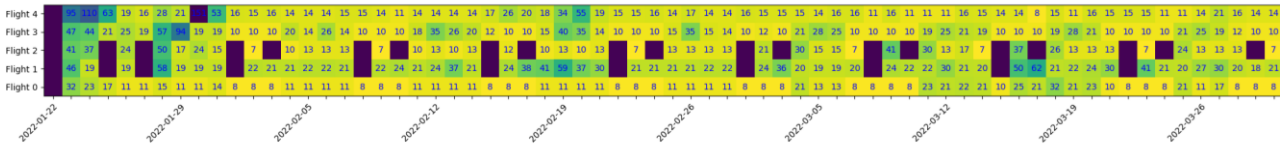

Figure 24. Trajectory heatmap

The flight prices are normalized for each individual trajectory using Euclidean normalization before generating the heatmap.
For a more efficient way to pass the heatmap input to the CNN model, the heatmap is converted to grayscale with each cell having a value from 0 to 255. A heatmap example in Figure 24 converted to grayscale is presented in Figure 25.

Figure 25. Trajectory heatmap converted to grayscale

A maximum possible trip length for this experiment was set to 25 days. The heatmap is then chunked into images of 25 pixels wide, with the chunk step being 5 days. For each heatmap chunk, the DP algorithm labels the target class for the CNN model training. There are two target classes – an input is a good trip candidate, or an input is not a good trip candidate. A trip score threshold value is set manually to a value of 80 to determine the class. Chunked images are presented in Figure 26.

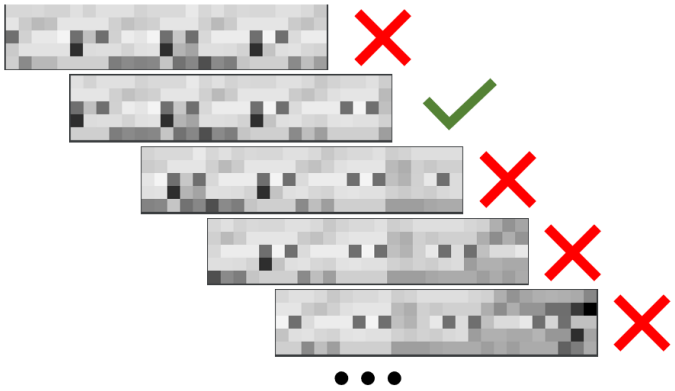

Figure 26. Trajectory heatmap chunks

For all of the heatmap chunks that CNN classifies as possible trip candidates, DP algorithm is then run to find the exact flights.

Figure 27. Exact trip flights in a positively classified heatmap

3.3.2. Experiment and results

Neural network training hyperparameters are as follows:

- Batch size: 32
- Optimizer: Adagrad [18]
- Loss function: Cross entropy
- Learning rate: 0.015

The model architecture is presented in Figure 28.

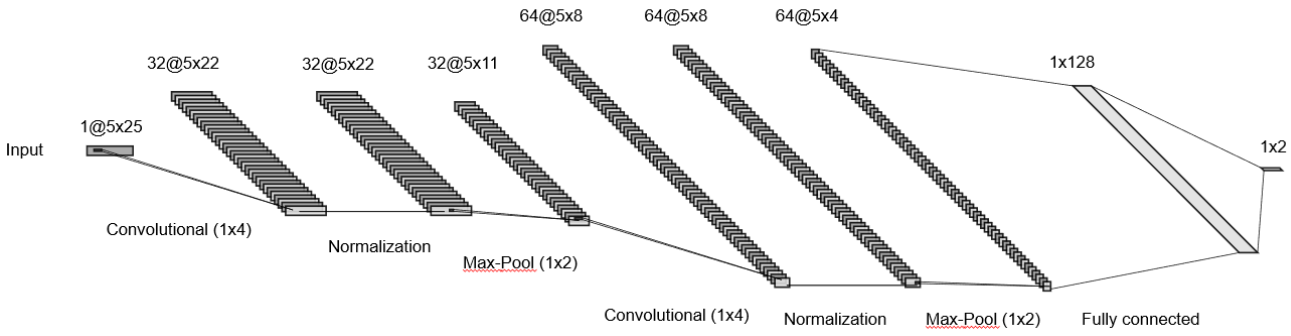

Figure 28. CNN model architecture

18120 total chunk images were used for the model training and 3712 images for accuracy testing. 50% of all images for the experiment were classified as containing a good trip, while the other ones as not containing.
After 5 minutes of training, the model has achieved approximately 94% accuracy and could process 2000 images per second.

3.4. Real-time trip generation using FNN

The trip classification method was limited, since the machine learning model did not take into account most of the functional requirements – minimal, maximal day amounts to stay in particular cities as well as trip length constraints. Also, the speed of the model was not fast enough for real-time trip generation. A new method that fixes both of the before mentioned problems was tested. Also, instead of classification, it performs regression, meaning that the quality of the trip can be evaluated.

This method uses a heuristic solution that allows to efficiently find the best trip offers using a feedforward neural network combined with the dynamic programming algorithm. The feedforward neural network model can narrow down the number of trajectories to a smaller amount of potential best trip candidates, while the DP algorithm is then able to select the *N* best trips. 3 scenarios with different constraints on the trip offer are tested.

A publication [19] based on this method was created and presented in a conference.

3.4.1. Method

The trajectories used for the experiment for the real-time trip generation method are made up from 100 selected European cities. 210000 trajectories were generated for the experiment. City selections are based on OSM PIS. Each city has a set of airports assigned to it, which is used to associate flight data with the city. The cities used are marked in Figure 29.

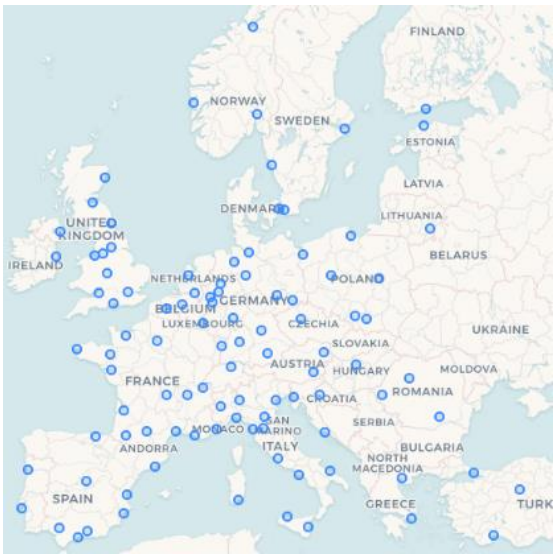

Figure 29. Selected cities for real-time trip generation experiment

The method to find the best *N* trips given *K* possible trajectories and flight data is as follows:

1. Pass the flight prices of *K* trajectories of the given date range to the FNN model. Each day can have at most a single flight for a given route between two cities.
2. Pass some amount *M* of best predicted trajectories and their flight prices to the DP algorithm.
3. Use the best *N* trajectories returned by the DP algorithm and the flight data to build the best *N* trip offers.

The sequence diagram for the method is presented in Figure 30.

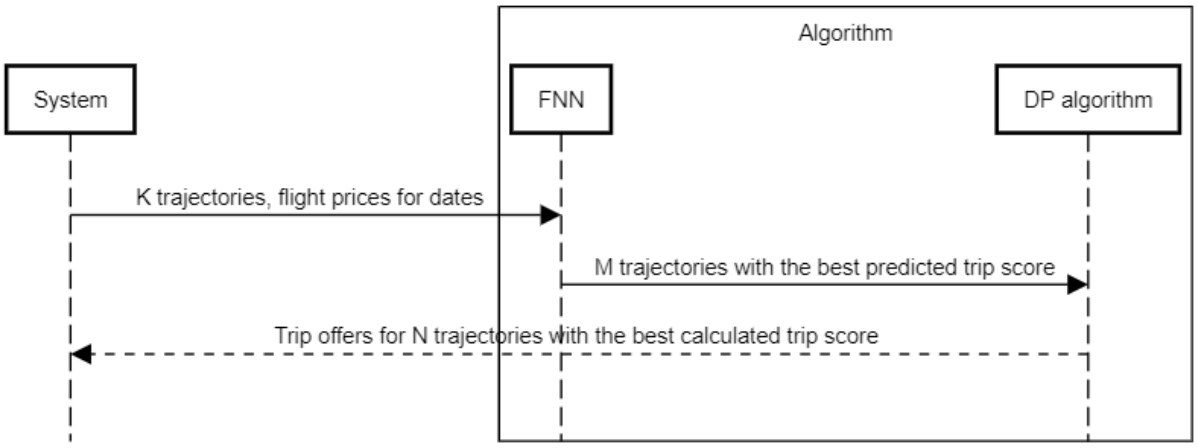

Figure 30. Real-time trip generation method using FNN

To determine if this method is viable in practice, we evaluate speed and accuracy metrics. Speed is measured as the combined computing time of FNN prediction and DP algorithm. Accuracy is determined by comparing the final output of *N* best trajectories to expected *N* best trajectories and diving the sum of matching pairs by *N*. Accuracy is influenced by the number of total trajectories passed to the FNN model and the number of best model predictions passed to the DP algorithm.

3 trip generation scenarios are explored. They differ by the constraints applied to what can be considered a viable trip.

- Scenario 1: No additional constraints.
- Scenario 2: Each city has a minimum, maximum and preferred number of days to spend in that city. Trip generation must respect the minimum and maximum constraints and apply a penalty if the time spent in a city does not match its preferred number. The penalty adds a value of 5 to the trip score per absolute day offset.
- Scenario 3: Same as scenario 2, also, a total trip length constraint is added. The trip length must be in one of the three intervals:
    - 10 – 13 days
    - 14 – 20 days
    - 21 – 24 days

The scenarios are visualized in Figure 31, where the X axis represents the flights, and the Y axis represents the days. The red cells mark the days in which it is impossible to take the flight to match the given constraints. The green cells represent the possible days to take flights if trip is starting from the earliest day (03-28). Yellow cells represent other possible days if trip were to start from another day. With each scenario, the amount of possible flight combinations is reduced. In the Figure 31 example, the min. and max. number of days to stay in every city are 3 and 7 respectively and the total trip length should span from 14 to 20 days. In the experiment, for all scenarios we constrain the maximum trip length to 24 days.

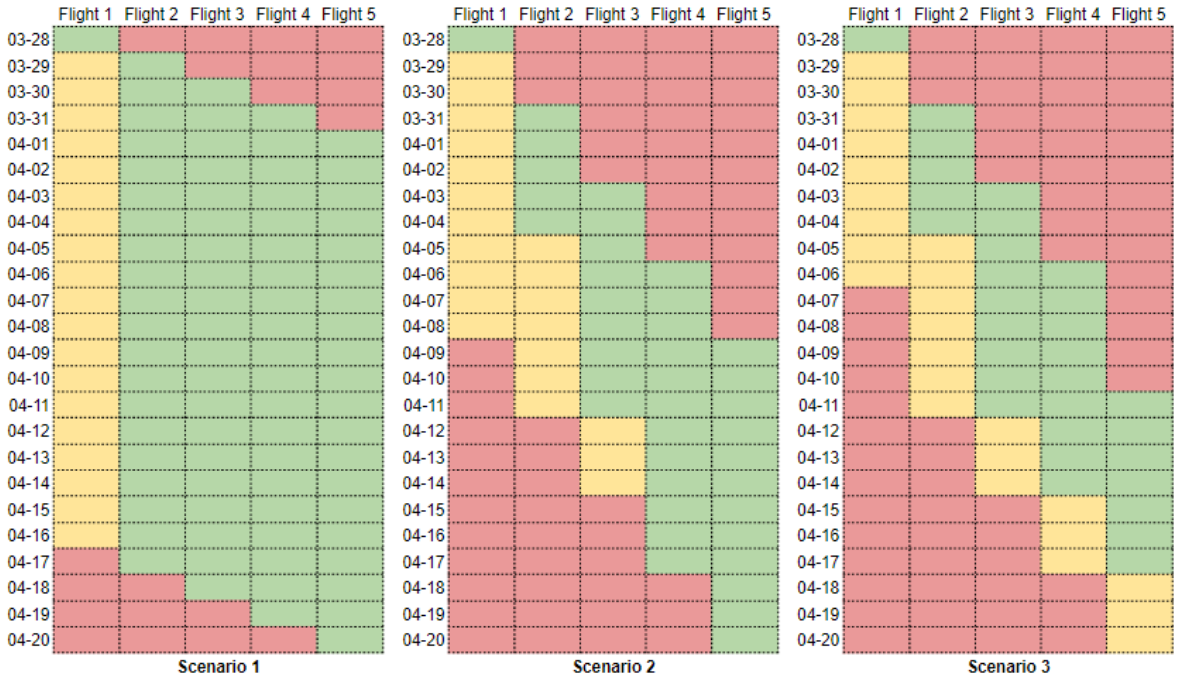

Figure 31. Testing scenarios

3.4.2. Experiment

Common neural network training hyperparameters for each scenario are as follows:

- Batch size: 128
- Optimizer: Adam [20]
- Loss function: Mean absolute error (MAE)
- Learning rate: Reducing learning rate on plateau by a factor of 0.5 on 3 consecutive epochs without improvement.
- Epochs: Until does not improve for 10 epochs or until 60.

The departure dates for all the flight data in the experiment span between 2021-03-28 and 2021-05-15. Flight data from 2021-03-28 to 2021-04-20 is used for training, while data from 2021-04-21 to 2021-05-15 is used for validation. For scenario 1, flight prices of 24 days for each of the 5 trajectory cities are passed to the model as an input, for a total input length of 120. If the flight data for a particular day is missing, it is passed to the model as a value of -1. For scenario 2, the number of min., max. and preferred days for each city is added to the input, which increases the input length to 135. For scenario 3, numbers for min. and max. trip length are added, for an input length of 137. The target trip scores for the model training were built using the DP algorithm. If not a single trip can be built for a trajectory under certain constraints, the target is set to a value of 1000. The model outputs a single value – a trip score. 210000 inputs were used to train model for scenarios 1 and 2, while scenario 3 trained with three times number of inputs (630000) due to three distinct intervals used for total trip length.

Neural network validation accuracy during training for each scenario is presented in figures Figure 32, Figure 33 and Figure 34. The used notation to describe the model architectures in the figure legends is as follows: $I\ x\ H * N\ x\ O$, where I is the number of inputs for the input layer, H is the number of inputs for each hidden layer, N is the number of hidden layers and O is the number of outputs (1 output describing the trip score). The best model architecture is highlighted with a yellow marker. In general, to obtain the optimal validation accuracy, the models had to become more complex as the trip constraints increased. Model for scenario 2 tends to overfit the most and the model state after 4th epoch is used for its metric check. Techniques such as dropout [21] and dataset scaling were tested but failed to improve the model accuracy.

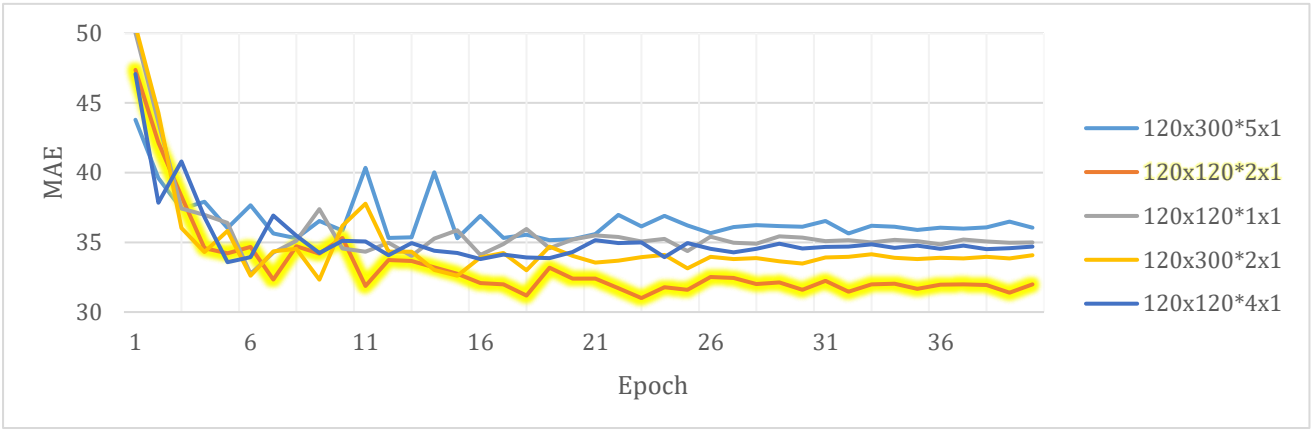

Figure 32. Scenario 1 validation accuracy

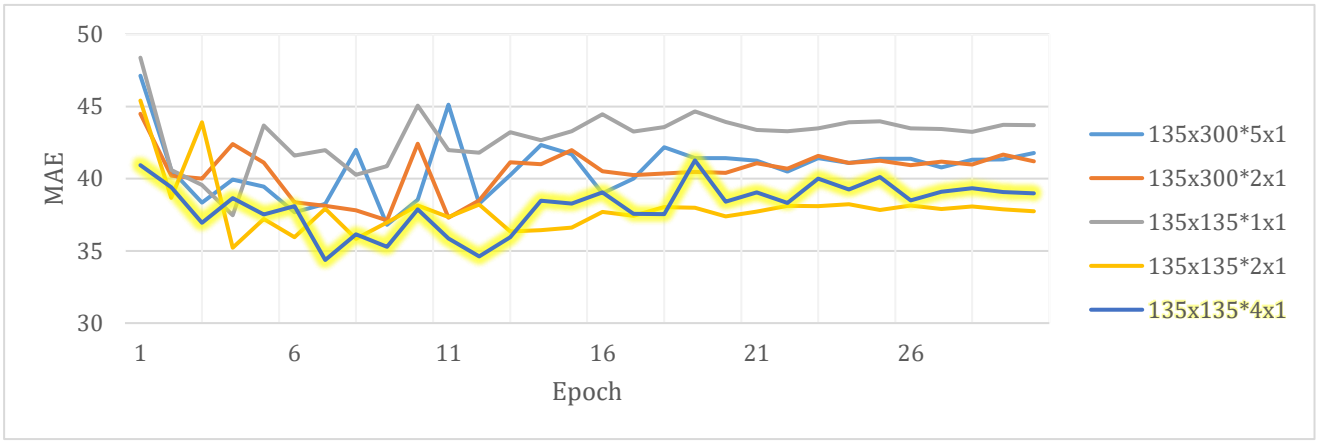

Figure 33. Scenario 2 model validation accuracy

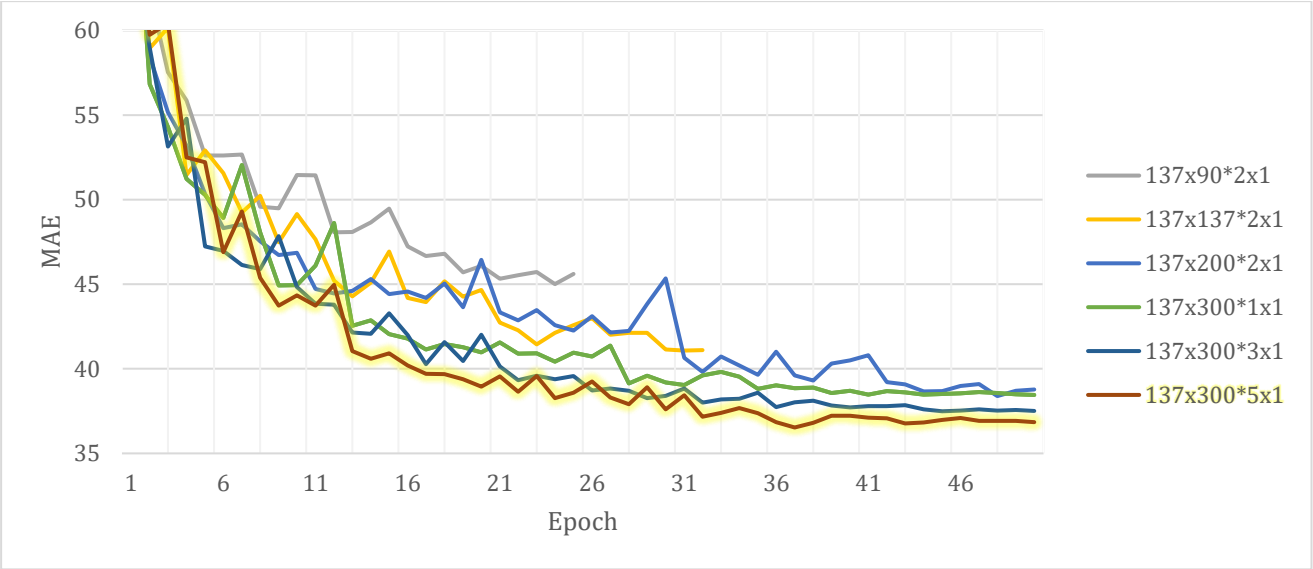

Figure 34. Scenario 3 model validation accuracy

The DP algorithm for this method only iterates through the dates which match the constraints of the min. and max. days to stay in a certain flight city and ignores days which do not match the total trip length constraint (such days are marked red in Figure 31).

3.4.3. Results

Accuracy and speed results are presented in tables Table 3 and Table 4. Accuracy match results were averaged over 50 test runs. In the accuracy result table, cells marked in red, yellow, and green represent respectively the worst, the second best and the best scenario for the testing parameters of the rightmost 3 columns. Columns "*Total predictions made*", "*Top N predictions to search in*" and "*Required top N matches*" denote respectively how many inputs were passed to the FNN model, how many of the best results then were passed to the DP algorithm, and how many final trip offers do we want to output. The cell values in bold mark the values for which at least 80% of the required top *N* matches do match, which is considered a good result.

It is possible to infer from the accuracy results that the more constrained the trip generation scenario is, the more accurate the final matches tend to be. Since the mean absolute error of the FNN model validation accuracy was lower for the more constrained models, this might not seem reasonable. However, it may be explained by the greater value of the standard deviation of more constrained scenario model target array (trip scores) compared to less constrained scenario targets. The final matches tend to be less accurate the more trajectory inputs are passed to the FNN model and the fewer top predictions are ran through the DP algorithm.

The speed results show that the performance of the FNN is extremely quick running faster than half of a second for 50000 trajectory inputs in 1st and 2nd scenario and in 1.7 seconds in 3rd scenario, which uses a more complex neural network architecture. The DP algorithm time decreases as the amount of trip constraints increases.

Table 3. Real-time trip generation accuracy results

Actual matches			Required top N matches	Top N predictions to search in	Total predictions made
1 Scen.	2 Scen.	3 Scen.			
5,48	6,72	**8,72**	10	50	2000
10,86	12,20	**16,22**	20	50	2000
23,04	23,84	31,78	50	50	2000
6,76	**8,26**	**9,50**	10	100	2000
13,92	15,38	**18,42**	20	100	2000
32,42	34,18	**41,74**	50	100	2000
8,96	**9,82**	**9,82**	10	250	2000
18,16	**19,26**	**19,46**	20	250	2000
43,62	**44,76**	**47,58**	50	250	2000
9,84	**9,98**	**9,84**	10	500	2000
19,70	**19,94**	**19,56**	20	500	2000
48,58	**48,90**	**48,82**	50	500	2000
3,60	3,92	6,08	10	50	10000
6,26	6,50	11,00	20	50	10000
11,32	12,32	21,28	50	50	10000
5,16	6,32	**8,12**	10	100	10000
9,22	11,06	15,34	20	100	10000
19,56	22,62	32,82	50	100	10000
7,00	**8,02**	**9,32**	10	250	10000
13,14	14,92	**18,40**	20	250	10000
29,28	34,88	**43,92**	50	250	10000
8,12	**9,24**	**9,76**	10	500	10000
15,46	**17,44**	**19,44**	20	500	10000
35,74	**41,90**	**48,04**	50	500	10000
3,48	1,24	3,92	10	50	50000
4,46	2,06	6,74	20	50	50000
5,80	4,24	12,12	50	50	50000
4,86	2,24	5,50	10	100	50000
6,70	3,44	9,94	20	100	50000
10,14	7,80	19,50	50	100	50000
6,60	6,12	**8,34**	10	250	50000
10,04	10,98	15,10	20	250	50000
17,10	21,58	31,74	50	250	50000
7,98	**8,44**	**9,44**	10	500	50000
12,70	15,54	**17,50**	20	500	50000
24,18	33,50	**40,68**	50	500	50000

Table 4. Real-time trip generation speed results

Trajectories	Scenario 1		Scenario 2		Scenario 3	
	FNN time	DP time	FNN time	DP time	FNN time	DP time
50	0:00:00:001	0:00:00:022	0:00:00:001	0:00:00:017	0:00:00:002	0:00:00:013
100	0:00:00:001	0:00:00:045	0:00:00:001	0:00:00:035	0:00:00:004	0:00:00:023
250	0:00:00:001	0:00:00:109	0:00:00:003	0:00:00:093	0:00:00:009	0:00:00:065
500	0:00:00:002	0:00:00:227	0:00:00:004	0:00:00:183	0:00:00:018	0:00:00:123
2000	0:00:00:008	0:00:00:881	0:00:00:016	0:00:00:653	0:00:00:070	0:00:00:470
10000	0:00:00:041	0:00:04:351	0:00:00:080	0:00:03:141	0:00:00:338	0:00:02:305
50000	0:00:00:191	0:00:21:987	0:00:00:423	0:00:15:972	0:00:01.706	0:00:11:449

The summarized differences between experiment scenarios are presented in Table 5. The most complex experiment scenario produces the most complex neural network model, which is the most accurate to infer through FNN, but slower. The DP speed is slower the less constraints there are on the trip generation.

Table 5. Result comparisons between experiment scenarios

Experiment scenario no.	**Scenario constraints**	**Neural network complexity**	**FNN accuracy**	**FNN speed**	**DP speed**
1	Least constraints	Least complex	Least accurate	Fastest	Slowest
2	Average	Average	Average	Average	Average
3	Most constraints	Most complex	Most accurate	Slowest	Fastest

3.5. Conclusions

The results for the 3rd scenario in the real-time trip generation using FNN show that for cities which contain as much as 50000 trajectories, it is possible to generate as much as 50 trip offers in which at least 80% of them match the best possible offers in under 2 seconds. This shows that the real-time trip generation algorithm can be applied in practice.

Conclusions

1. Conducted literature analysis showed that although there is a lot of research being done in analyzing TSP and its modifications, the solutions are not applicable to our problem, since most of them do not consider the directional nature of commercial flights and the waiting period in each city.

2. The designed frontend API and the graphical user interface fully enable the functional requirements of the system.

3. Conducted research shows that the real-time trip generation algorithm can be applied in practice and be integrated into the trip planning software system. The 3rd scenario in the real-time trip generation using FNN experiment satisfies the functional on non-functional requirements for the algorithm. Also, the results show that for cities which contain as much as 50000 trajectories, it is possible to generate as much as 50 trip offers in which at least 80% of them match the best possible offers in under 2 seconds.

4. Speed evaluation of the neural network models during the experiments proved that more complex models take significantly more amount of time to be inferred.

5. The quality of the project meets the requirements raised by the client and the client is happy with the outcome.

List of references

1. [1] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of computer computations,* pp. 85-103, 1972.
2. [2] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," 1976.
3. [3] E. Klarreich, "Computer Scientists Break Traveling Salesperson Record," *Quanta Magazine,* 2020.
4. [4] C. K. Joshi, Q. Cappart, L.-M. Rousseau and T. Laurent, "Learning the Travelling Salesperson Problem Requires Rethinking Generalization," 2020.
5. [5] D. Gavalas, C. Konstantopoulos, K. Mastakas and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Springer Science,* 2014.
6. [6] R. Gama and H. L. Fernandes, "A reinforcement learning approach to the orienteering problem with time windows," 2020.
7. [7] W. Souffriau, P. Vansteenwegen, J. Vertommen and G. Vanden, "A personalized tourist trip design algorithm for mobile tourist guides," *Applied Artificial Intelligence,* 2008.
8. [8] Y. Mei, F. D. Salim and X. Li, "Efficient Meta-heuristics for the Multi-Objective Time-Dependent Orienteering Problem," *European Journal of Operational Research,* 2016.
9. [9] A. R. S. C. Gopinath Rebala, An Introduction to Machine Learning, Springer, 2019.
10. [10] R. N. B. Kajaree Das, "A Survey on Machine Learning: Concept, Algorithms and Applications," *International Journal of Innovative Research in Computer and Communication Engineering,* 2017.
11. [11] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *Neural Networks,* vol. 61, pp. 85-117, 2015.
12. [12] A. Zell, " Simulation Neuronaler Netze [Simulation of Neural Networks]," no. 1, p. 73, 1994.
13. [13] L. Loss, "Tourism review," 2019.
14. [14] V. Karantzavelou, "Travel & Tourism in 2018 contributed $8.8 trillion to the global economy," 2019.
15. [15] Kayak, "About".
16. [16] N. Galov, "Where is TripAdvisor Going? 39+ Signpost Statistics," 2019.

17. [17] Kiwi, "Company Info".

18. [18] E. H. Y. S. John Duchi, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research 12,* pp. 2121-2159, 2011.

19. [19] A. K. Grantas Gadliauskas, "Machine learning algorithm application in trip planning," *Vilnius University Open Series,* pp. 25-34, 2022.

20. [20] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," 2014.

21. [21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* 2014.

Appendices

Appendix 1. Article published in the Proceedings of the Conference "Lithuanian MSc Research in Informatics and ICT"

Machine learning algorithm application in trip planning

Grantas Gadliauskas, Andrius Kriščiūnas

Kaunas University of Technology, Faculty of Informatics, K. Donelaičio St. 73, 44249 Kaunas, Lithuania

grantas.gadliauskas@ktu.edu, andrius.krisciunas@ktu.lt

Abstract. This article explores how machine learning can be applied in efficiently solving a variation of the Travelling Salesman Problem (TSP) in the context of air travel tourism. Large number of cities create too many trip route combinations to be efficiently evaluated in real time. The method proposed uses a feedforward neural network to narrow down the number of trip route combinations, while a more traditional algorithm based on dynamic programming is then able to select the best trip offers. It was shown that the method could be applied in practice to achieve almost real-time generation of best possible trip offers while evaluating a large amount of real-world flight data.

Keywords: travelling salesman problem, flight search, combinatorial optimization, neural network.

• Introduction

Consider a tourist who wants to visit several different cities in a specific date range in a round trip from his home city. The tourist might also have preferences to which cities one wants to visit or avoid. A list of *N* best possible trip offers then should be provided to the user, based on the real-world flight data. The quality of the trip is determined by its price, but additional metrics could be added.

Since flight data updates very often and the number of possible date ranges is immensely huge it is not practical to pre-calculate all the offers. On the other hand, finding the best offers in real-time is inefficient due to the need to compute the best scored combination of flights for a large amount of possible trip routes.

In the combinatorial optimization domain, the more simplified version of this problem is well known as the Travelling Salesman Problem (TSP) [1]. More recent works on the topic also include machine learning approaches such as one by Chaitanya K. *et al.* [4] which makes use of neural networks to perform TSP efficiently with hundreds of nodes. For our problem, however, the number of nodes (possible trip flights) will never be more than a few hundred, but the more important issue is the number of trip routes growing exponentially because of the number of different cities.

This article proposes a heuristic solution that allows to efficiently find the best trip offers using a feedforward neural network combined with a tree search algorithm based on dynamic programming (hereinafter DP). The feedforward neural network (hereinafter FNN) model can narrow down the total number of possible trip route combinations to a smaller amount of potential best trip candidates, while the algorithm based on dynamic programming is then able to select the *N* best trips. 3 scenarios with different constraints on the trip offer are tested.

- **Method**

Each trip contains a set of cities *T, |T| = N,* with a particular order. A trip starts at the start city t_1 and the last city visited is denoted as t_n. Every trip is a round trip and ends at the start city t_1. The ordered set *T* is referred to as a *trajectory.*

The trajectories used in our experiment are made up from 100 selected European cities. City selections are based on OpenStreetMap Place Importance Score (OSM PIS). Each city has a set of airports assigned to it, which is used to associate flight data with the city. The cities used are marked in Figure.

Each of the trajectories consist of 5 cities. Trajectories are generated based on the real-world flight routes which seldom change. Since there could be a total of $100 \cdot 99 \cdot 98 \cdot 97 \cdot 96$ trajectories, it would be impractical to consider all the possible combinations. Trajectory amount can be reduced by selecting only the most attractive trajectories – ones with the best trajectory round score and combined OSM PIS. Round score is calculated by dividing the total trajectory distance by the minimal possible distance connecting all the cities. Since only 5 cities make the trajectory, calculating the round score is trivial. 210000 trajectories were generated for our experiment. An example of a trajectory with its adjacent cities connected by a blue line is presented in Figure.

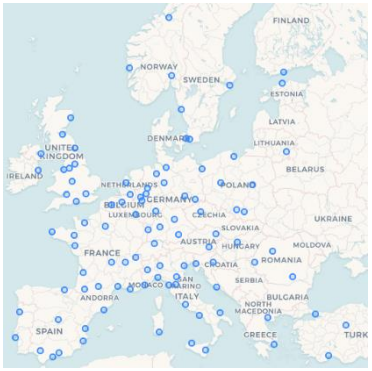

Figure 1. Selected cities

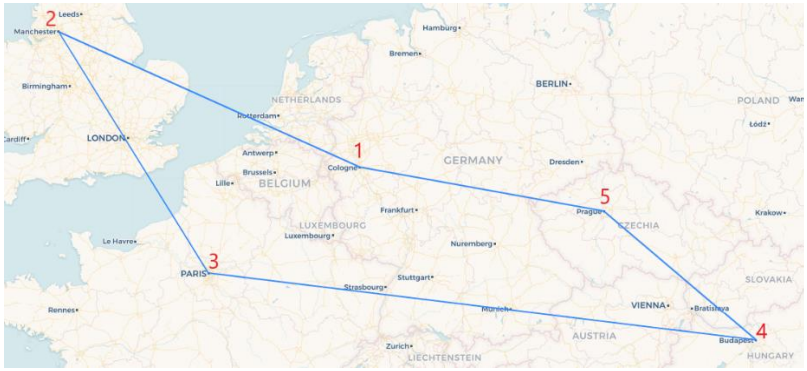

Figure 2. Trajectory graph example

The score of the trip equals to the sum of its flight ticket prices. The lower the score, the better the trip is considered.

Our method to find the best *N* trips given *M* possible trajectories and flight data is as follows:

1. Pass the flight prices of *M* trajectories of the given date range to the FNN model. Each day can have at most a single flight for a given route between two cities.
2. Pass some amount of best predicted trajectories and their flight prices to the DP algorithm.
3. Use the best *N* trajectories returned by the DP algorithm and the flight data to build the best *N* trip offers.

To determine if our method is viable in practice, we evaluate speed and accuracy metrics. Speed is measured as the combined computing time of FNN prediction and DP algorithm. Accuracy is determined by comparing the final output of *N* best trajectories to expected *N* best trajectories and diving the sum of matching pairs by *N*. Accuracy is influenced by the number of total trajectories passed to the FNN model and the number of best model predictions passed to the DP algorithm.

3 trip generation scenarios are explored in this research. They differ by the constraints applied to what can be considered a viable trip.

- Scenario 1: No additional constraints.

- Scenario 2: Each city has a minimum, maximum and preferred number of days to spend in that city. Trip generation must respect the minimum and maximum constraints and apply a penalty if the time spent in a city does not match its preferred number. The penalty subtracts a value of 5 from the trip score per absolute day offset.
- Scenario 3: Same as scenario 2, also, a total trip length constraint is added. The trip length must be in one of the three intervals:
    - 10 – 13 days
    - 14 – 20 days
    - 21 – 24 days

The scenarios are visualized in Figure 31, where the X axis represents the flights, and the Y axis represents the days. The red cells mark the days in which it is impossible to take the flight to match the given constraints. The green cells represent the possible days to take flights if trip is starting from the earliest day (03-28). Yellow cells represent other possible days if trip were to start from another day. With each scenario, the amount of possible flight combinations is reduced. In the Figure 31 example, the min. and max. number of days to stay in every city are 3 and 7 respectively and the total trip length should span from 14 to 20 days. In our experiment, for all scenarios we constrain the maximum trip length to 24 days.

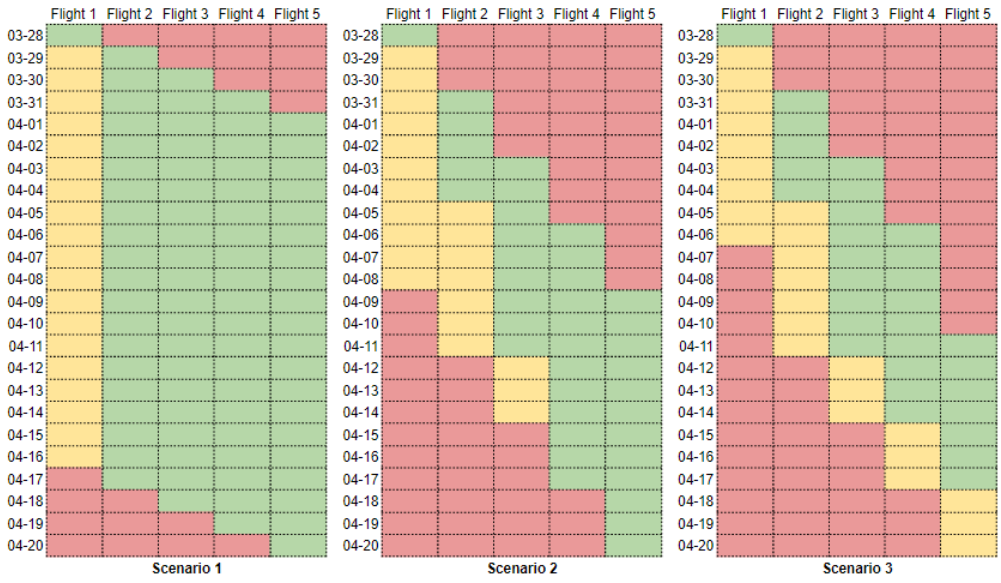

Figure 3. Testing scenarios

- **Experiment**

The computing environment used for this research has the following parameters:

- RAM memory: 32 GB
- CPU: AMD Ryzen 7 3700X 8-Core Processor 3.60 GHz (1 CPU core used)
- OS: Windows 10 64-bit architecture

Python 3.9 programming language is used to program the software. PyTorch machine learning framework is used to create the neural network model.

Common neural network training hyperparameters for each scenario are as follows:

- Batch size: 128

- Optimizer: Adam [19]
- Loss function: Mean absolute error (MAE)
- Learning rate: Reducing learning rate on plateau by a factor of 0.5 on 3 consecutive epochs without improvement.
- Epochs: Until does not improve for 10 epochs or until 60.

The departure dates for all the flight data in the experiment span between 2021-03-28 and 2021-05-15. Flight data from 2021-03-28 to 2021-04-20 is used for training, while data from 2021-04-21 to 2021-05-15 is used for validation. For scenario 1, flight prices of 24 days for each of the 5 trajectory cities are passed to the model as an input, for a total input length of 120. If the flight data for a particular day is missing, it is passed to the model as a value of -1. For scenario 2, the number of min., max. and preferred days for each city is added to the input, which increases the input length to 135. For scenario 3, numbers for min. and max. trip length are added, for an input length of 137. The target trip scores for the model training were built using the DP algorithm. If not a single trip can be built for a trajectory under certain constraints, the target is set to a value of 1000. The model outputs a single value – a trip score. 210000 inputs were used to train model for scenarios 1 and 2, while scenario 3 trained with three times number of inputs (630000) due to three distinct intervals used for total trip length.

Neural network validation accuracy during training for each scenario is presented in figures Figure 32, Figure 33 and Figure 34. The used notation to describe the model architectures in the figure legends is as follows: $I\ x\ H * N\ x\ O$, where I is the number of inputs for the input layer, H is the number of inputs for each hidden layer, N is the number of hidden layers and O is the number of outputs (1 output describing the trip score). The best model architecture is highlighted with a yellow marker. In general, to obtain the optimal validation accuracy, the models had to become more complex as the trip constraints increased. Model for scenario 2 tends to overfit the most and the model state after 4th epoch is used for its metric check. Techniques such as dropout [20] and dataset scaling were tested but failed to improve the model accuracy.

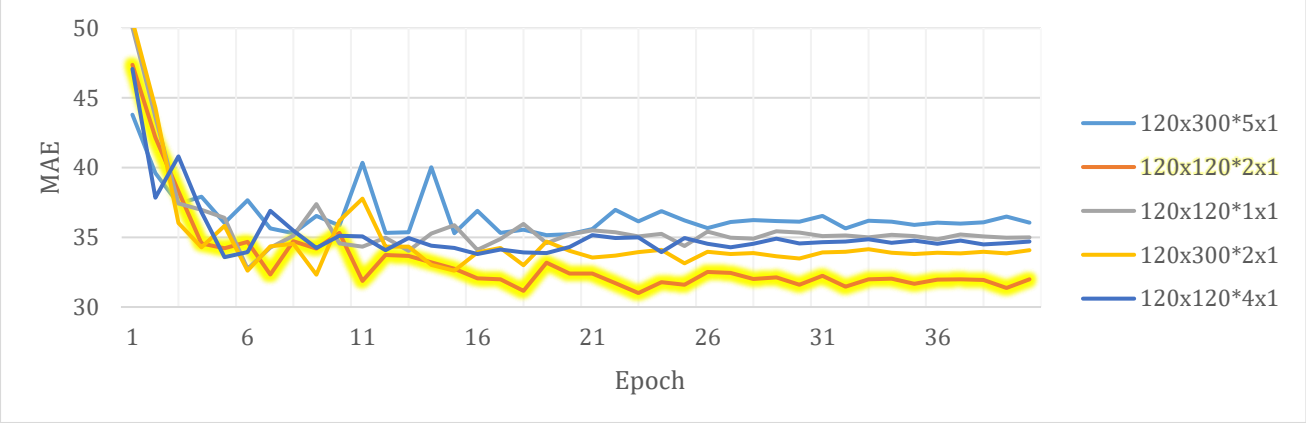

Figure 4. Scenario 1 validation accuracy

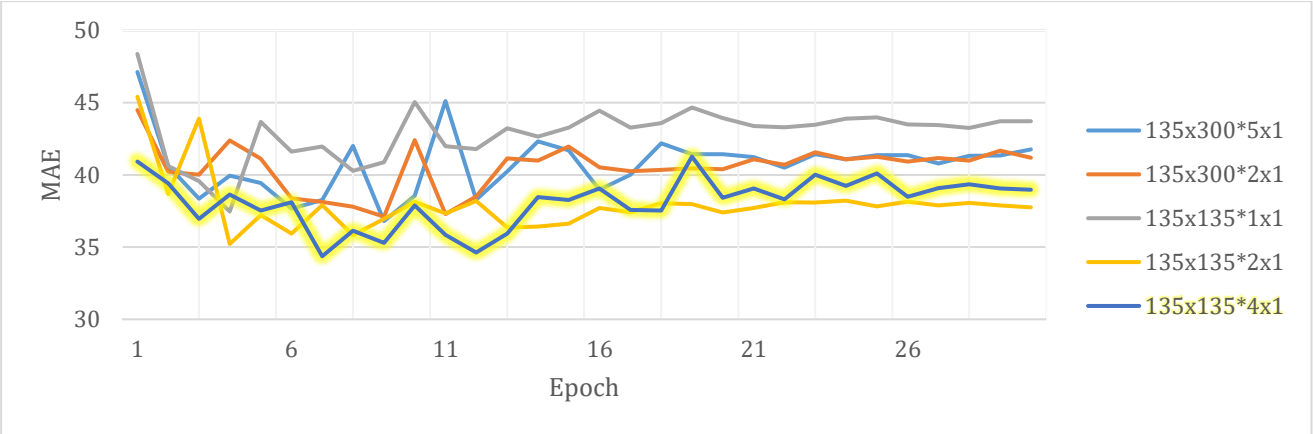

Figure 5. Scenario 2 model validation accuracy

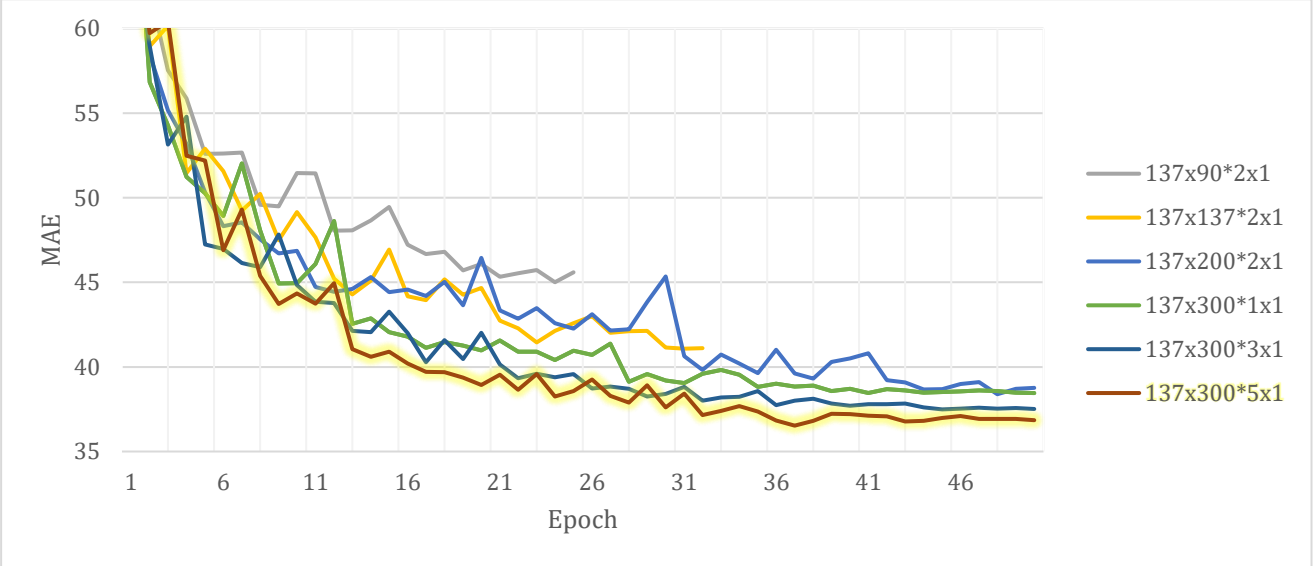

Figure 6. Scenario 3 model validation accuracy

The dynamic programming algorithm is used to find the actual best possible flight combination for a given trajectory. It works like a brute force tree search algorithm, but with optimizations. Instead of checking each possible flight combination, it stops traversing the flights if the flight for a given day was already traversed and had a better total flight price. It also only iterates through the dates which match the constraints of the min. and max. days to stay in a certain flight city and ignores days which do not match the total trip length constraint (such days are marked red in Figure 31).

- **Results**

The final accuracy and speed results are presented in tables Table 3 and Table 4. Accuracy match results were averaged over 50 test runs. In the accuracy result table, cells marked in red, yellow, and green represent respectively the worst, the second best and the best scenario for the testing parameters of the rightmost 3 columns. Columns "*Total predictions made*", "*Top N predictions to search in*" and "*Required top N matches*" denote respectively how many inputs were passed to the FNN model, how many of the best results then were passed to the DP algorithm, and how many final trip offers do we want to output. The cell values in bold mark the values for which at least 80% of the required top *N* matches do match, which is considered a good result.

It is possible to infer from the accuracy results that the more constrained the trip generation scenario is, the more accurate the final matches tend to be. Since the mean absolute

error of the FNN model validation accuracy was lower for the more constrained models, this might not seem reasonable. However, it may be explained by the greater value of the standard deviation of more constrained scenario model target array (trip scores) compared to less constrained scenario targets. The final matches tend to be less accurate the more trajectory inputs are passed to the FNN model and the fewer top predictions are ran through the DP algorithm.

The speed results show that the performance of the FNN is extremely quick running faster than half of a second for 50000 trajectory inputs in 1st and 2nd scenario and in 1.7 seconds in 3rd scenario, which uses a more complex neural network architecture. The DP algorithm time decreases as the amount of trip constraints increases.

• Conclusions

In this article it was investigated if combining the speed of feedforward neural networks and the accuracy of traditional search algorithms can be used to quickly generate attractive trip offers using real world flight data. The results show that for cities which contain as much as 50000 trajectories, it is possible to generate as much as 50 trip offers in which at least 80% of them match the best possible offers in under 2 seconds under the constraints of this experiment. This shows that the method can be applied in practice, and it will be strongly considered to be integrated into a newly developing trip planning software system.

References

[1] R. M. Karp, „Reducibility among combinatorial problems," *Complexity of computer computations,* pp. 85-103, 1972.

[2] C. K. Joshi, Q. Cappart, L.-M. Rousseau, T. Laurent, „Learning the Travelling Salesperson Problem Requires Rethinking Generalization," 2020.

[3] D. P. Kingma, J. Ba, „Adam: A Method for Stochastic Optimization," 2014.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, „Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research,* 2014.

Table 1. Accuracy results

Actual matches			Required top N matches	Top N predictions to search in	Total predictions made
1 Scen.	2 Scen.	3 Scen.			
5,48	6,72	**8,72**	10	50	2000
10,86	12,20	**16,22**	20	50	2000
23,04	23,84	31,78	50	50	2000
6,76	**8,26**	**9,50**	10	100	2000
13,92	15,38	**18,42**	20	100	2000
32,42	34,18	**41,74**	50	100	2000
8,96	**9,82**	**9,82**	10	250	2000
18,16	**19,26**	**19,46**	20	250	2000
43,62	**44,76**	**47,58**	50	250	2000
9,84	**9,98**	**9,84**	10	500	2000
19,70	**19,94**	**19,56**	20	500	2000
48,58	**48,90**	**48,82**	50	500	2000
3,60	3,92	6,08	10	50	10000
6,26	6,50	11,00	20	50	10000

11,32	12,32	21,28	50	50	10000
5,16	6,32	**8,12**	10	100	10000
9,22	11,06	15,34	20	100	10000
19,56	22,62	32,82	50	100	10000
7,00	**8,02**	**9,32**	10	250	10000
13,14	14,92	**18,40**	20	250	10000
29,28	34,88	**43,92**	50	250	10000
8,12	**9,24**	**9,76**	10	500	10000
15,46	**17,44**	**19,44**	20	500	10000
35,74	**41,90**	**48,04**	50	500	10000
3,48	1,24	3,92	10	50	50000
4,46	2,06	6,74	20	50	50000
5,80	4,24	12,12	50	50	50000
4,86	2,24	5,50	10	100	50000
6,70	3,44	9,94	20	100	50000
10,14	7,80	19,50	50	100	50000
6,60	6,12	**8,34**	10	250	50000
10,04	10,98	15,10	20	250	50000
17,10	21,58	31,74	50	250	50000
7,98	**8,44**	**9,44**	10	500	50000
12,70	15,54	**17,50**	20	500	50000
24,18	33,50	**40,68**	50	500	50000

Table 2. Speed results

Trajectories	**Scenario 1**		**Scenario 2**		**Scenario 3**	
	FNN time	**DP time**	**FNN time**	**DP time**	**FNN time**	**DP time**
50	0:00:00:001	0:00:00:022	0:00:00:001	0:00:00:017	0:00:00:002	0:00:00:013
100	0:00:00:001	0:00:00:045	0:00:00:001	0:00:00:035	0:00:00:004	0:00:00:023
250	0:00:00:001	0:00:00:109	0:00:00:003	0:00:00:093	0:00:00:009	0:00:00:065
500	0:00:00:002	0:00:00:227	0:00:00:004	0:00:00:183	0:00:00:018	0:00:00:123
2000	0:00:00:008	0:00:00:881	0:00:00:016	0:00:00:653	0:00:00:070	0:00:00:470
10000	0:00:00:041	0:00:04:351	0:00:00:080	0:00:03:141	0:00:00:338	0:00:02:305
50000	0:00:00:191	0:00:21:987	0:00:00:423	0:00:15:972	0:00:01.706	0:00:11:449

Appendix 2. Certificate of participation in the Conference "Lithuanian MSc Research in Informatics and ICT"

Pažymėjimas

Grantas Gadliauskas

Kauno technologijos universitetas

dalyvavo III-oje konferencijoje

Lietuvos magistrantų informatikos ir IT tyrimai

vykusioje 2022 m. gegužės 16 dieną
ir skaitė pranešimą

Machine learning algorithm application in trip planning

Programinis komitetas:

Prof. Gintautas Dzemyda

Prof. Olga Kurasova

Prof. Julius Žilinskas

Vilnius, 2022-05-16

Vilniaus universitetas