UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Lucas Gabriel da Silva Félix

Planet Caravan: Gerador de Rotas Completas para Viagens

Belo Horizonte
2022

Lucas Gabriel da Silva Félix

Planet Caravan: Gerador de Rotas Completas para Viagens

Versão Final

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais, como requisito parcial à obtenção do título de Mestre em Ciência da Computação.

Orientador: Pedro Olmo Stancioli Vaz de Melo
Coorientadora: Carolina Ribeiro Xavier

Belo Horizonte
2022

Lucas Gabriel da Silva Félix

Planet Caravan: Full Trip Planner

Final Version

Thesis presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

Advisor: Pedro Olmo Stancioli Vaz de Melo
Co-Advisor: Carolina Ribeiro Xavier

Belo Horizonte
2022

Félix, Lucas Gabriel da Silva.

F316p Planet caravan: [manuscrito] full trip planner / Lucas Gabriel da Silva Félix — 2022.
119 f. il.

Orientador: Pedro Olmo Stancioli Vaz de Melo.
Coorientadora: Carolina Ribeiro Xavier.
Dissertação (mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação
Referências: f. 100-108.
1. Computação – Teses. 2. Sistemas de recomendação – Teses. 3. Sistemas de recomendação – Turismo –Teses. 4. Sistemas de recomendação - Pontos de interesse para turismo –Teses. 5. Heuristica – Teses. I. Melo, Pedro Olmo Stancioli Vaz de. II. Xavier, Carolina Ribeiro. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. IV. Título.

CDU 519.6*82 (043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6/1510 – Insituto de Ciências Exatas da UFMG

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

PLANET CARAVAN: A FULL TRIP PLANNER

LUCAS GABRIEL DA SILVA FELIX

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. PEDRO OLMO STANCIOLI VAZ DE MELO - Orientador
Departamento de Ciência da Computação - UFMG

PROFA. CAROLINA RIBEIRO XAVIER - Coorientadora
Departamento de Ciência da Computação - UFSJ

PROFA. JUSSARA MARQUES DE ALMEIDA GONÇALVES
Departamento de Ciência da Computação - UFMG

PROFA. FERNANDA SUMIKA HOJO DE SOUZA
Departamento de Computação - UFOP

Belo Horizonte, 11 de Março de 2022.

This work is dedicated to all essential people present in my life.

Acknowledgments

First of all, I would like to express my gratitude to God for giving me the health and patience to finish my master's degree.

To my wife Clara, I would like to thank for being my number one supporter, and for all love, affection, faith, and patience that were fundamental for me to finish this work.

To my mother, Heloisa, I would like to thank her for all the love and dedication, and for being my inspiration in the search for knowledge.

To my father and to my stepmother, Mario and Selma, for all their support during this period and during the pandemics.

To my advisors Pedro and Carol, for all the patience and teaching during this period. All the help and knowledge were essential to this work.

To my friends Washington, Amir, Christian, Kevin, Kelvin, Bruno, Diego, Thiago, Carlos Magno, Gleyberson, João Caetano, and João Moreria for all the technical and non-technical support during this period.

Finally, my thanks to all those who contributed, directly or indirectly, to the development of this work

"*Parece que a possiblidade de abrir todas as portas faz com que ninguém entre em nenhuma delas de corpo e alma.*"

(Humberto Gessinger)

Resumo

Um dos setores que mais se beneficiou da expansão da internet foi o turismo. A internet permitiu que as pessoas pudessem compartilhar informações de suas viagens, auxiliando outros turistas a decidir os melhores destinos. Devido a grande quantidade de dados em redes sociais e plataformas especializadas em turismo disponíveis (e.g. Foursquare, TripAdvisor), se tornou cada vez mais difícil escolher quais os melhores locais para se visitar, trazendo aos usuários o problema conhecido como sobrecarga de informação ou paralisia por análise. Assim, considerando que é difícil identificar um local para ser visitado, planejar uma viagem completa é considerado um cenário ainda mais difícil.

Para atenuar este obstáculo, técnicas computacionais como Sistemas de Recomendação (SR) têm sido utilizadas. Contudo, considerando o contexto turístico, técnicas tradicionais de SR não levam em consideração diversas variáveis consideradas importantes neste cenário, como preço, distância e horário de funcionamento dos locais. Assim, visando modelar da melhor maneira este problema, os trabalhos focam na tarefa de planejamento automático de viagens, também conhecido como Orienteering Problem (OP). Por sua vez, o OP compreende na tarefa de identificar um caminho em um grafo o qual maximiza a utilidade do usuário, enquanto respeita uma restrição do custo do caminho. Esta é uma tarefa considerada computacionalmente cara e geralmente é resolvida por meio de heurísticas de otimização. Para modelar da melhor maneira possível restrições de cenários reais, trabalhos na literatura adicionam diferentes restrições (e.g. horário de funcionamento de locais, custo de vistas). Contudo, muitos destes deixam de lado aspectos importantes em cenários reais como personalização, viagens de múltiplos dias e seleção de hotéis.

Assim, neste trabalho nós atacamos o problema descrito acima por meio de uma metodologia composta de um SR, responsável pela personalização das rotas geradas, e heurística de otimização, responsável por atender as diversas restrições de uma viagem real. Nós denominamos nosso metodologia como Planet Caravan.

Para avaliar a metodologia proposta, nós testamos diferentes técnicas SR e heurísticas de otimização, introduzimos 5 novas bases de dados coletadas do TripAdvisor as quais possuem dados de cidades no Brasil e Europa, e também uma restrição de compromisso que permite aos turistas gerar rotas em torno de sua agenda. Nossos resultados mostram que heurísticas como **Algoritmo Genético** e **GRASP** possuem os melhores resultados entre as técnicas de otimização avaliadas. Por último, avaliamos a metodologia proposta com usuários reais por meio de uma aplicação web. Nossos resultados são promissores e mostram o potencial deste trabalho para aplicações com turistas reais.

Palavras-chave: Sistemas de Recomendação; Heurísicas de Otimização; Turismo; Recommendação de Tours; Pontos de Interesse.

Abstract

One of the services that most benefited from the internet expansion was tourism. The internet allowed people to share information about their trips, assisting other tourists to decide the best destination. However, the high data availability on social networks and platforms specialized in tourism (e.g. Foursquare, TripAdvisor), made it difficult to choose the best places to visit, bringing a new problem known as information overload or analysis paralysis. Thus, considering that is difficult to identify one place to visit, planning a complete trip is considered to be an even harder scenario.

To tackle this problem, computational techniques such as Recommender Systems (RS) are been applied. Nevertheless, considering the tourism context, traditional RS techniques do not take into consideration several variables that are important in tourism such as the venues' price, distance, and working hours. Hence, aiming to better model such problem works focus on the task of automating trip planning, known in the literature as Orienteering Problem (OP). The OP consists of the task of identifying in a graph the path that maximizes the users' utility while respecting a distance constraint. This task is considered to be computationally expensive and is usually solved through optimization heuristics. To meet the real-world constraints in a trip works in the literature consider different constraints (e.g. places working hours, visit costs). However, several of these works leave aside important aspects for the real-world cases, such as personalization, multiple days of travel, and automated hotel selection.

In this work, we tackle the problem presented above with a methodology composed of an RS, responsible for the personalization of the generated routes, and an optimization heuristic, responsible for meeting the several constraints considered in a travel. We named our methodology as Planet Caravan.

To evaluate the proposed methodology, we test different techniques on RS and optimization heuristics, introducing 5 new datasets collected from TripAdvisor of cities in Brazil and Europe. Also, we introduce an appointment constraint that enables users to generate routes around their schedules. Our results show that the **Genetic Algorithm (GA)** and **GRASP** are the most suited technique in our evaluation scenarios. Lastly, we evaluate the proposed methodology with real users through a web application. In this scenario, our results show the potential of our proposal with real users.

Keywords: Recommender Systems; Optimization Heuristics; Tourism; Tour Recommendation; Points of Interest.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Motivation

Before the internet, the spread of tourism information relied on two main ways: personal experiences exchange, from one person to another, and paper-based recommendations, made by guides and magazines [46, 44]. In both cases, these recommendations influenced other people's choices [46, 86]. Nevertheless, these were in general generic suggestions and were based on one person's opinion. In a scenario as intangible and heterogeneous as tourism, many voices can bring many points of view [3], however, these recommendations were restricted to the personal social circle.

In this scenario, the internet was one of the catalysts for tourism development [86]. The advance in connectivity impact directly this industry, which has increased its value since the 2009 global crisis [40] until the 2020 COVID-19 pandemic, showing how this was one of the services that most benefited from the internet expansion [86].

Through the internet, users can now gather in forums, communities, travel blogs, and Social Networks (SNs). These platforms made it possible to seek Points of Interest (POI), provide and receive information about places, sharing experiences from a unique point of view. Studies show that 80% of American travelers use SNs while traveling and more than half of this percentage share information of their journey with their contacts [86].

These studies also show how SNs are the tool most adopted between travelers[86]. Between these platforms, three different types of SNs are generally used: General-Purpose SN, Location-Based SN (LBSN), and Travel SN (TSN). General-purpose SNs as Facebook and Instagram, usually contain general information and not only content related to tourism. Even though these platforms enable their users to geo-localize the shared content, these are usually restricted to the user's social circle and are not structured to easily identify the tourist opinion about the place through a rating or a review.

The last two types of platforms are more focused on tourism and have been extensively used in the literature in tourism works [40, 44]. With different purposes, LBSNs,

as Yelp and Foursquare, are more centered on sharing the user's current location sharing their footsteps, while the TSNs, like TripAdvisor, are responsible for joining in one platform several pieces of information about locations, accommodations, transport, food, attractions, and services [86]. Aside from that, TSNs also give the user the possibility to plan trips and get recommendations about users with the same taste [41], while comparing prices and looking at different perspectives about a place.

Even though TSN platforms concentrate most of the information needed to plan a trip in one place, users still experience difficulty with the amount of data available, precluding to distinguish which option (or set of options) is the best [59]. To tackle this problem, Recommender Systems (RS) are usually applied, focusing on suggesting POIs based on the user history. RS approaches tackle the information overload problem reducing the number of options for the users and enabling them to choose between a smaller set. However, while planning a trip not only the taste of users can be considered, but also aspects as the price, working hours, and distance can influence the user experience, which is not considered by the RS. In conclusion, planning a trip is a complex and time-consuming process [44].

To facilitate this process and tackle this problem, several works on the literature focus on the task of automating trip planning. The approaches available in the literature classify the task of selecting and touring between places a hard-to-solve computational problem, that can be mapped into the Orienteering Problem (OP) [30, 75]. As stated in [78], this problem is a combination of the traveling salesman and the knapsack problem, where the goal is to find a path in a graph that maximizes a utility function, while respecting a distance constraint. It is noteworthy that in the literature a wide variety of restrictions have been taken into consideration for this problem [44]. The OP modeling can be easily mapped to the tourism scenario, if we consider that the user wants to find the route that maximizes his benefit while respecting money and distance budgets. When applied to the tourism scenario, this problem is also known as Tourism Trip Design Problem (TTDP), and besides taking into consideration the user constraints, it adapts for more realistic scenarios, also considering the environment and place constraints, as if the place is open, its opening and closing hours and the places category [28].

The routes recommended by these approaches are an advance in comparison to the word of mouth or paper-based recommendations, due to the fact that they summarize the information available online considering the opinion of many peers and not only a unique expert. Despite these recent advances, the routes generated still fail to properly model the constraints needed for a tourist. For example, most works do not consider multiple-day travel or a hotel selection in their work [53, 78, 14, 65]. Consequently, these works have little use in practice given that tourists usually travel at least 3 days [24]. In this scenario, the work of [44] made a big advance enabling users to generate personalized routes with hotel selection for multiple days of travel. Still, this work fails to test their approach in

practice with real users and does not evaluate their modeling with proper data, given that LBSN does not present attributes of the places and only the user history check-in.

Given these facts, in this work, we tackle the TTDP by proposing an architecture composed of a **recommender system and a optimization heuristic**, which are responsible for personalization based on user history and respecting several users, places, and environmental constraints. We name our methodology Planet Caravan.

We present new modeling for TTDP, testing several **Collaborative Filtering (CF)** RS and optimization techniques as a **Genetic Algorithm (GA)**, **Clonalg**, and **GRASP**. We test our modeling with proper data gathered from TSN, namely TripAdvisor, using datasets from 5 different cities in Brazil and Europe. Besides, we introduce a new restriction that enables users to personalize their routes even more, by defining the start and end hours to visit specific places, which we call an appointment constraint. This restriction enables to generate a personalized route around the user schedule.

1.2 Problem Statement

Recall that the problem we face in this work is personalized automated route planning recommendations, which as also known as Orienteering Problem or Tourist Trip Design Problem. In this work we make a per-user tailored route recommendation, considering users' preferences, place, environment, and user constraints. To solve this problem, we use a methodology that joins RS and optimization heuristics, first focused on identifying for each user the rating for not visited places, while the second aim to respect all constraints defined in our problem.

POI Recommendation/Matrix Completion In our first task, consider a city with n places, and a tourist which is visiting such city. This user is represented by a vector of ratings $R_i \in \mathbb{R}_+$, which contains a rating to each place i previously visited by this user. Considering that the user visited k of the n places in the city, the goal of this task is to fill R such that a personalized (based on previous experience) rating (utility) to the $n-k$ non-visited places is given. Figure 1.1, shows a small example of the matrix completion process in a city with 5 different places and 3 users.

Route Generation Considering a simpler version of the Tourist Trip Design Problem, this problem can be formulated as follows. Let $G = (V, E)$ be a directed-complete graph representing the city. Let $V = \{v_0, v_1, ..., v_n\}$ represent the set of vertices which correspond to places available to be visited, where each place $v \in V$ is associated with a utility $u_v \in$

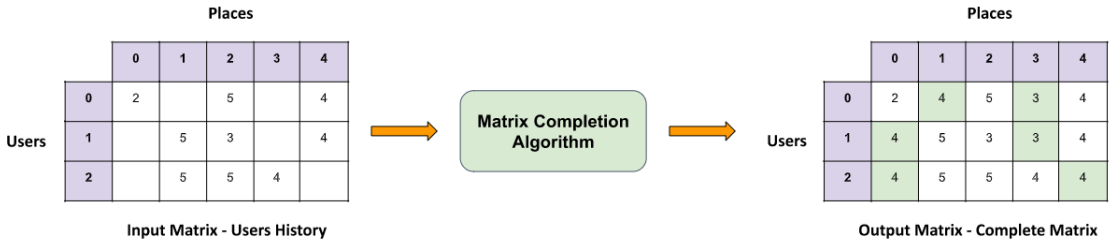


Figure 1.1: Example of route represented in the mathematical modeling

$\mathbb{R}_+$, which where defined in the step before. A set of edges $E = \{(i, j) \in V \times V : i \neq j\}$ connects each place, where the edge $e_{i,j}$ exists if a place i is available to be visited from j. Each edge connecting the POIs has its corresponding weight $W = \{w_{i,j}, i \neq j\}|W_{i,j} \rightarrow \mathbb{R}_+$, that denotes the distance between two places. For each tourist, a constant budget $B_i \in \mathbb{R}_+$ defines the maximum distance that the tourist is willing to travel. Hence, our aim is to find a cycle (e.g. the path that starts and ends at the same node) in G that maximizes the utility u while respecting the budget B that limits the sum of edges weight (i.e. the sum of distances).

1.3 Research Questions

Our goal with this work is to propose a new model capable of generating personalized routes while respecting a thorough set of constraints. These constraints are listed below:

- **Money budget:** Maximum amount of money that the user is willing to spend in the trip.
- **Distance budget:** How many kilometers a user can walk per day.
- **Activity hours:** Restricts the users activity hours.
- **Amount activities per day:** Maximum amount of places that can be visited in one day.
- **Mandatory places:** Mandatory places to be visited.
- **Appointments:** Mandatory places that must be visited within a schedule.
- **Places working hours:** Visits are limited to the places working hours.

- **Places category:** Restricts the amount of visits in a day that can be made in places at a given category (e.g. restaurants).
- **Places working day:** Restricts visits when a place is not open for visitation.

Lastly, we consider the climate as an environmental constraint. Even though this is not an explicit restriction in the modeling, we filter outdoor places when the climate condition is not ideal. We formally define these constraints in Chapter 4.

To solve such a problem generating a personalized route that respects all the constraints listed above, we define a methodology divided into two main steps, a recommender system, and optimization heuristic which are responsible for personalization and generation of a feasible route, respectively. Both these steps have a vast literature that motivated us to evaluate different methods.

In the first step, two types of RS techniques are used for the POI recommendations task. First, more generalist techniques, suited for different scenarios, based on collaborative filtering. Second, approaches specifically designed for POI recommendations, being tailored for LBSN datasets. In this scenario, we define our first research question, aiming to evaluate which of these different approaches are better suited for our context.

In the second step, due to the high computational complexity to attend to the constraints defined in our modeling, exact algorithms are not suited for solving such problems [27]. Thus, works on the literature apply different optimization heuristics to solve the TTDP in a smaller computational time [44, 45, 74]. Given that different approaches have been used in the literature, in this work we select 5 different optimization heuristics focused on solving the TTDP. We develop a **Genetic Algorithm, Clonalg, Grasp, Greedy and Random** approaches. The last two are naive models that we use to compare the more complex methods. Due to the several approaches that can be used to solve the TTDP, we define as our second research question which heuristic is better suited for the TTDP.

Lastly, our modeling considers several real-world constraints, and these can impact differently on the user experience. In our third research question, we investigate the impact of the constraints in the routes generated.

As such, our main research questions are:

RQ 1: Which technique of Recommender System is better suited for defining the user ratings for each place in the TripAdvisor dataset? Different from previous works that only analyze one recommender system technique, we compare several approaches, defining the one that better suits the dataset that we use.

RQ 2: Which heuristic is better suited for solving the Tourist Trip Design Problem? We make an extensive study comparing 5 different heuristics for our modeling of TTDP. We analyze the quality of the routes generated by each heuristic discussing the scenarios where each heuristic is better suited.

RQ 3: What is the impact of the constraints in the routes generated? We evaluate the routes generated under different scenarios varying the cities, users, and constraints. By doing so, we verify the impact of the users' constraints on the quality of the routes generated, also analyzing scenarios where is harder to recommend feasible routes.

1.4 Contributions

In this work, we introduce new modeling for the TTDP, focusing on generating personalized routes that are closer to what a user expects on a trip. We design a formulation for the problem that enables users to plan multiple days of personalized tours with hotel selection, with a proposal similar to [44]. Our main differential is a new appointments constraint that enables users to generate routes around their schedule. Besides that, we make an extensive evaluation of our modeling evaluating different RS and optimization heuristic techniques. We evaluate these strategies with 5 new datasets containing data available from cities around the globe. Lastly, due to the high practical application of the TTDP, we evaluate our modeling with real users in 5 of these cities, through the Planet Caravan web application that enables users to easily plan a trip.

Chapter 2

Background

This chapter presents a description of techniques that we use for the tasks of (i) POI recommendation (Section 2.1) and (ii) Route recommendation through optimization heuristics (Section 2.2). Recall that our main task is to recommend personalized routes for a user while respecting constraints. Thus, recommender systems are in charge to identify user preferences given the user history, predicting a rating ($\hat{y}$) for each POI that was not visited by the user. The heuristics focus on generating feasible routes that maximize a utility (i.e. the sum of $\hat{y}$) while respecting user, place and environment constraints.

2.1 Recommender Systems

Recommender Systems (RS) are algorithms developed to provide to a user through the learning of historical data personalized suggestions [54]. These algorithms tackle the problem known as information overload [94] or choice paralysis [25], by suggesting the most suitable items to specific users, predicting the consumer interest [54]. In our context, the RS is responsible for predicting **to each place** p **not visited by the user** u a rating ($\hat{y}_p$) that summarizes how much u will like that place.

In the literature, recommender system methodologies are divided into three main categories which we briefly describe below.

1. **Collaborative Filtering (CF)**: Algorithms in this category are based on user past interactions with items to make estimations about similar items which could be consumed [66].

2. **Content Based (CB)**: Algorithms in this category analyze a set of descriptive features of items previously rated by a user and build a model based on user history [52, 63].

3. **Hybrid Methods**: Algorithms in this category combine the advantages of different types of recommenders to improve the accuracy capacity in a recommendation task [2].

In this work, we focus on the use of CF techniques for the recommendation step of our work. As we will present in Section 2.1.1, CF approaches are well suited for several scenarios achieving good results and not requiring many features to provide meaningful recommendations. In opposite, content-based and hybrid methods require datasets with large amounts of descriptive features. Even though the datasets used in our work present this characteristic, the high cost of data treatment and the focus on both RS and optimization techniques lead us to focus on CF methods and define as future work the use of more complex methodologies in RS.

2.1.1 Collaborative Filtering

Collaborative Filtering (CF) methods rely on user past behavior [42] identifying what is popular among similar users to produce new recommendations [66, 19]. As stated in [7], CF approaches can provide meaningful and accurate recommendations and do not require user profile and domain knowledge, making it much simpler to model the data and avoiding extensive data collection [42].

Techniques based on CF produce new recommendations by modeling the interactions between two different objects: items and users [42]. To model these interactions using a CF approach there are two main methodologies established in the literature: Memory-based (i.e. neighborhood) and Model-based (i.e. latent factors) approach [63].

Memory-based techniques focus on the recommendation based on what is popular among close neighbors. The algorithm works by computing the relationships between items or users [42], which can lead to two main modelings in memory-based techniques: Item-Item and User-User. An item-item approach evaluates the user preference for an item based on the history ratings (i.e. items previously consumed) [63]; A user-user approach evaluates the user preference for an item based on the neighbor's history ratings.

Model-based techniques are mostly presented by latent factor models, such as SVD and NMF. To perform recommendation model-based approaches, transforms both items and users to the same latent factor space, making items and users directly comparable [63]. As stated in [42], "the latent space tries to explain ratings by characterizing both products and users on factors that were automatically inferred from user feedback".

In the following section, we introduce the mathematics behind each of the techniques used exploiting the upsides and downsides of the usage, we divide the algorithms into three groups Neighborhood Models, Latent Factor Models, and Naive Models.

2.1.2 Neighborhood Models

K-Nearest Neighbors (kNN) is a simple yet effective machine learning technique. The main idea of this model is to identify the closest neighbors of an item that we want to classify in space and categorize it based on their neighbors' label [60]. Even though **kNN** is relatively simple and intuitive [42], this algorithm has shown to be efficient in several scenarios [19, 11], especially in the recommendation context. Besides that, algorithms based on **kNN** naturally provide intuitive explanations and recommendations based on users' recent feedback [42].

In this work, we deal with a regression task, thus, we use several **kNN** techniques to provide personalized place recommendations. Basic **KNNs** prediction Equation 2.1 is presented below:

$$\hat{y}_{ui} = \frac{\sum\limits_{j \in N_u^k(i)} sim(i,j) \cdot r_{uj}}{\sum\limits_{j \in N_u^k(i)} sim(i,j)}, \tag{2.1}$$

where u is the target user, i is the target item, j is a user history item, k is the number of neighbors, $N_u^k(i)$ is the set of nearest neighbors of item i that were rated by the user u, $r_{u,j}$ is the user history rating for the item j, and $sim(i,j)$ is a similarity function that defines how related are the items i and j.

As stated in [42], one of the core aspects of memory-based CF is the similarity measure between items/users. Hence, the success of the method depends on the metric used to find the most similar users [7]. This work combines **kNN** with several similarity metrics, verifying which one is more suited for our context. These metrics are defined below:

- **Cosine Similarity**: Measures the likeness between two items by capturing the angle between both items when they are represented in the user space [7]
- **Pearson Similarity**: Measures the correlation in terms of rating patterns received by two items represented in a user-space (each dimension corresponds to a user that rated that item) [7]
- **Pearson Baseline**: Measure the similarity of the deviations of users and items [63]

- **Mean Squared Difference (MSD)**: Measures the rating difference that two users have given an item [77]

Lastly, we point that variations of the **kNN**, namely **kNN Mean**, **kNN Z-Score**, were used in this work. We present their prediction Equation in Chapter A.

2.1.3 Latent Factor Models

Singular Value Decomposition (SVD) is a latent factor technique that models both users and items in a joint latent factor space of dimensionality f such that interactions between users and items are modeled as inner products [43].

Consider that each item i is associated with a vector $q_i \in \mathbb{R}^f$, q_i measure how much a item have those factors, while $p_u \in \mathbb{R}^f$ measures the interest that a user has in items that most represent a factor [63]. Thus, the resulting dot product between q_i^T and p_u captures the interaction between a user and an item. As stated in [63] this dot product captures "the overall interest of the user in characteristics of the item". Equation 2.2 shows how the ratings are predicted by **SVD**. The Equation is also taken into consideration the global average rating μ and the baseline predictors b_i and b_u, that capture the variance in ratings.

$$\hat{y}_{ui} = \mu + b_u + b_i + q_i^T p_u \tag{2.2}$$

When compared to classic neighbor models, **SVD** can increase prediction accuracy in most scenarios [42]. Besides that, this model is flexible allowing to integrate of multiple data forms [43] and tackling the problem of scalability in RS given that model-based approaches perform dimensionality reduction [8]. However, this technique along with all other model-based techniques has the downside of hard interpretation of the latent factors, been explainability an advantage for memory-based models [42]. Also, **SVD** assumes that the input matrix of the algorithm must be complete (i.e. without missing values). Nevertheless, in the RS scenario, the majority of ratings are unknown [42], in some cases achieving about 99% of sparsity.

SVD++ Tackle the problem of missing values using a gradient descent technique that models only the observed ratings while avoiding overfitting through a regularized model [42]. Besides, **SVD++** add a second set of item factors $y_i \in \mathbb{R}^f$, where y_i characterize users based on the set of historical items [63]. Equation 2.3 presents how **SVD++** predicts the item rating.

$$\hat{y}_{ui} = \mu + b_u + b_i + q_i^T \left(p_u + |I_u|^{-\frac{1}{2}} \sum_{j \in I_u} y_j \right) \tag{2.3}$$

Non-Negative Matrix Factorization (NMF) This algorithm works similarly to **SVD**, considering the factors $q_i \in \mathbb{R}^+$ and $p_u \in \mathbb{R}^{\mathrm{f}}$, and capturing the interaction between a user and an item by a dot product. The main difference is the non-negativity constraint and the fact that **NMF** focuses on the explainability problem of latent models when dealing with implicit feedback data (views, click) [2]. However, in our scenario, we work with explicit feedback (i.e. ratings, reviews, votes), in this case, we do not have interpretability advantages of this technique [2]. Equation 2.4 presents how **NMF** predicts the item rating.

$$\hat{y}_{ui} = q_i^T p_u \tag{2.4}$$

2.1.4 Naive Models

Naive forecasting models are non-personalized algorithms that work based exclusively on history observation. In this work, we use three different naive baselines. Two baselines are based on the historical **average of the item and user**, given by Equations 2.5, 2.6 respectively. Lastly, a **random algorithm** that generates the prediction from a normal distribution was used. The Equation 2.7 presents how the ratings are predicted.

$$\hat{y} = \frac{1}{|I_u|} \sum_{i=0}^{|I_u|} r_{ui}, \forall u \in U \tag{2.5}$$

Where $|I_u|$ is the set of all users that rated an item i, and r_{ui} is the rating given by users u for an item.

$$\hat{y} = \frac{1}{|U_i|} \sum_{i=0}^{|U_i|} r_{ui}, \forall i \in I \tag{2.6}$$

Where $|U_i|$ is the set of all users that rated items i.

$$\hat{y} = \mathcal{N}(\hat{\mu}, \hat{\sigma^2}) \tag{2.7}$$

Where $\hat{\mu}$ is the average rating of the items in I and $\hat{\sigma}$ is the standard deviation of the ratings.

Recall that, neither of the naive baselines is personalized. Even though, the two baselines based on average consider the user and item history these techniques will give the same rating always.

2.1.5 Summary of Recommender Systems

In this section, we reviewed several techniques for personalized and unpersonalized recommendations. Table 2.1 lists all recommendation techniques considered in this work, showing their main characteristics. Here, we exploit a diverse set of techniques, aiming to investigate the impact of the best of them in the route recommendation scenario.

Method	**Personalized**	**Item-Item**	**User-User**	**Use Similarity**	**Category**
kNN-basic	✓	✓	✓	✓	Memory based
kNN-Mean	✓	✓	✓	✓	Memory based
kNN-Z-Score	✓	✓	✓	✓	Memory based
SVD	✓	X	X	X	Model based
SVD++	✓	X	X	X	Model based
NMF	✓	X	X	X	Model based
Item-Mean	X	X	X	X	Naive
User-Mean	X	X	X	X	Naive
Random	X	X	X	X	Naive

Table 2.1: Recommender systems techniques are considered in this work.

2.2 Optimization Heuristics

Optimization is a field of mathematics and computer science that studies methods and techniques to find optimal feasible solutions among several candidate solutions in difficult scenarios [31, 4]. Basically, techniques in this field, work by minimizing or maximizing a function (or many functions) so-called objective function, while respecting a set of constraints.

In the literature, several algorithms have been proposed such as combinatorial optimization, linear, nonlinear, and dynamic programming [31]. These algorithms fall in the category of exact approaches, i.e. they search the space of solutions until finding the best overall solution or global optimum.

It is noteworthy that exact algorithms have high computational complexity, hence not all instances of problems can be solved through exact algorithms [27]. To better illustrate that, consider Travel Salesman Problem (TSP) a classic computer science problem, which has an exact solution found in a $O(2^n)$ time complexity by a Dynamic Programming implementation [79]. Complementary, consider the Table 2.2 based on the work of [27], where the author illustrates the time behavior of algorithms with different complexity.

Time Complexity	$n = 10$	$n = 20$	$n = 30$	$n = 40$	$n = 50$	$n = 60$
n	.00001 s	.00002 s	.00003 s	.00004 s	.00005 s	.00006 s
n^2	.0001 s	.0004 s	.0009 s	.0016 s	.0025 s	.0036 s
n^3	.001 s	.008 s	.027 s	0.64 s	.125 s	.216 s
n^5	.1 s	3.2 s	24.3 s	1.7 min	5.2 min	13.0 min
2^n	.001 s	1.0 s	17.9 min	12.7 days	35.7 years	366 centuries
3^n	.0059 s	58 min	6.5 years	3855 centuries	2×10^8 centuries	1.3×10^3 centuries

Table 2.2: Execution time of algorithms with different complexity. This Table was based on the one present in [27].

Considering TSP instances where $n \geq 40$ the algorithm would take days to find an optimal solution. In any real case scenario, it would be impossible to use exact approaches given the need for information in real-time. Independent of the technique proposed to solve problems like the TSP and Knapsack Problem, it would not be possible to find an optimal solution in a timely manner given that these problems belong to the NP-Complete class [16].

In cases where the problem is demonstrably in NP-Complete or NP-Hard, the solution found was to use algorithms that can achieve not-exact-acceptable results in a reasonable amount of time, the so-called heuristics [31]. Following one of the definitions presented in the work of [64], a heuristic is a strategy that is used to improve efficiency or discover solutions to complex problems. As stated in [55], a heuristic can often be wrong but is right most of the time making it useful as a guide. Solutions in this class are designed

to provide better computational performance when compared to exact techniques, at expense of lower accuracy [31].

Recall that in this work we tackle the Tourism Trip Design Problem, which corresponds in a junction of two different problems, the TSP and the Knapsack problem[78, 34, 44], and as shown in the work of [39] this is an NP-Hard problem. Hence, in this work, we use several heuristics and meta-heuristics to solve the tourist problem. In the following sections, we present each of the heuristics used in this work to solve the TTDP.

2.2.1 Genetic Algorithm

Genetic Algorithm (GA) is a bio-inspired evolutive computation technique that is based on the mechanism of natural selection that relies on the Darwin principle of survival of the fittest [4]. This algorithm is a dynamic and powerful tool for solving search and optimization problems, which explains the technique's popularity [68]

The **GA** works by first starting a random set of candidate solutions. The set of solutions in **GA** is called population, the population is composed of chromosomes and each chromosome cell is called a gene.

First, the algorithm evaluates the population, giving each individual a score, so-called fitness, that describes how good a chromosome is. Based on this score, chromosomes are selected for recombination, hence, the best fitted have a better chance to be selected to be parents of the new population.

Next, the crossover step takes from each parent individual a portion of his genes combining them to form a new chromosome called a child. In [20, 56], the authors describe the idea behind the crossover as the simulation of the mixing of genetic material when organisms reproduce.

Lastly, the mutation step focuses on ensuring the genetic variability over the population, avoiding similar chromosomes, and local optimum [37]. To accomplish so, the mutation step works by changing random genes of the chromosomes. Thus, genes that once a value are changed to another valid value within a mutation probability.

2.2.2 Clonalg

Clonalg (Clonal Selection Theory) is an artificial immune system based on the clonal selection theory of acquired immunity [10], been proposed in [17]. This method shares similar principles with **GA**, and in the literature **clonalg** has shown some interesting results, surpassing other heuristics in scenarios of function optimization and pattern recognition [71]. Besides that, the low cost and relatively simple implementation have encouraged us to use this technique for the tourist problem.

Following the steps described in [17], to perform optimization **clonalg** works by first initializing an antibody pool A of size N, similar to the population in **GA**. This antibody pool is then partitioned in two: (i) memory antibody A_m - Antibodies that will be used in main evolutive procedures and (ii) remaining antibody A_r - Antibodies used to introduce genetic variability in the system.

From the first set, A_m, the best n antibodies are selected based on affinity measure (fitness). These antibodies are cloned forming a clone population A_c; it is noteworthy that the amount of clones is directly proportional to the affinity score. The clone population then passes through a hypermutation scheme, forming a population A_{cm}, where the mutation is inversely proportional to the affinity, hence the best antibodies are less mutated them those with a smaller affinity. The best individuals from the A_{cm} are selected to compose the remaining antibody set, been some members of A_m replaced by members of the A_{cm}. In the last step, d antibodies from the A_m are replaced by memory antibodies, introducing population diversity. This last step takes into consideration the affinity of the cells to make replacements, thus, better the fitness, small are the chances of replacement.

2.2.3 GRASP

GRASP (Greedy Randomized Adaptive Search Procedures) is a meta-heuristic for combinatorial optimization [62]. Different from **GA** and **Clonalg**, **GRASP** is not inspired by nature elements, been the algorithm principle-based a greedy randomized selection. Greedy randomized algorithms work the same way as pure greedy algorithms, with the difference that randomization is used to build different solutions along with the runs [62]. The constructed solutions are not taken into consideration for the next iteration, thus, is not possible to see an evolutive procedure [55]. Due to this fact, the work of [22] describes **GRASP** as a repetitive sampling technique.

Following the steps described in [62, 22], **GRASP** can be divided into two main

phases: (*i*) Construction and (*ii*) Local Search. In the first phase, a solution is constructed in a greedy randomized manner. By following this principle, different solutions are constructed along with the runs which guarantee a high variability on the constructed candidates [62]. If the constructed solution is not feasible, it is either repaired or discarded. When a feasible solution is obtained, the solution is stored in a Restricted Candidate List (RCL). From the RCL a solution is selected and then passed to the second phase, local search. This phase makes an iterative process over the candidate solution replacing parts of the solution for neighbor values.

2.2.4 Naive Solutions

As in the naive RS, these heuristics are simpler, being characterized for having a small computational time. As a downside, naive methods do not perform well for all instances. In this work, we use two naive heuristics: a (*i*) **Random** and (*ii*) **Greedy** approach.

The **random approach** works by stochastic selecting values to compose the solution. As we use stochastic approaches that work with computational intelligence behind them, it is expected that **GA, Clonalg, and GRASP** have a better performance than a completely random method.

The **greedy approach** works by sorting the values and selecting those not present in the solution (avoiding repetition) that better optimize the fitness function.

2.2.5 Summary of Heuristics

In this section, we reviewed several heuristics used for function optimization. Table 2.3 list all heuristics considered in this work, showing their main characteristics. Here, we exploit a diverse set of heuristics and meta-heuristics, aiming to investigate those which can generate the best routes in different scenarios.

Method	**Bio-Inspired**	**Evolutive**	**Local Search**	**Category**
Genetic Algorithm	✓	✓	X	Bio Inspired
Clonalg	✓	✓	X	Bio Inspired
GRASP	X	X	✓	Meta-heuristic
Greedy	X	X	X	Naive Model
Random	X	X	X	Naive Model

Table 2.3: Heuristics are considered in this work.

Chapter 3

Related Work

In this chapter, we discuss related studies to our work dividing into three main topics: datasets used in tourism-related work (Section 3.1); POI Recommender Systems (Section 3.2); and Tourism Trip Design Problem (Section 3.3), where we present works similar to in focus with our proposal. Finally, in Section 3.4 we present a summary of the literature, distinguishing our contribution in a broad view.

3.1 Datasets

The act of travel is part of human history and became much easier in the last years due to the increase in mobility and information access [38]. With the popularization of the web, people were encouraged to search about new destinies that they could visit. Besides, users were now able to share their experiences with other people through social networks and traveling specific platforms [86]. The popularization of these platforms made studying tourism a simpler task given the amount of data available [46, 3].

There are currently available in the literature datasets extracted from Location-Based Social Networks (LBSNs) as Gowalla, Foursquare and Yelp, and Travel Social Networks (TSNs) as TripAdvisor, Real Travel, and Travello, which are mainly used for the task recommendation and route generation.

LBSNs are social platforms where users share in real-time their geospatial location and visit timestamp, composing a visit check-in [5]. The users of these platforms show a pattern of use on a regular daily basis, and about 90% of their transitions are between places are within the 50 km range [51]. Corroborating with that, the work of [35] shows that most of the locations are followed by fewer than 5 locations consecutively, showing that users have a similar visitation pattern.

Considering LBSNs datasets available in the literature, two different sets of features

can be found. Datasets from Foursquare and Gowalla present a small set of features, composed of the User identifier, POI identifier, POI category, and timestamp. In this case, works that employ Foursquare and Gowalla data assume that if a user visited a place, then, the user has liked, due to the fact that no explicit feedback is available [44, 88, 87]. On the other hand, Yelp data is more suited for recommendation, given that besides having the check-in data, it enables the users to share an explicit evaluation for each POI. However, in the case of route recommendation (TTDP) it lacks some vital information to produce and evaluate routes in the real world, as the places price.

Besides LBSNs, TSN data is also widely used in tourism evaluation [40, 78]. TSNs are social platforms focused on tourism where users can share places visited and reviews with other travelers. Different from LBSNs, TSNs have a focus on a "couch review", i.e. the travelers make their review after visiting a place and in many cases months after that visit. In this scenario, the check-in time is not available, only the month and year of visit. However, the information presented in a TSN review is more detailed, given that the user supplies explicit ratings to aspects such as place service, cleanness, and location, also providing a written review. The main advantage of TSN datasets over Yelp data is that places owners also share their profile, defining features such as price, and opening hours, having a closer relationship with the travelers.

Besides the use of TSNs/LBSNs data, it is also common in the literature the use of different datasets in tourism-related works. In some cases, external datasets are used to increase the amount of data and enhance the quality of data, making it more suited for real scenarios [40]. Below, we make a small summary of other data sources applied in the literature.

- **General Purpose Social Networks:** Social media platforms that not focus only on tourism or tourism-related data, but also on other subjects [58]. Examples: Twitter, Flick, Instagram.
- **Human Mobility Platforms:** Platform that works as digital maps or guides for users in the city [28, 9, 12, 78, 24, 85]. Examples: GoogleMaps, OpenStreet Maps.
- **Synthetic:** Randomly generated data [53, 49, 93, 33].
- **Open-source encyclopedia:** Collaborative encyclopedias, where any person can edit the information available [9, 85]. Example: Wikipedia.
- **Governmental Data:** Data gathered and made available by governmental platforms [74, 29, 49, 93, 58].
- **Blogs and Travelogues:** Social diaries that are available to any reader [36]. Example: Travel Triangle.

With exception of human mobility platforms, like Google and OpenStreet maps, that are used to complement the data by supplying information about the real distance between POIs [28, 9, 12, 78, 24, 85], all other data sources presents disadvantages in comparison with LBSNs and TSNs datasets.

Even though data from General-purpose SNs, open-source encyclopedias, blogs, and travelogues present a vast amount of textual [9, 36, 85] and image data [9, 85, 14, 65] that are used to extract features of places and users, the treatment of the data is usually very costly, due to the fact that these are usually in an unstructured format. Besides that, explicit feedback in the format of ratings is for the most part never present, which makes it harder to extract and measure the opinion of a user towards a POI.

In conclusion, TSN and LBSN datasets are usually more suited for tourism scenarios, due to the fact that these data have been extensively evaluated in several works [51, 86, 40, 44], and are less costly to treat due to their semi-structured format. If we compare TSN and LBSN directly, each of them will be suited for different tasks. For example, TSN data cannot be used on the task of the Next POI recommendation due to the granularity of timestamp data. However, TSN data usually present more attributes from POIs given that these are filled by the place owners. These attributes are essential when testing applications in real-world scenarios.

3.2 POI Recommendation

POI recommendation is the task of suggesting new places for users to visit based on their history. Since the use of LBSN has popularized, the interest of academics and industry has increased further [92, 51]. As stated in [5], "location data bridges the gap between the physical and digital world and enables a deeper understanding of users' preferences and behavior". The same statements work for other tourism-related data. In the tourism context, several papers have developed solutions that aim to recommend places, been called e-service or e-tourism. As pointed out in [6], tourism is a field that depends on the personal interests and preferences of people. Thus, recommender systems assist tourism by giving the opportunity to offer a better and personalized service to customers.

Different from conventional recommendation scenarios (e.g. movie recommendation), POI recommendation is considered to be a harder task, due to geographical (e.g. physical constraints between the POIs) [13], social (e.g. friends can influence on the user visit) [92, 5] and temporal (e.g. places are more searched on summer) influences [51]. Be-

sides, the number of instances tends to be much smaller, sparser, and noisier than in book and movie scenarios [6, 84]. Also, the heterogeneity of information in social networks describes the user activity from a variety of perspectives and through different types of data (e.g. photos, text, check-ins), which lead to a vast literature that focuses on different approaches for accomplishing the POI recommendation task [15, 90, 21].

First works on POI recommendation system, focused on recommending a top k list of locations to be chosen by the user [15, 48, 82, 90]. Due to the many aspects that make the POI recommendation scenario harder, works propose methodologies based on the use of additional information. As stated in [76], additional data allows for more accurate recommendations than traditional methods, thus, works exploit geo-coordenates [15, 48], social ties [82, 90], POI category [89, 47] and timestamp [90] to enhance predictions quality.

The works of [15, 48] consider the geographical influence in their recommendation. The geographical influence is crucial in POI RS due to physical constraints [13]. By considering such influence, algorithms avoid recommending places too far from the main location, due to the fact that users tend to check in around several centers (e.g. home and workplace) [15]. Corroborating with that, the work of [48] verify that users tend to visit new places near locations already visited, leveraging their model by the use of geographical influence on different regions.

As previously stated, one important aspect that can influence the POIs visited by the user is their social ties, given that friends in LBSN share more common interests than non-friends [92]. Even though some authors consider the social influence limited [15], the work of [76] considers users connections as important as geographical constraints. Social influence has been treated in different ways in the literature. The work of [76], known as **LFBCA**, models a social graph where both user preference and social influence are modeled by distinct edges [92]. In [82, 90] the social influence is modeled by using the user history similarity (based on check-ins) and friends based similarity (based on direct links) and using both of them in a CF system called Friend-based Collaborative Filtering (FCF).

The work of [89], known as **GeoSoca**, joins geographical, users, and categorical influence. In their work, authors state that the category of a POI reflects its business activities, nevertheless, this information can be leveraged to discover patterns of preferential categories in the user history. As stated in [47], the categorical information has an influence due to users' specific hobbies. However, in some cases, even though, categorical influence enhances prediction accuracy, it creates a bubble for the user, where only a few categories will be recommended.

The work of [90] considers that another aspect of human mobility that can assist in the recommendation is the sequential patterns presented in human movement. Sequential influences are not fully explored in the single POI recommendation scenario, however is

a solid ground for the scenario of Next POI recommendation (NPR).

NPR consists of the task of recommending a POI for a user based on his current location [84]. The task of NPR is considered to be harder than traditional POI recommendation given that sequential check-in interactions are even sparser, thus, the amount of data is more scarce [84, 91]. The work of [91] states that the geographical distance between POIs is what makes NPR a different task than other next recommendation problems. The work of [84] considers the temporal aspect fundamental on NPR task, given that POIs have different popularity in different time slots (e.g., a club is more visited in night periods). The work of [35] focused on identifying similar locations and using regular transitions to identify transitional patterns, hence, this work directly tackles the sparsity problem.

If we consider all recommendations within a period of time it is possible to identify routes, which directly relate NPR to the task investigated in this work. Nevertheless, the task of NPR considers as input the current time and place, and does not take into consideration several constraints that are considered in routing problems as (*i*) distance time between places, (*ii*) transportation type, (*iii*) budget and (*iv*) place working hours (i.e. open and end hours). Nevertheless, we believe that with adaptations it would be possible to use results of the Next POI recommendation algorithms in automated route planning techniques.

Another branch of POI recommendation is the scenario of in/out-of-town POI recommendation [83, 61]. Recall that several studies have shown that users exhibit a pattern of visiting nearby POIs and use geographic information as leverage for their recommendation [15, 48]. Nevertheless, when visiting places outside users' usual locations these algorithms cannot perform so well, making the task of recommending in this scenario even harder due to sparsity [21]. Nevertheless, some efforts in the literature focus on good recommendations in scenarios when users are in their hometown and when they are out of their town.

In [61], authors state that in scenarios of out-of-town recommendation the task of POI recommendation is more important, given the little knowledge that a user has on the new town. The authors also argue that in this scenario recommending sets of nearby locations can be beneficial for users, given their reduced schedule when traveling. With a different perspective, the work of [81] argues that the main problem in out-of-town recommendation is the cold-start problem, given that the interest of users drifts in different cities and the travel intention can also affect their visits.

In this work, we test 5 different literature proposals to our task of recommendation. It was considered works that deal with different influences in their evaluations. This work was considered **USG** [82] and **MGMPFM** [15] which considers geographical influence, **GeoSoca** [89] which considers geographical and categorical influences, **LFBCA** that considers social influence [76], and **LORE** [90] that considers geographical, social, and

temporal influences. It is noteworthy that these works originally use LBSN datasets in their evaluation, thus, some adaptations were made so it can deal with TripAdvisor data. This process is detailed in Chapter 5.

3.3 Route Recommendation

In the tourism context, several works tackle the Tourist Trip Design Problem (TTDP), which aims to design the tour that maximizes the user profit, while respecting user and place constraints. As stated in [78], this problem is a combination of the traveling salesman and the knapsack problem, resulting in the so-called Orienteering Problem (OP) [30]. The first work to model the OP in the tourism scenario was [75]. Given the high applicability of this problem on the tourism scenario, the authors named the OP in the tourism context as Tourism Trip Design Problem (TTDP).

The OP can be formulated as follow, let $G = (V, E)$ be a graph where each vertex $v \in V$ has associated a weight $w_v \in \mathbb{R}_+$ of profits, been $W = \{w_0, w_1, ..., w_v\}$. Given a starting node s, a terminal node f, where $s, f \in V$, and a positive time budget B, the goal is to find a path from s to f (in our tour $s = f$), with length at most B such that the total profit of the visited nodes is maximized.

As we can see, this problem can be easily mapped in the tourist problem if we consider the city as a graph the nodes in V are the places that can be visited, the edges in E are the connections between places, w is the profit gained in each visited place and B would be the amount of time available to spend in the city. In this scenario, we consider as a basic restriction the time constraint. In the literature the OP modeling has been re-visited through many perspectives, adding more constraints [74, 28, 1].

Place Time The modeling considered in OP does not deal with one important aspect when traveling which is the time windows (i.e. opening and closing time of places). Hence, OP modeling considers that all POIs are available on a 24 x 7 basis [28]. However, in realistic scenarios where each place is associated with working hours, the OP fails to properly model these characteristics. Thus, new modeling called Orienteering Problem with Time Windows (OPTW) was proposed considering an additional constraint where each place has its own time window. Hence, in the OPTW besides having to maximize the path (or route) in a time budget B, each node is associated with a time window where the place can be accessed [30, 26, 73].

Time Budget/Daily Distance/Max. Daily Places In different works, the constraint B, usually considered as the time budget, can also be replaced by different restrictions as daily distance [44, 58, 65, 32], or maximum daily places to be visited [65, 32, 85]. In all cases, these constraints aim to limit the number of places that will be visited in one day.

Multiple Transport Types The consideration of the places' working hours (time windows) in the TTDP lead the works [88, 87] to also consider the uncertainty of traveling time due to traffic conditions. When traveling many different options are available for moving from one place to another, such as walking, taxi, metro, and transport apps [32]. Each of these options would have a different time and money cost. As stated in [88], the traveling time can impact deeply on the route schedule given that POI has working hours. The works of [32, 9] tackle this problem by enabling the user to choose between multiple transportation types. Nevertheless, the authors do not consider several important aspects when treating multiple transportation types. First, this work does not consider the time spent moving from a place to the bus stop or train station (when using this transportation type). Second, is not considered that can be a waiting time until the public transportation passes. Aiming to tackle this problem, the work of [28] uses different modeling for OP, so-called Time-Dependent OP (TDOP) which was introduced by [23]. In the TDOP another variable was also considered in the OP which considers the time-dependence when moving along nodes [30]. This time-dependence is taken into consideration when selecting public transportation. Lastly, the work of [12] gives the user option to select between walking or car in their travels. To avoid traffic-related modeling and problems, the work of [24] only considers on-foot-tours in their work.

Climate Data One important factor that can directly impact the traffic and user experience is the weather [28]. In [40], authors show the importance of climate features on the forecast of visits in a place. Especially when considering outdoor (i.e. open door) places, the weather can spoil the user experience. Avoiding to recommend outdoor places in bad weather scenarios, the work of [28, 29] makes use of climate data. In this scenario, the author does not take into consideration specific temperatures, but the general character of the day, as rainy or hot.

Rec. Sys. Personalization Another important aspect of tour recommendation is personalization. Through the use of personal data, it is possible to maximize the user experience, selecting only places that are similar to the user's historic taste [74]. In the tourist problem, several types of personalization were considered and can lead to a discussion on what is considered to be personalized. For example, the works of [49, 93], consider user constraints as a type of personalization given that the spatio-temporal route

structure will be directly linked to when and where the route will start. Going further in personalization the work of [70] made personalized routes through the use of large questionnaires, enabling to make a descriptive profile of what users liked and what would be recommended. Similarly, the work of [69] used semantic matching to evaluate the similarity of a user profile with other profiles evaluated. The works of [26, 67] focused on constructing routes without the need for excessive data input by the user. To make it personalized, the authors enabled users to edit these routes through adding and removal of POIs in the route. More recently, the work of [32] focus on creating routes based on ordered places that the user indicated. In [74], the authors personalize their routes by using users' context, interest, and keywords inputted by the user. Even though these operations could lead to a more suitable route for the user, the number of manual operations needed was too excessive.

One of the advantages of using social media data is the fact that these platforms concentrate a lot of user data, requiring near minimum user input. Taking advantage of this fact, the work of [53] made use of data from LBSN in a Collaborative Filtering RS to identify users' preferences and posteriorly recommend routes. This methodology is called "Filter first, tour second" [44]. Following this methodology, the works of [88, 87, 44, 14] made use of RS with heuristics to make personalized tour recommendations. With exception of [14], all other works used CF as the main technique in RS. In [14], authors used a hybrid RS based on deep learning. In their methodology POI textual and category information is taken into account in the RS.

In another direction, the work of [9, 65] recommends personalized routes by considering the similarity of places previously visited with new locations.

Money Cost Even though data extracted from LBSNs can enhance personalization in routes, as stated before, one feature that is not present in this type of dataset is the price of activities. When considering realistic tour recommendations, the cost can be essential to define if a user will visit a place or not. The work [53, 44] introduces money constraints in their model. Nevertheless, in their evaluation random values are used as price per category, thus, the datasets used in these works are not suited for evaluation in real scenarios. The work of [65] only considers the travel cost given multiple transportation modes, instead of considering the cost of visiting a place. To the best of our knowledge, Planet Caravan is the only work to consider real data to analyze the validity of the budget constraint.

Single/Multiple Days One factor that is especially impacted by the money budget is the number of days that a user will be traveling. The work of [24] asserts that most trips range from 3-8 days, which makes most of the works in the literature unsuitable for real scenarios, given that only a few of them perform multiple-day tour recommendations [74,

28, 12, 44, 24]. Multiple day tours can be considered a more difficult task given that opening the possibility for categorical diversification, hotel selection, and unexpected events may make the planning infeasible [74].

Category/Diversity Diversification in tours is a common matter not only on works that perform recommendations on a multiple-day basis. Diversification focus on recommending distinct places on the route, avoiding recommending bundles of venues of the same category. Diversification has been treated in three different ways in the literature. One simple approach is to avoid visiting repeated places on the same day, which we consider to be done by all proposals. Another way is to consider only n POIs from the same category can be visited in a day, where $n > 0$ [44, 53]. In this case, the diversification is left to user choice and enables to make category-specific tours within the tours, for example, historic and gastronomic tours. A more sophisticated way is to consider user history to select the categories most visited and then select places that are within the users past categories [58]. In this case, the recommendations made by the algorithm can be very similar, leading to the problem of recommendation bubble [57].

Hotel Selection The work of [24] states that multiple days proposals can lead to unbalanced tours where the places with the highest ratings are concentrated in the first days. Thus, this work aims to produce multiple day tours where the satisfaction of tourists is maximized each day. The main limitation of this work is that the authors consider that each day the user will start the tour at a point in the center of a target city. Considering realistic scenarios this only would be possible if the user selects a hotel in the city center, which cannot happen always given the cost of hotels. Besides that, hotel selection can be a tiresome task for tourists that are not familiar with their destination area, and this scenario is worsened in long trips [30], where the hotel selection is essential to good travel. To solve this case, a new variant of the OP was introduced in [18], the so-called OP with Hotel Selection (OPHS) or Orienteering Problem with Intermediate Facilities (OPIF). In this case, the goal is to determine a fixed number of connected places that maximize the sum of collected profits. In these tours, the first and the last places of each day is the selected hotel [30]. In this scenario, the hotels' utility is not considered. In [28, 44], authors state that the hotel selection is essential for the recommendation of good routes, due to the fact that a poor hotel selection can lead to high travel time between the hotel and POIs.

Mandatory Places Lastly, when visiting a new place, there are some places that the tourist can consider a must-visit, for example, the Cristo Redentor in Rio de Janeiro. In this case, the tours generated can or cannot pass through the Cristo Redentor. In cases where the tourist wants to visit a place, but the route does consider that place, the

user experience is not fully achieved. To tackle this problem, the work of [44] introduces the concept of mandatory places. Mandatory places are points pre-selected by the user that must be visited. Thus, the work of [44] focus on generating a tour that contains all mandatory points and maximizes the score of the places visited.

3.4 Summary of Related Works

In this section, we reviewed previous studies on POI and route recommendations that provide motivation for research goals in this work. In comparison to prior work, we perform personalized tour recommendations with hotel selection for multiple days, considering aspects such as climate, time windows, money budget, and multiple transportation. Besides that, we also propose the use of a new constraint called appointments. By using the appointments constraint we enable users to build a trip around their schedule. For example, consider a researcher in a conference in a new town that wants to watch only a few presentations and in their free time wants to visit other places in the city. In this case, the person inputs their appointment, and the trip that maximizes the researcher schedule is assembled. Moreover, in our work, we use data from TripAdvisor which contains the real price of hotels, restaurants, and attractions. To the best of our knowledge this is the first work to make an evaluation using real cost data.

The Tables 3.1, 3.2 and 3.3 summarize the previous studies showing their constraints, datasets, and algorithms used. In the last row of the first two tables, we show how our work, and how it compares itself to the literature.

Paper	Single Day	Time Budget	Place Time	Category	User Time	Multiple Days	Daily Distance	Diversity	Restaurant Time	Mult. Transport Types	Money Cost	Max. daily places	Hotel	Mandatory Places	Appointment	Rec. Sys	Type of Rec. Sys
[44]	✓	-	✓	✓	✓	✓	✓	-	-	-	✓	-	✓	✓	-	✓	CF
[53]	✓	✓	-	✓	-	-	-	✓	-	-	✓	-	-	-	-	✓	CF
[45]	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[49]	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[29]	✓	-	✓	-	✓	-	-	-	-	-	-	-	-	-	-	-	-
[9]	✓	✓	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-
[33]	✓	✓	-	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-
[58]	✓	✓	-	✓	-	-	✓	-	-	-	-	-	-	-	-	-	-
[14]	✓	✓	-	-	-	-	-	✓	-	-	-	-	-	-	-	✓	Hybrid
[78]	✓	✓	✓	✓	-	-	-	-	✓	-	-	-	-	-	-	-	-
[88]	✓	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	CF
[65]	✓	-	✓	✓	-	-	✓	-	-	-	✓	✓	-	-	-	-	-
[32]	✓	✓	-	✓	-	-	✓	-	-	✓	-	✓	-	-	-	-	-
[12]	✓	-	✓	-	✓	✓	-	-	✓	✓	-	-	-	-	-	-	-
[85]	✓	✓	-	-	-	-	-	✓	-	-	-	✓	-	✓	-	-	-
[87]	✓	✓	✓	-	-	-	-	✓	-	-	-	-	-	-	-	✓	CF
[36]	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
[28]	-	-	✓	-	✓	✓	-	-	✓	✓	-	-	✓	-	-	-	-
[74]	-	✓	✓	-	✓	✓	✓	-	✓	-	-	-	-	-	-	-	-
[24]	-	✓	✓	-	-	✓	-	-	-	-	-	-	-	-	-	-	-
Planet Caravan (Our Work)	-	-	✓	✓	✓	✓	✓	-	-	-	✓	✓	✓	✓	✓	✓	CF

Table 3.1: Constraints literature review.

Paper	**Dataset**	**Dataset Type**	**Algorithm/Heuristic**
[44]	Foursquare	LBSN	GRASP
[53]	Gowalla, Synthetic	LBSN	DBScan
[45]	GoogleMaps	Maps	GA
[49]	Governmental, Synthetic	Governmental	GA, DEA
[29]	Governmental	Governmental	ILS
[9]	Flickr, Wikipedia, Google Maps	SN	ILS
[33]	Foursquare, Synthetic	LBSN	Greedy
[58]	Twitter, Governmental	SN	Vader
[93]	Governmental, Synthetic	Governmental	ACO, DEA
[14]	Flickr	SN	ILS
[78]	TripAdvisor, Google Maps	TSN	GA
[88]	Yelp, Foursquare	LBSN	Greedy, DP, Heuristic, DFS
[65]	Flickr	SN	NSGA-II
[32]	Foursquare	LBSN	DP, Greedy
[12]	Minube, Google Maps	TSN	K-Means, HC, WBFS
[85]	Flickr, Wikipedia, TripAdvisor, GoogleMaps	SN	GA
[87]	Yelp, Foursquare	LBSN	Greedy, DP, Heuristic, DFS
[36]	Travelogues	Blogs and Travelogues	Static Heuristic
[28]	TripAdvisor, Google Maps	TSN	ILS
[74]	Governmental	Governmental	GRASP
[24]	GoogleMaps, Foursquare	Maps	Greedy, MaxMin
Planet Caravan (Our Work)	TripAdvisor, OpenStreet Maps, Blogs	TSN	GA, Clonalg, GRASP, Greedy, Random

Table 3.2: Datasets and algorithms used in the literature.

Initials	**Algorithm**
DFS	Depth First Search
DP	Dynamic Programming
ACO	Ant Colony Optimization
DEA	Different Evolutionary Algorithm
GA	Genetic Algorithm
HC	Hill Climbing
WBFS	Weighted Best-First Search
ILS	Iterated Local Search

Table 3.3: Algorithms used in the literature.

Chapter 4

Route Generation Framework

This chapter describes the solution we present for the Orienteering Problem. Note that in this work we focus on constructing personalized routes and modeling real-world constraints. To accomplish so we divide our problem into two main tasks: personalization through a matrix completion process, and route construction through a linear mixed-integer programming model for the OP. We base our modeling on the work of [44]. In summary, a linear mixed-integer model is an optimization problem in which a set of integer and continuous variables co-exist. The constraints are all linear equations and inequalities, and the objective is a function to be minimized/maximized [80].

POI Recommendation/Matrix Completion In our first task, consider a city with n places, and a tourist which is visiting such city. This user is represented by a vector of ratings $R_i \in \mathbb{R}_+$, which contains a rating for each place i previously visited by this user. Considering that the user visited k of the n places in the city, the goal of this task is to fill R such that a personalized (based on previous experience) rating (utility) to the $n-k$ non-visited places are given.

Route Generation Assume now that this tourist (user) is traveling this city for p days with the aim of visiting a maximum number m of POIs every day. This user aim is to generate daily tours dividing the places in the city into three sets: Mandatory set $\mathcal{M}$ that consists of the set of must-visit POIs, Appointments $\mathcal{A}$ that consists of the set of must-visit POIs with an associated schedule, and an Optional set $\mathcal{O}$ that consists on the set of POIs that the user can optionally visit during the p days. Note that while the places in the sets $\mathcal{M}$ and $\mathcal{A}$ must be visited at least once, not all places in $\mathcal{O}$ may be visited.

Considering that in the matrix completion step each POI i is associated with a rating (utility) R_i that defines the user's personal preferences, being this value defined by a recommender system technique, the **tourist goal is to find a subset of $\mathcal{M} \bigcup \mathcal{A} \bigcup \mathcal{O}$ which maximizes the utility of places visited**. Considering that a variable X is a binary matrix that defines whether a place j was visited after i in a day d, the **objective function** of this problem is expressed by the equation below:

$$\max \sum_{d=0}^{p} \sum_{i=0}^{n} \sum_{j=0}^{n} X_{i,j}^{d} R_i, \tag{4.1}$$

Let $G = (V, E)$ be a directed-complete graph representing the city. Let $V = \{v_0, v_1, ..., v_n\}$ represent the set of vertices which correspond to places available to be visited, where each place $v \in V$ is associated with a **utility** R_v, a **category** $\mathcal{C}_v$, a **cost** $C_v \in \mathbb{R}_+$, a **visiting time** s_v^d for each day d, and a **opening** O_v^d and **closing** time H_v^d. The first h nodes of this graph are hotels, and n is the number of nodes in G. Additionally, $\mathcal{V}$ defines the set of all places in the route, and $\mathcal{V}_c$ is the set of places excluding starting and ending points. Each place $v \in V_c$ can be either a mandatory, an appointment, or an optional place. Each edge connecting the POIs has its corresponding weight $D = \{D_{i,j}, i \neq j\} | D_{i,j} \to \mathbb{R}_+$, that denotes the distance between two places. Our aim can be adapted in this graph to **find a cycle (e.g. the path that starts and ends at the same node) in G that maximizes the utility R while respecting the sets of real-world constraints**. Is noteworthy that to each place attribute defined above a correspondent user constraint value is defined. Below we present all these constraints and the users' inputs.

Recall that we defined the **cost** (e.g. entrance fees, food price) of visiting each place i as C_i. In this scenario, users define the total budget $b \in \mathbb{R}_+$ that they are willing to spend on all the travel. The **cost constraint** is defined by the Equation 4.2:

$$\sum_{d=0}^{p} \sum_{i=0}^{n} \sum_{j=0}^{n} X_{i,j}^{d} C_i \leq b \tag{4.2}$$

Users also define the amount $w \in \mathbb{R}_+$ that is willing to cover per day. The Equation 4.4, presents the **daily distance constraint**:

$$\sum_{i=0}^{n} \sum_{j=0}^{n} X_{i,j}^{d} D_{i,j} \leq w, \ \forall d \in \{1, 2, \ldots, p\} \tag{4.3}$$

Besides being daily limited by the maximum distance w, the users route is also daily limited by the maximum amount of places $m \in \mathbb{Z}_+ | m > 0$, that can be visited. The **daily activities constraint** is presented in the Equation 4.4:

$$\sum_{i=0}^{n} \sum_{j=0}^{n} X_{i,j}^{d} \leq m + 1, \ \forall d \in \{1, 2, \ldots, p\} \tag{4.4}$$

The Equation 4.5, ensures the solution **connectivity** and handle symmetrical solutions.

$$X_{i,j}^{d} + X_{j,i}^{d} \leq 1, \ \forall d \in \{1, 2, \ldots, p\}, \ \forall i \in \mathcal{V}_c, \ \forall j \in \mathcal{V}_c \tag{4.5}$$

Further, the Equation 4.6 defines the **multiple visit constraint**, avoiding re-visits in the same day:

$$\sum_{i=0}^{n} X_{i,k}^{d} = \sum_{j=1}^{n} X_{k,j}^{d} \leq 1, \ \forall d \in \{1, 2, \ldots, p\}, \ \forall k \in \mathcal{V} \tag{4.6}$$

Recall that in our modeling each place is also associated with a **category**, such as hotels, churches, restaurants, and museums. Let $\mathcal{C}$ denote the set of categories $\mathcal{C} = \{0, 1, ..., n_c\}$, where the category $\mathcal{C}_0$ defines a hotel. Let a binary matrix $K_{i,c}$ denote whether a POI i belongs to the category $c \in \mathcal{C}$. In this scenario, we define two category-based constraints. First, a value $q^d \in \mathbb{Z}_+ | q^d > 0$, defined by the user restricts the number of places from the same category that can be visited in one day. The **category constraint** is presented in the Equation 4.7:

$$\sum_{i=0}^{n} \sum_{j=0}^{n} K_{i,c} X_{i,j}^{d} \leq q^d, \ \forall d \in \{1, 2, \ldots, p\}, \ \forall c \in \mathcal{C}n\{0\} \tag{4.7}$$

Following, a **hotel constraint** defines that only the first and last place visited in a day can be a hotel, given by the Equation 4.8:

$$\sum_{i=h}^{n} \sum_{j=h}^{n} K_{i,c} X_{i,j}^{d} = 0, \ \forall d \in \{1, 2, \ldots, p\}, \ c \in \mathcal{C}, \ c = 0 \tag{4.8}$$

Recall that we defined a set $\mathcal{M}$ of mandatory places to be visited. Places in this set, must be visited at least one time during all traveling days this POIs must be allocated. Thus, the Equation 4.9 presents the **mandatory constraint**:

$$\sum_{d=0}^{p} \sum_{i=0}^{n} X_{i,k}^{d} = \sum_{d=0}^{p} \sum_{j=1}^{n} X_{k,j}^{d} \geq 1, \ \forall k \in \mathcal{M} \tag{4.9}$$

Next, we present all time-related constraints. A place time interval is composed of an **opening time** O_i^d and a **closing time** H_i^d where the user can visit the POI. Each place has also a s_i^d that defines the **time spent** in a place i in a day d. For the user, the variable a_i^d defines the arrival time on the place i on a day d. Whenever a user traverses the arc $(i, j) \in E$ a time cost $D_{i,j}$ incurs on the user time. Even though this time cost is independent of the day, it is dependent on the user transportation type (e.g. on foot, by car).

First, we define the auxiliary Equation 4.10 which defines the time of activity end for a place i:

$$f(i, j, d) = X_{i,j}^{d}(a_i^d + s_i^d + D_{i,j}) \tag{4.10}$$

Following, the Equation 4.11 presents the **overlapping time constraint** that verifies if there is no overlapping between the place i activity end and place j activity start:

$$f(i,j,d) \leq a_j^d,\ \forall d \in \{1,2,\ldots,p\},\ \forall i \in \mathcal{V}_c,\ \forall j \in \mathcal{V}_c \tag{4.11}$$

The Equation 4.12 defines the **place open constraint** which avoids visits outside the place working hours by verifying if a place is open during the visit time:

$$O_i^d \leq X_{i,j}^d(a_i^d + s_i^d) \leq H_i^d,\ \forall d \in \{1,2,\ldots,p\},\ \forall i \in \mathcal{V}_c,\ \forall j \in \mathcal{V}_c \tag{4.12}$$

The **user activity hours constraint** defined in the Equation 4.13, consider users activities time box to define wheter visit a place or not:

$$u_s^d \leq f(i,j,d) \leq u_e^d,\ \forall d \in \{1,2,\ldots,p\},\ \forall i \in \mathcal{V},\ \forall j \in \mathcal{V}, \tag{4.13}$$

where, u_s^d is the users' activity start time and e u_e^d end time, defining the user hour to leave the hotel and the maximum hour that the user must be at the hotel.

Lastly, we present the constraint related to the set of appointments $\mathcal{A}$. The places in the set $\mathcal{A}$ are associated with a start hour S_k^d and end hour E_k^d of the appointment, defining the time that the user must be at the place and the time that must leave, where $k \in \mathcal{A}$ is an appointment to be visited in the day d. In this case, the user can arrive earlier for the appointment but can not be late. Similarly, the tourist can leave the place after the end of the appointment, but can not leave earlier.

The Equation 4.14 defines the **appointments constraint**, which defines that the places in $\mathcal{A}$ must be visited at least once.

$$\sum_{i=0}^{n} X_{i,k}^d = \sum_{j=1}^{n} X_{k,j}^d \geq 1,\ \forall d \in \{1,2,\ldots,p\},\ \forall k \in \mathcal{A} \tag{4.14}$$

Complementary the Equation 4.15 defines the **appointments hours constraint**, which associates each place in the appointment set to its correspondent time interval.

$$S_k^d \leq X_{i,k}^d(a_k^d + s_k^d) \leq E_k^d,\ \forall d \in \{1,2,\ldots,p\},\ \forall k \in \mathcal{A} \tag{4.15}$$

Figure 4.1 illustrates how the routes are represented with a small example. Considering a city with 6 different places, where the places 0 and 1 are hotels, and the remaining POIs belong to other categories, the figure shows a sample route for a 2-days trip, presenting its initial representation and its representation in the X matrix.

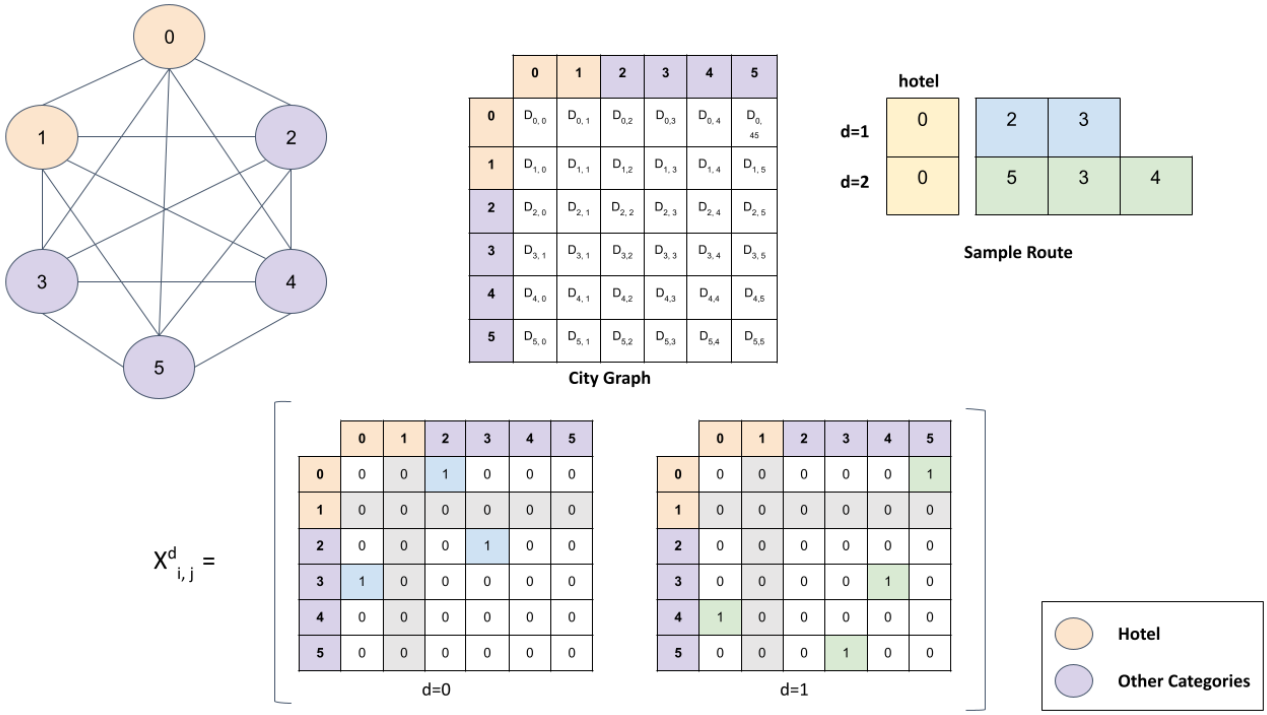


Figure 4.1: Example of route represented in the mathematical modeling.

Chapter 5

Experimental Methodology

In this chapter, we present the methodology proposed to tackle the Orienteering Problem. We organize this chapter into four main sessions followed by a summary section. First, in Section 5.1, we described datasets. Section 5.2 introduces the architecture: Personalize First - Tour Second where we show the implementation and parameterization of the techniques used. In Section 5.3 we present implementation details of the heuristics tested in our work. Lastly, in Section 5.4 we present the experimental evaluation, defining how was analyzed the effectiveness of the proposed methodology.

5.1 Data Collection

Recall that, most works in POI and route recommendation use LBSN data in their evaluations given the availability of these datasets in the literature [44, 53, 33]. Nevertheless, as discussed before, LBSN data do not present some characteristics that we consider essential to a real-world evaluation in scenarios of route recommendation as POI working hours, availability, and cost. In this case, data from TSN, as TripAdvisor is more suited for such analyzes. Besides containing more information, TripAdvisor is currently the most popular travel website [40], having about 390 million monthly unique visitors. In this section, we present our datasets which contain all TripAdvisor's data available from 5 different cities: **Tiradentes, Ouro Preto, San Gimignano, Cannes, and Ibiza**, enriched with cost data, map, and environmental data.

	Info	Tiradentes	Ouro Preto	San Giminignano	Cannes	Ibiza
Dates	Start Date	01/12/2007	01/08/2005	01/08/2003	01/03/2004	01/07/2005
	End Date	05/05/2020	10/05/2020	04/05/2020	11/05/2020	14/05/2020
Review Info	# of Restaurants	29541	25254	44235	22155	201841
	# of Attractions	13518	27513	18302	18560	40554
	# of Hotels	13299	14705	27936	30357	54051
	# of Vacation Rentals	46	73	1217	2676	1489
POI Info	# of Restaurants	147	295	105	47	1510
	# of Attractions	36	110	70	54	145
	# of Hotels	173	176	165	162	5
	# of Vacation Rentals	6	9	172	431	226
Stats.	# of users	24701	25753	64874	51007	122986
	Sparsity	99.3%	99.5%	99.6%	99.7%	99.8%
	City Area[1]	$83KM^2$	$1245KM^2$	$138KM^2$	$19.62KM^2$	$571.6KM^2$

Table 5.1: Basic description of the TripAdvisor dataset for each city.

5.1.1 Datasets Description

Our dataset collection contains **all the data available on TripAdvisor** from 5 different cities around the world: Tiradentes, Ouro Preto, San Gimignano, Cannes, and Ibiza. We choose these cities because we wanted all the data available for entire cities. We point out that we tried to collect larger cities like New York and Paris, nevertheless, the collecting time was prohibitive. From the datasets we collected, the POIs of each city can be divided into 4 categories: restaurants, attractions, hotels, and vacation rentals.

To extract this data, we developed a crawler responsible for automatically browsing the TripAdvisor website collecting all content available to users and POIs. The data collected was firstly in an unstructured format (e.g. HTML). A parser was developed to retrieve the content within each page, pre-process, and store it in a semi-structured format (e.g. CSV) data. From the data scrapped from TripAdvisor we mostly use POI data (e.g., working hours, location, category, and price) and user historic data (e.g., previous ratings).

Table 5.1 shows the basic information of the data extracted for each city. In the **dates** rows, it is possible to see the first and last entrance dates collected by each of the cities. In the **review info** rows, it is possible to see the number of reviews for each of the general categories considered. In the **POI info**, it is possible to see the amount of POIs within each category. Lastly, we show some basic **statistics** for each of the datasets considered.

In Chapter A we present the maps of each of the cities considered in our study.

Even though TripAdvisor data contains rich features that can be taken as leverage in models, is common to find missing data. For example, in Tiradentes datasets, only 34% of the restaurants have the real price information. This scenario is worse in public spaces (e.g. plazas, churches, and landscapes), where the amount of information is more scarce,

due to the fact that place administrators do not supply much information. In view of the importance of the price information for real scenario evaluation of this problem, in both cases, it was used different strategies to fill the costs. For restaurants and hotels, the price was filled by measuring the similarity between the POIs within each category. We also complement this data by using external data collected from **Quanto Custa Viajar** [2]. In the second scenario, the attractions price was filled as 0, due to the fact that the cost of these public places is irrelevant when compared to hotels and restaurants.

One important aspect that must be considered when dealing with geographical data is the distance between points. If we consider simple metrics such as the Haversine or Euclidean, the distance between the points can be much smaller than the real distance, due to the fact that these metrics do not consider physical constraints between the places. To overcome this problem, we use maps from OpenStreetMaps (OSM) to measure the distance between the POIs in the city. OSM shows as an advantage the fact that all data is open, avoiding costs of consulting the data.

In this work, it is considered climate data in the evaluations. By considering environmental data we avoid recommending to users outdoor places in cases of inappropriate temperatures, such as hot and snowy days. For this task, we gather data from different sources due to the fact that we deal with cities from different countries. For Brazilian cities, we gathered historic climate data from the National Institute of Meteorology (Inmet) [3]. For the European cities, we gathered data from the Climate Change Knowledge Portal (CCKP) [4]. From the INMET data, we have daily climate features since 2006, in this case, we aggregate the min and max temperature monthly. For the CCKP data, we have had a monthly climate since 1991.

5.2 Architecture

Our main goal is to recommend a personalized route for users given their constraints. To accomplish this we divided our proposed methodology into three steps: (*i*) Recommendation, (*ii*) Data treatment and filtering, and (*iii*) Optimization.

The work of [44] previously proposed a methodology with a similar structure called Filter-First, Tour Second. However, such naming does not fit well our work, given that their proposal focuses on selecting top-k places to visit (filter step) and then perform route generation based on these k places (tour step). In Planet Caravan the optimiza-

[2]https://quantocustaviajar.com/
[3]https://portal.inmet.gov.br/
[4]https://climateknowledgeportal.worldbank.org/download-data

tion algorithm has the task to select the best places to visit, due to the fact that will not only consider a place based on its rating but also consider the constraints (e.g. distance, category, working hours). Hence, we name the methodology proposed in Planet Caravan as *Personalize First, Tour Second*. Figure 5.1 gives an overview of our proposed methodology.

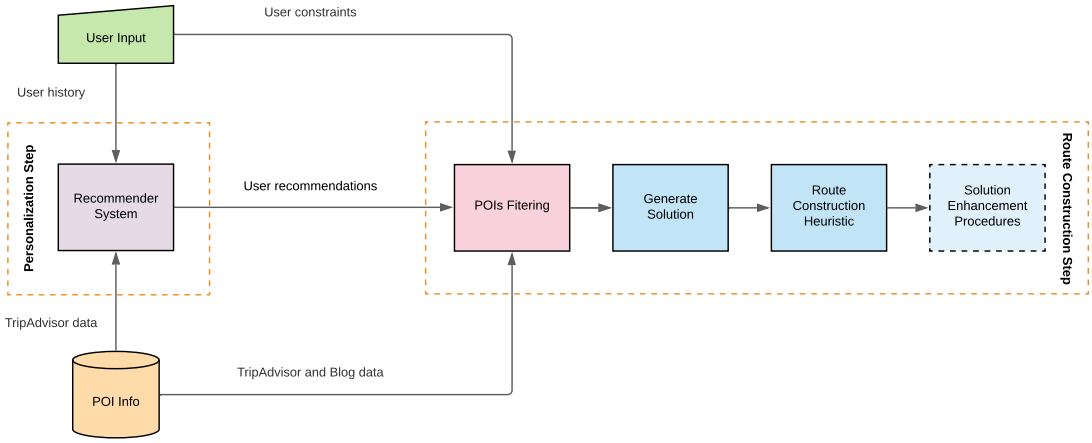


Figure 5.1: Overview of the methodology proposed in Planet Caravan.

Step 1: Recommendation

The first step of our methodology consists of the personalization part, where the focus is to extract users' preferences, and posteriorly make a route that maximizes those preferences in each place. To accomplish that, this module outputs for each user a rating for every place in the town. This way, the heuristics can make a route having a high variety of places to choose from. This step comprehends to *Personalize First*.

TripAdvisor data has explicit feedback in form of ratings that ranges from $1-5$, where 1 is a place that the user did not like and 5 is a place that the user has liked much. Thus, the recommendation algorithms have as input: place id, user id, rating. In our work, we use the CF and random algorithms implementations available in the library Surprise [5] available for the Python language.

Besides that, we also adapted five LBSN POI recommendation algorithms from the literature to evaluate how these algorithms would behave using TSN datasets. As these algorithms are more sophisticated, it takes a bigger amount of features. Table 5.2 shows the features used in each algorithm.

[5] http://surprise.readthedocs.io/

Algorithm	User ID	Place ID	Rating	User History	Geo	Category	Timestamp	Social
CF algorithms	✓	✓	✓	✓	-	-	-	-
USG [82]	✓	✓	-	✓	✓	-	-	✓
GeoSoca [89]	✓	✓	-	-	✓	✓	-	✓
LORE [90]	✓	✓	-	-	✓	-	✓	✓
MGMPFM [15]	✓	✓	-	-	✓	-	-	-
LFBCA [76]	✓	✓	-	-	-	-	-	✓
Unpersonalized	✓	✓	✓	✓	-	-	-	-
Random	✓	✓	✓	-	-	-	-	-

Table 5.2: Features used in each of the LBSN algorithms used in our work.

As we are adapting LBSN POI recommendation algorithms to work with TSN data, some adjustments were made to the algorithms' input and output.

On the inputs, the **USG** [82], **GeoSoca** [89], **LORE** [90] and **LFBCA** [76] algorithms are needed as additional input social-ties data. To generate such data (not present on TripAdvisor), we considered that if two users visited the same place a social connection was formed between them. Other features needed for input as geographic location, timestamp, and category are present in the data collection, thus, no adaptation was needed for such features.

On the output of these algorithms, a top-k recommendation sorted by a rating in the range from $0-1$ (0 most unlikely to recommend, 1 most likely to recommend) is presented. As we aimed to have a prediction for each place in the city, and the TripAdvisor feedback in a $1-5$ range, we selected k equal to the size of POIs in the city, thus having a rating for each place in the city, and multiplied the prediction by 5, hence, getting the output in the TripAdvisor's range. Lastly, it is noteworthy that in our framework any recommender system can be used as input for the optimization step, the only thing needed is output is in the format of User ID, Place ID, and Rating.

Step 2: Data treatment and filtering

The second step of our methodology consists of the data treatment and filtering, where we pre-process the data and join the users' ratings with places features (e.g. working hours, price, category, geo-location). By joining the data, we are able to filter invalid POIs considering the users' parameters. User parameters are inputted by the tourist, allowing Planet Caravan to establish the traveler constraints, filter invalid POIs (e.g. expensive POIs, POIs outside activity hours), and define the aspects of the route. These parameters are:

- ***Travel dates (start and end)*** : The dates that the user plan to be traveling

- **Activity hours**: The start and end hours that the user wants the activities to happen
- ***Transportation type***: Type of transportation that the user will use to move around places (e.g. on foot, car)
- ***Max. daily walk***: Max amount of KMs that a user is willing to cover in a day
- ***Max. category repetition***: Max amount of times that a place in a category can appear in a day
- ***Money budget***: The amount of money that the users plan to spend in the trip
- **Max. places**: Max. amount of places that can be visited in one day
- ***Mandatory places***: A list of places mandatory for the user to visit
- ***Appointments***: A list of places that are mandatory and have a schedule constraint attached
- ***Max. and min. temperature***: The limit temperature for outdoor POI visitation

As the output of this step, we have a user-tailored dataset of places that can be selected by the optimization algorithm. This output is structured to have all the data needed for each of the constraints in the optimization module.

Step 3: Route construction

In the third step of our methodology, we present the module responsible for constructing the routes. This module is divided into two main parts (*i*) Solution Generation and (*ii*) Route Constructing Heuristic.

Solution Generation Considering the initial set of POIs P which contains all valid POIs for the tourist, this set can be dismembered in two subsets: H, which contains hotels and vacation rentals, and A, which contains attractions and restaurants.

The start solution of our problem is given by an array of POIs S^0, where $S^0 \subseteq A \bigcup H$. The size of S^0 is n, where $n = p \times v + 1$, being v the maximum amount of POIs that can be visited in one day, p the number of days that the user will stay in the city, and one hotel.

During the Solution Generation phase, the selected places are defined following the Equation 5.1.

$$S_i^0 = \begin{cases} f_{selection}(H), & \text{if } i = 0 \\ f_{selection}(A), & \text{otherwise,} \end{cases} \tag{5.1}$$

where $i = 0$ is a hotel or vacation rental and $i > 0$ is given by attractions and restaurants.

The selection function $f_{selection}$, can be different according to the heuristic used. If the algorithm is **greedy**, the POIs in S^0 are selected in a greedy fashion according to the place rating, the higher the better. All other algorithms (**Genetic Algorithm, Clonalg, GRASP, and Random**), have a similar selection function that selects POIs to compose S^0 in a random fashion.

Construction Heuristic After defining the start solution S^0, a construction heuristic tries to associate each place in S^0 with a visit time, so-called check-in C. To accomplish so, we develop a greedy algorithm [1] to define the places visiting order, and posteriorly associate each POI to a check-in.

This algorithm generates a final solution S^* by iteratively enqueuing a valid sequence of visits, i.e., a transition from a current place i to a place j that respects all constraints. Every time a place is enqueued in S^* this place is dequeued from S^0. In cases where adding a place i in S^* makes the solution infeasible, this place i is replaced from the beginning to the end of S^0 to be re-evaluated later. This process repeats until all places in S^0 are dequeued, or after visiting a hotel all places in S^0 were evaluated but none is feasible.

Lastly, after defining S^*, we associate each place in S_i^* with a check-in C_i, that is given by a tuple:

$$C_i = (S_i^*, t_i^{ini}, t_i^{end}) \tag{5.2}$$

Where t_i^{ini} is the start and t_i^{end} the end time of each place i in S^*.

The Equation 5.3, defines the start time for each POI. The first place to be visited must be a hotel and is defined to set a location to start the trip. All other places have the start time defined by the end time of the last place and plus the walk time until the current place given by the distance matrix $D_{i-1,i}$.

The Equation 5.4 defines the end time for each POI. The end time for the first place is defined by the start of the travel. For all other hotels, the place end time is defined by the user time to start the activities in the day d. In these cases, every time the users re-visit a hotel the time will passes until the next day activity starts hours. Thus, the hotel will be the last activity of each day and can be considered a daily separator. For all other places, the end time is defined by the start time of that place plus the time recommended to be spent on that place s_i. The time recommended is retrieved from the TripAdvisor data.

$$t_i^{ini} = \begin{cases} U_s^0, & \text{if } i = 0 \\ t_{i-1}^{end} + D_{i-1,i}, & \text{otherwise} \end{cases} \tag{5.3}$$

$$t_i^{end} = \begin{cases} U_s^d, & \text{if } S_0^* = S_i^* \\ t_i^{ini} + s_i, & \text{otherwise} \end{cases} \tag{5.4}$$

After defining the sequence and visit time of each place, our framework verifies if all constraints defined in Chapter 4 are been satisfied. If all constraints are feasible then a positive fitness is returned. Otherwise, the fitness value is multiplied by -1, penalizing the solution. We noteworthy that this penalization does not consider several aspects of a route, for instance, the number of constraints that were violated. In this scenario, in future work we would like to explore better the penalization applied over the route, investigating concepts such as weak and strong constraints.

Algorithm 1 Route Construction Heuristic

```
1:  procedure CONSTRUCT_ROUTE(S^0)
2:      flag ← True
3:      i, n_iter ← 0
4:      h ← S_0^0
5:      S* ← ∅
6:      S*.enqueue(h)
7:      S^0.dequeue(h)
8:      while flag do
9:          if verify_constraints(S*, S_i^0) then
10:             S*.enqueue(S_i^0)
11:             S^0.dequeue()
12:             i ← i + 1
13:             n_iter ← 0
14:             if |S^0| = 0 then
15:                 S*.enqueue(h)
16:                 flag ← False
17:         else
18:             S^0.enqueue(S_i^0)
19:             S^0.dequeue()
20:             if n_iter = |S^0| then
21:                 n ← |S*| − 1
22:                 if S*[n] = h then
23:                     flag ← False
24:                 else
25:                     S*.enqueue(h)
26:                     n_iter ← 0
```

5.3 Heuristics

Recall that the Tourism Trip Design Problem corresponds to a junction of two different problems, the TSP and the Knapsack problem[78, 34, 44], and as shown in the work of [39] this is an NP-Hard problem. Hence, it would not be possible to find an optimal solution in a timely manner given that this problem belongs to the NP-Complete class [16]. In this case, the solution found was to use algorithms that can achieve not-exact-acceptable results in a reasonable amount of time, the so-called heuristics [31]. These heuristics have specific enhancement procedures that generate solutions with better fitness.

In this work, we test five different optimization heuristics to solve the TTDP, namely: **Random, Greedy, GRASP, Genetic Algorithm (GA), Clonalg**. As presented in Chapter 3, the **Random Greedy** are naive approaches,**GRASP** based on a greedy randomized technique, and **GA** and **Clonalg** are bio-inspired techniques.

All these algorithms take as input a vector $P = \{S_0^1, S_0^2, ..., S_0^n\}$ of solutions, where n is the size of the population. As pointed before for **Random, GRASP, GA** and **Clonalg** the vector P will be of random solutions, while for the **Greedy** algorithm the vector P will be of greedy solutions. Recall that this step is described before as **Solution Generation** step. After, we pass through the **Construction Heuristic** step, aiming to measure the quality of each solution. It is noteworthy that at end of this process, no enhancement procedure has been made in the solutions, but each solution S_0^i has an associated fitness in a vector $F = \{f_1, f_2, ..., f_n\}$, where $f_i \in \mathbb{R}$.

Naive Techniques Based on the fitness f_i, both naive techniques will return as output a solution $i \in P$ where $f_i = Max(F)$. It is noteworthy that for these approaches no enhancement procedure is made over the solution initially generated on the **Solution Generation** step. For the other heuristics, different enhancement procedures are applied over P.

In this scenario, while the **random algorithm** will naturally generate different solutions, the **greedy algorithm** will select the places with the highest fitness to compose the route and make several routes with different orders for these selected places. With the shuffling, even though the POIs in the greedy solutions are similar, the order of the places is different, hence, producing different fitness values. In cases where several places have the same rating (e.g. all places have the same rating), the selection is made in a random fashion between those POIs with the highest ratings.

GRASP Recall that **GRASP** works with two main operations **Solution Construction** and **Local Search**. The stage of **Solution Construction** is made by taking as

input the vector P and measuring the fitness F for each solution. For the solutions P, a solutions, where $a \in Z_+|a < |P|$, are selected to compose the **Restricted Candidate List**, based on a greedy criteria which aims to selected those solutions with highest fitness. This process generates an sorted by fitness vector of solutions $P^{sel} = \{S_0^i, S_0^j, ..., S_0^k\}$, where $(i, j, k) \in P$, $|P^{sel}| = a$ and $F_{P_0^{sel}} \geq F_{P_1^{sel}} \geq ... \geq F_{P_a^{sel}}$. From the P^{sel} vector a solution must be selected. In our scenario an solution $s = P_i^{sel}|i \in P^{sel}$, where i is defined based on the cardinality approach, given by the Equation 5.5. Being $\alpha \in \mathbb{R}|0 \leq \alpha \leq 1$ a hyper-parameter that defines how greedy is the selection criteria, thus as closer of 0 α is the greedier is the selection criteria.

$$i = \lfloor |P^{sel}| \times \alpha \rfloor \tag{5.5}$$

Algorithm 2 Local Search

```
    procedure REPLACE_PLACES(S^0, A, R, T, C, C)
2:      i ← 0
        S* ← S^0
4:      for p ∈ S^0 do
            if R_p < 5 then
6:              similar_places ← retrieve_best_similar_POI(p, A, R, T, C, C)
                flag ← True
8:              j ← 0
                while flag do
10:                 S*_i ← similar_places_j
                    if verify_constraints(S*) then
12:                     flag ← False
                    else
14:                     S*_i ← p
                    j ← j + 1
16:         i ← i + 1
```

After selecting a solution s, the next step of **GRASP** consists on the **Local Search** over the solution s. The **Local Search** works by iteratively replacing the POIs that compose the solution s. It is noteworthy, that the task of replacing the POIs in the solution is not so trivial, given that each POI is attached to the visiting hours and distance of the previous and next POI. To solve this problem we propose our local search, presented in [2], that greedily identifies POIs similar to the one being replaced. The characteristics considered by our algorithm to evaluate if a place can be the substitute is similar category $\mathcal{C}$, cost C and working hours T, and a rating R that must be higher than the rating of the POI being replaced. By considering such similar aspects we avoid disrespecting the category and visiting hours constraints, and by considering a place with a higher rating we increase the solution fitness. At end of the Algorithm[2], a new solution S^* is returned with all the POIs swapped from the solution. **GRASP** will repeat the

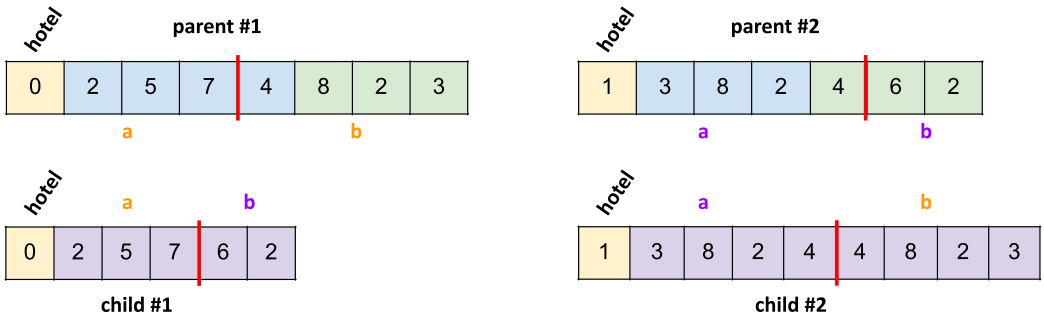


Figure 5.2: Crossover step performed in the Genetic Algorithm.

procedures described above for x iterations and will have as output the solution S^* which has the biggest fitness. It is noteworthy that the local search operation is very costly, due to the fact that each swap demands a re-evaluation of the solutions, which is made by the procedure that verifies constraints.

Bio-Inspired Techniques Now we present the enhancement procedures of the two bio-inspired heuristics tested in our work **GA** and **Clonalg**. The **GA** is characterized for having three solution enhancement procedures: **Selection, Crossover, and Mutation**, while **Clonalg** also present three enhancement procedures: **Selection, Clone, and Mutation**.

For the **selection operation**, which aims to select those solutions that are best fitted, we use for both **GA** and **Clonalg** a tournament selection criteria. For the **Tournament based selection**, the heuristics randomly selects two solutions i, j, where $i, j \in P$. These two solutions are compared in terms of fitness, and the solution with the highest fitness is selected to compose a new population P^{sel}. The Equation 5.6 shows how the selection population is defined. It is noteworthy that the max function returns the i or j according to each solution with the highest fitness.

$$P_k^{sel} = P_{max(F_i,F_j)} \ \forall k \in \{0, 1, ..., n-1\} \tag{5.6}$$

The second operator of the **GA**, known as the **crossover operator**, aims to enhance the quality of solutions by mixing the routes generated. The idea behind the crossover is the simulation of the mixing of genetic material when organisms reproduce [20, 56]. For this work, we use a **cut and splice technique**, which works by randomly selecting solutions $i, j \in P^{sel}$, and for those solutions, the operator arbitrarily selects cut points and joins these cuts to generate new solutions. For each pair of solutions i and j selected, two new individuals s_k^c and $s_(k+1)^c$ will be generated as result of cut and splice operator, where $k \in Z_+$. The output of this step is a population $P^c = \{s_1^c, s_2^c, ..., s_n^c\}$. Figure 5.2 illustrates how the cut and splice work on the individuals in our context.

For the **Clonalg**, the second operation, known as **clone**, aims to replicate the solutions directly proportional to its fitness score. The amount of clones for each individual is defined by the condition Equation 5.7, where a is the maximum amount of copies that a solution can have.

$$P_k^c = \begin{cases} \lceil (\frac{F_k}{max(F)}) \times a \rceil & \text{if } f > 0, \forall k \in \{1, 2, ..., n\} \\ 0 & \text{otherwise} \end{cases} \tag{5.7}$$

After defining the number of copies of each individual a new vector of solutions called P^c is created with all the clones for each solution.

The last step of both **GA** and **Clonalg** are similar and aim to input variability on the solutions generated. The **mutation step** works by randomly selecting individuals $i \in P^c$ based on a probability p that is a hyper-parameter of the model. For each solution selected arbitrary genes are chosen and then replace with other random genes. It is noteworthy that we do not consider hotels (the first position of each solution) in the mutation step, given that by replacing the hotel there are chances of creating an invalid individual due to the distance between the hotel and the POIs (distance constraint). At end of this process, a population $P = \{s_1^m, s_2^m, ..., s_n^m\}$ of mutated solutions is generated. Figure 5.3 illustrates the mutation for our scenario.

The enhancement procedures presented above are then repeated until it reaches the stopping criteria. It is noteworthy that the route recommended is the one with maximum fitness when the process stops. Thus, the final solution recommended is generated at the construction heuristic step, hence, having a check-in for each place that will be visited.

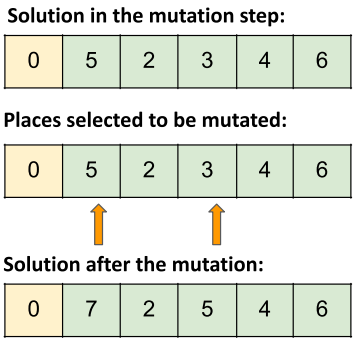


Figure 5.3: Mutation step example.

5.4 Experimental Evaluation

In this section, we present the experimental evaluation and parameters used to evaluate the algorithms. Given the high amount of parameters for the RS, heuristics, and the users, we opt to define a small set of parameters for each step. In this section, we also show the metric used to evaluate the RS and describe how we assess the quality of the routes generated by Planet Caravan.

5.4.1 Parametrization

In the RS step, we used standard parameters defined in the implementations. For the heuristics we make an empiric study selecting the parameters that better balance the fitness and the execution time. In Table 5.3, we define each of the hyperparameters used in the heuristics. **Greedy** and **random algorithms** are not present given their lack of hyper-parameters.

	GA	**Clonalg**	**GRASP**
# of Iterations (Generations)	100	50	100
# of Solutions (Population)	100	50	100
Mutation Rate	0.15	0.2	-
Max Amount of Clones	-	3	-
# Selected to Clone	-	15	-
Selection Criteria	-	-	Cardinality
Alpha	-	-	0.1

Table 5.3: Users' parametrization profile.

For the users, we select a set of parameters to be evaluated, which evaluate different aspects that we consider important when traveling that is, the total cost of a trip and the distance traveled in one day. Table 5.4 describes these users' profiles.

Profile	**Money**	**Max. Distance**	**Description**
A	2500	10 KM	Spends a little, walks a little
B	2500	20 KM	Spends a little, walks a lot
C	5000	15 KM	Spends average, walks average
D	10000	10 KM	Spends a lot, walks a little
E	10000	20 KM	Spends a lot, walks a lot

Table 5.4: Users' parametrization profile.

Besides that, the other parameters needed for the users are fixed. For each city, we use a different set of parameters, given the differences in their environment (e.g. climate, the distance between points).

To better illustrate that, consider the cities Ibiza and Ouro Preto. Ibiza is the city with the second largest area while Ouro Preto is the city with the largest area. If we evaluate the distribution of the POIs through the maps available we will see that most of the POIs in Ouro Preto are concentrated in a small region, while the POIs in Ibiza are spread through the island. In our evaluation, we realize that in some cities due to this broad distribution of the places, walking on foot was not a viable option. Thus, in these cases, we use a transportation mode by car.

The main difference between traveling on foot or by car is the speed, $1.38m/s$ and $8.3m/s$, respectively, and, the type of map that is used. In the first case, it is used a walking map, which gives the shortest paths between the places without considering road restrictions. This enables the user to use shortcuts to get into places in a shorter time. In the second case, a drivable map is used, which will give the shortest distance between two places while respecting road restrictions. It is noteworthy that in both cases the geographical constraints will be respected.

Table 5.5 shows a set of specific parameters for each city showing the set of fixed parameters. For other users' parameters, we define them later in Table 5.6. It is noteworthy, that due to stochastic factors in all algorithms, to statistically guarantee our results all parameter combinations were made 100 times.

	Tiradentes	**Ouro Preto**	**San. Gimiagno**	**Cannes**	**Ibiza**
Month of Travel	1	1	7	7	7
Activity hours	8 AM - 22 PM	8 AM - 22 PM	8 AM - 22 PM	8 AM - 22 PM	8 AM - 22 PM
Max. category repetition	4	4	4	4	4
Max. places	7	7	7	7	7
Min. Temperature	21	20	20	20	22
Max. Temperature	25	25	27	25	28
Transportation Mode	Foot	Foot	Foot	Foot	Car

Table 5.5: Users' parametrization profile.

5.4.2 Recommender Systems Evaluation

We evaluate the accuracy of RS by the means of Root Mean Squared Error (RMSE). In comparison to other regression metrics, RMSE presents advantages in the fact that it maintains the scale of the values, which makes it more interpretable than the other metrics. As lower the value of the RMSE, the better is the model. RMSE is defined in the context of RS as:

$$RMSE = \sqrt{\frac{1}{|\hat{N}|} \sum_{\hat{y}_{ui} \in \hat{N}} (y_{ui} - \hat{y}_{ui})^2}, \tag{5.8}$$

where $\hat{N}$ is the set of predictions made by the algorithm, y_{ui} is the real rating given by a user u for an item i, and $\hat{y}_{ui}$ is the prediction rating of the item i for the user u.

We also use Temporal K-Fold Validation (Time Series Spit) to statistically guarantee the results of RS. In this approach we divided the data into fixed time intervals, training with $k-1$ of the folds and testing with 1. In our work, we consider $k = 5$. Figure 5.4, illustrates how the temporal k-fold is performed in our methodology. The first graph shows the complete data of the city. The other figures present how the train and test data are selected. We can see that the model always trains with the past to predict the future.

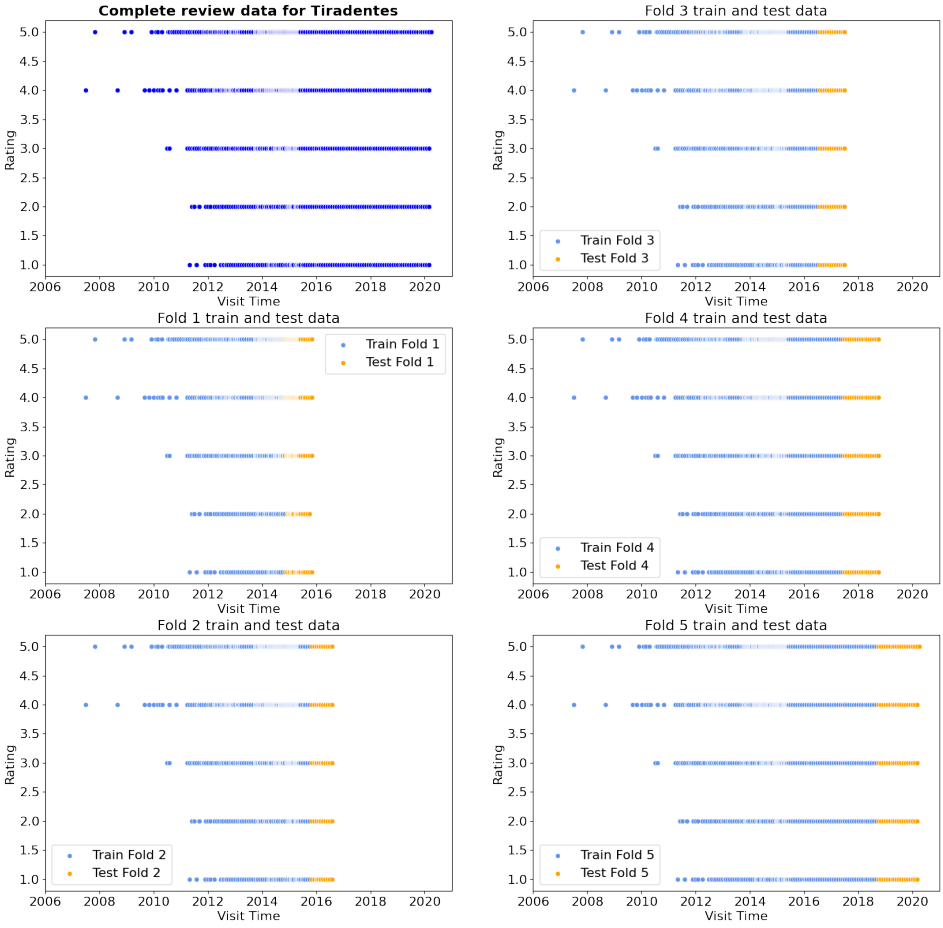


Figure 5.4: Temporal K-Fold example.

5.4.3 Route Evaluation

In our evaluation, given the high amount of users and parameters, we use a pre-selected set of parameters defined in Table 5.5.

Parameter	Min. Value	Max. Value
# days of travels	3	6
# of mandatory places	0	3
# of Appointments	0	3

Table 5.6: Variable parameters.

To evaluate the routes generated, we use a qualitative approach. The survey [50] specifies two different ways to evaluate such routes: (*i*) real-life evaluation, and (*ii*) heuristic-based evaluation.

In the first scenario, the generated routes are compared with the routes previously made by the user evaluating in terms of information retrieval metrics, such as precision, recall, and f-score [9, 85, 14, 65]. In the second scenario, the routes are evaluated in terms of the utility (sum of the ratings of visited places), the popularity of the POIs visited and the amount of POIs visited in the tour.

One disadvantage of the first evaluation is that it does not consider the benefit of taking that route, but how close the routes generated are to the users' previous routes. Nevertheless, as the users do not make optimal choices, we believe that this evaluation leaves aside that the routes generated must be better than the users' past routes.

In this regard, works that use the second approach, namely [53, 44, 29], make a qualitative evaluation in terms of utility, considering the benefit of the users for using the routes recommended by their system. Nevertheless, these works fail in, (*i*) show the gain of the recommended route over routes previously done by the user, and (*ii*) fail to test multiple users, only verifying one user. Thus, the practical benefit of using a recommender system to provide personalized routes is left aside.

Aiming to fill these gaps and make a deep evaluation of our framework, in this work we make a qualitative evaluation of the routes generated. For each city, we select 4 different users to be evaluated, 2 travelers with few visits, and 2 users with a broad history within a period of time. Hence, testing in a scenario with a broad history, and in the cold-start scenario.

Chapter 6

Experimental Results

In this chapter, we present the results achieved through the proposed methodology. First, we discuss the personalization step results, analyzing several types of RS that can be applied to the tourism context and answering our **RQ1** in Section 6.1. For that analysis, we use the RS techniques introduced in the previous chapter (see Table 5.2). We also evaluate the quality of LBSN RS on the datasets that we collected.

In Section 6.2 we make an analysis of our entire framework. We analyze the behavior of the heuristics under different sets of parameters, assessing their quality under different scenarios. Through this analysis, we answer the **RQ2**. We also evaluate the impact of the user parameters and the constraints on the quality of the generated routes, answering the **RQ3**. We compare the route generated by Planet Caravan with the routes in the user history and identify the tourists gain by using our approach. Besides that, we also verify cases where the quality of the generated routes is inadequate, establishing scenarios where is not possible to generate feasible routes.

6.1 Recommender System

Recall that the RS plays an important role in our methodology, given that, together with the user parameters, RS is the module responsible for providing personalization to the generated routes. Hence, in order to make a complete evaluation over the datasets collected, we compare 35 different RS strategies, between then, 3 unpersonalized naive models, 5 literature LBSN RS, 3 model-based CF techniques, and 24 memory-based CF with variations on the modeling and similarity metric.

Among the algorithms evaluated, only the LBSN RS, namely **GeoSoca, LFBCA, LORE, MGMPFM**, and **USG**, use a bigger amount of features, as shown in Table 5.2. Even though these last techniques are suited for LBSN datasets, in this work we investigate

if these RS can perform well on our datasets.

Table 6.1 presents the average results and confidence interval of each algorithm. To compare the average results of the cross-validation (with 5 folds), we analyze the statistical significance with a paired t-test with 95% confidence. When comparing several algorithms this technique is strongly recommended for hypothesis testing on average effectiveness [72]. In the table, results with a green arrow present the best algorithm, when presenting a yellow circle the table shows algorithms that are statistically equal in results.

Technique	**Ouro Preto**	**Tiradentes**	**SG**	**Cannes**	**Ibiza**
User Non-Personalized	0.83(0.02)	0.86(0.04)	1.12(0.04)	1.01(0.02)	1.11(0.04)
Item Non-Personalized	0.89(0.05)	0.85(0.03)	0.90(0.06)	0.98(0.03)	0.98(0.03)
SVD	0.91(0.06) ▲	0.87(0.04) ▲	0.92(0.06) ▲	0.99(0.03) ▲	1.00(0.03)
SVD++	0.91(0.06)	0.88(0.04)	0.94(0.07)	1.00(0.03)	1.00(0.03)
Basic User Msd	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.07(0.03)	1.06(0.04)
Basic User Pearson	0.96(0.06)	0.92(0.04)	1.00(0.07)	1.07(0.03)	1.06(0.04)
Basic User Cosine	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.08(0.03)	1.06(0.04)
Mean Item Pearson	0.96(0.06)	0.92(0.04)	1.01(0.06)	1.07(0.04)	1.06(0.04)
Z-Score Item Pearson	0.96(0.06)	0.92(0.04)	1.01(0.06)	1.07(0.04)	1.06(0.04)
Basic Item Pearson	0.96(0.06)	0.93(0.04)	1.02(0.06)	1.08(0.04)	1.07(0.04)
Basic User Pearson Baseline	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.07(0.03)	1.06(0.04)
Z-Score Item Pearson Baseline	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.07(0.04)	1.06(0.04)
Mean Item Pearson Baseline	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.07(0.04)	1.06(0.04)
Mean Item Cosine	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.08(0.03)	1.07(0.04)
Z-Score Item Cosine	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.08(0.03)	1.07(0.04)
Mean Item Msd	0.96(0.06)	0.92(0.04)	1.02(0.06)	1.08(0.04)	1.07(0.04)
Basic Item Pearson Baseline	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.08(0.04)	1.07(0.04)
Z-Score Item Msd	0.96(0.06)	0.93(0.04)	1.02(0.06)	1.08(0.04)	1.07(0.04)
Basic Item Msd	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.09(0.04)	1.08(0.05)
Basic Item Cosine	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.09(0.04)	1.08(0.05)
Mean User Msd	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.09(0.03)	1.08(0.04)
Z-Score User Msd	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.09(0.03)	1.08(0.04)
Z-Score User Cosine	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.09(0.03)	1.08(0.04)
Mean User Cosine	0.97(0.06)	0.93(0.04)	1.02(0.06)	1.09(0.03)	1.08(0.04)
Z-Score User Pearson	0.97(0.05)	0.94(0.04)	1.01(0.07)	1.10(0.03)	1.09(0.04)
Mean User Pearson	0.97(0.05)	0.94(0.04)	1.01(0.07)	1.10(0.03)	1.09(0.04)
Z-Score User Pearson Baseline	0.98(0.05)	0.93(0.04)	1.02(0.06)	1.10(0.03)	1.09(0.04)
Mean User Pearson Baseline	0.98(0.05)	0.93(0.04)	1.02(0.06)	1.10(0.03)	1.09(0.04)
NMF	0.98(0.05)	0.95(0.04)	1.03(0.06)	1.11(0.04)	1.10(0.04)
LFBCA	1.20(0.07)	1.13(0.08)	1.16(0.06)	1.32(0.03)	1.22(0.03)
RANDOM	1.25(0.04)	1.19(0.04)	1.27(0.07)	1.42(0.03)	1.35(0.05)
LORE	1.35(0.10)	1.52(0.11)	1.24(0.09)	1.41(0.08)	1.55(0.11)
GEOSOCA	1.44(0.15)	1.60(0.17)	1.26(0.10)	1.57(0.15)	1.83(0.19)
MGMPFM	1.61(0.13)	1.75(0.16)	1.34(0.11)	1.84(0.22)	2.24(0.23)
USG	1.76(0.17)	1.84(0.17)	1.44(0.15)	1.97(0.28)	1.73(0.28)

Table 6.1: Recommender Systems RMSE Results for each of the cities. We highlight the best and worst values for the **personalized algorithms**.

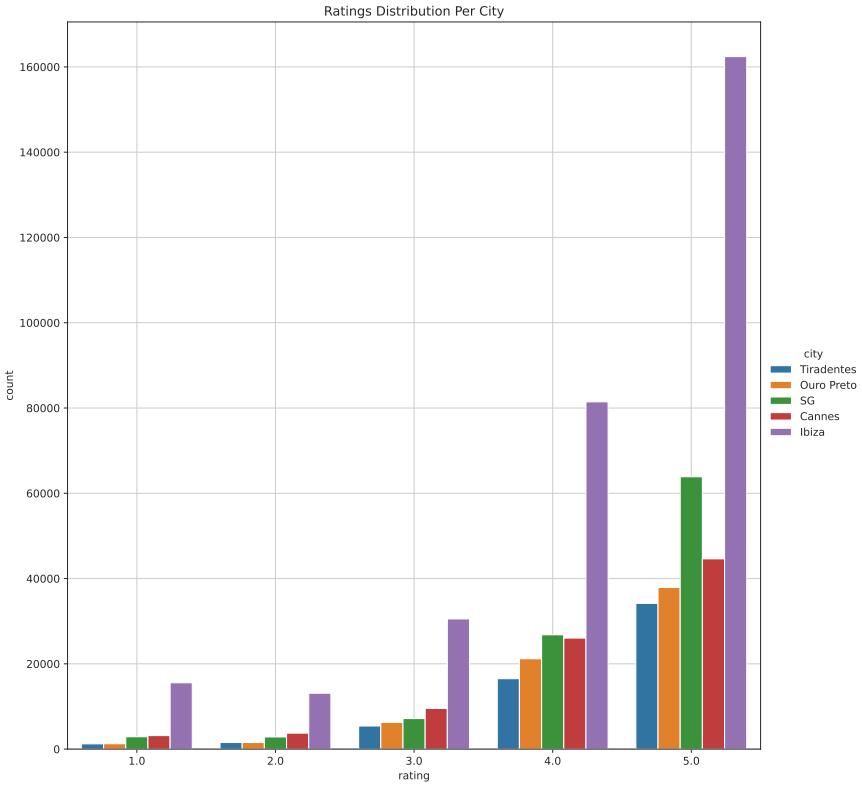


Figure 6.1: Distribution of the ratings per city. As we can see the ratings are mostly concentrated between 4 and 5.

Note that in some cases non-personalized methods (**Item and User Non-Personalized**) show a better RMSE than personalized results. This is due to the distribution of the ratings over the cities, which are mostly concentrated between the 4 and 5 ratings, as we can see through Figure 6.1. Nevertheless, as these models are not personalized, it does not offer unique recommendation per user. Thus, non-personalized techniques work very well in most cases, nevertheless, in cases where the user does not like a place, these models will produce results that can lead to a bad user experience. Hence, in this works it is considered the **best-personalized approach** that minimizes the RMSE value. In this scenario, the best technique for all cities was **SVD**.

By analyzing Table 6.1, it is possible to see that **SVD** and **SVD++** show the best results in our tests, suggesting that model-based techniques are superior to CF techniques. Nevertheless, **NMF**, which is also a model-based approach, figures between the worst results. We believe that this is due to the fact that **NMF** works better with implicit feedback [2], while our datasets are completed composed of explicit feedback.

In this work, we tested many memory-based CF techniques. Even though their results are not the best, these techniques show a good RMSE value and could be used in scenarios where the users value more the interpretation of the results, as where their recommendations come from. Nevertheless, even with the combination of similarity techniques and different prediction functions, these combinations do not show a significant

result change.

Even though the **random algorithm** is a simple baseline, we observe through Table 6.1, that the random approach has a better performance in most of the cities than the LBSN POI RS considered in our work. In addition to that, these LBSN POI RS techniques show the worst performance of all the algorithms considered. For the cities of Ouro Preto, Tiradentes, SG, and Cannes the **USG** technique show the worst results, while for the Ibiza city, **MGMPFM** shows the lowest RMSE values. We believe that these bad results are a sum of several factors.

USG, LORE, LFBCA, and **GeoSoca** uses social ties to enhance the quality of the model. As stated before, it was not possible to collect the social ties present in the TripAdvisor platform. This directly impacts these algorithms' prediction quality, due to the fact that these relations were constructed based on the users' co-visits on the same place, instead of their followings/followers as it is on TripAdvisor.

USG and **LORE** use a similar approach to measure social ties. These algorithms consider that users are connected given how close is their place most visited, which is called residence [51]. In our case, given that the users are tourists, most of them stay at hotels/vacation rentals, nevertheless, they only make one review of these places. Thus, the strategy of defining the social-ties by the user residence does not work on TripAdvisor data.

LORE relies on the timestamp of the visit to identify patterns of sequential check-ins. As TripAdvisor's timestamp data is in a higher granularity (monthly) than LBSN datasets (seconds), this model fails to identify this sequential influence. As **LORE** is both time and social-aware, this algorithm is not appropriate for the TripAdvisor dataset.

LFBCA is the only of the models that perform better than the **random algorithm** for all cases. Even though this technique relies on social-ties to perform its recommendation, the first step of this technique is based on Bookmark-Coloring Algorithm (BCA) to identify the similarity between users [51]. Posteriorly, this similarity is used to perform recommendations on a CF algorithm. As most of the users only visit a few places, the similarity between the users tends to be higher, given that most of the users visit the same places. Consequently, if ratings were considered as a part of the prediction equation of the CF algorithm, most of the predictions would be between 4 and 5 given the high similarity, thus, the RMSE would be lower. However, as the output is a ranking and we do not weigh the similarity and the rating given by the user, this algorithm does not have a better performance.

The algorithm **MGMPFM** only takes into consideration the geographical influence when making the recommendation, which is properly represented in the TripAdvisor dataset. Nevertheless, this algorithm shows the worst results for the city of Ibiza. As this algorithm uses a Poisson Factor Model (PFM) to properly fit the data, several parameters are taken. As we used the standard parametrization, we believe that with a

good parameterization it would be possible to achieve better results for the TripAdvisor datasets. Nevertheless, the study made by [51] **MGMPFM** shows poor performance for their experiments, considering datasets from LBSN. Thus, it is possible that this model does not have a good performance in any scenario, hence, the parametrization would not produce better results.

Besides the points listed above, these algorithms, as stated before, were implemented to generate top-k rankings for LBSN networks. Thus, most of these works do not perform a metric reduction to enhance the quality of prediction. Second, by evaluating Figure 6.2, it is possible to see that LBSN presents a different visit distribution when compared to the visits of TripAdvisor, which can lead to these bad results. LBSN users are more frequent even in places with few visits, and each user has at least 10 visits.

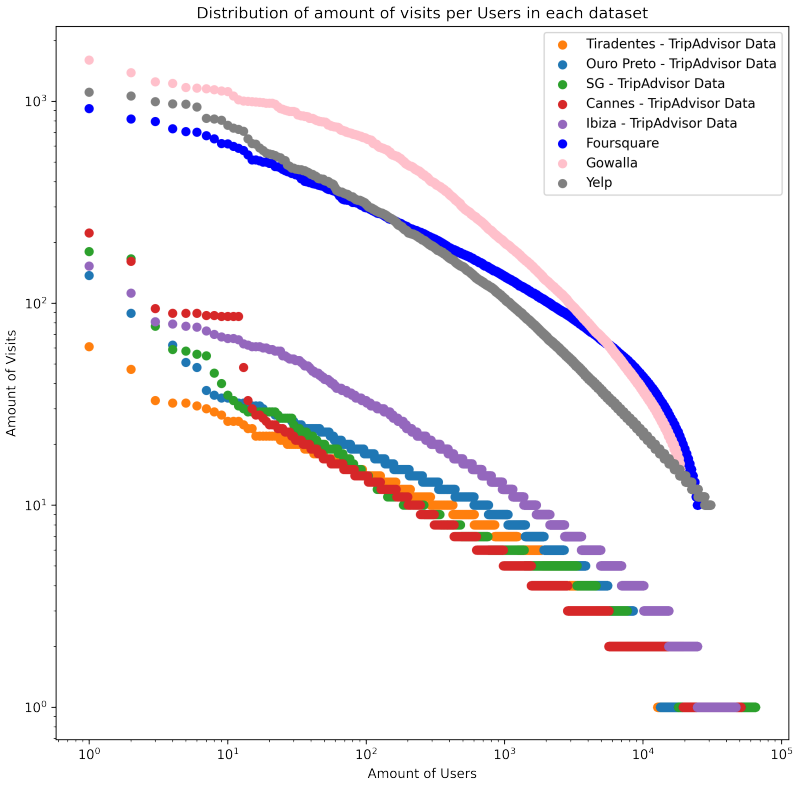


Figure 6.2: Distribution of the number of visits per dataset.

Thus, our evaluations show that LBSN baselines do not perform well on TripAdvisor datasets, and this is mostly due to the granularity of the data. Nevertheless, we believe that with major adaptations in the models it would be possible to use the influence factors considered in these approaches. Nevertheless, works that rely on features that are not properly presented in the TripAdvisor dataset, such as social ties and visit time, we believe that is harder to achieve good results.

Lastly, answering the **RQ1: Which type of Recommender System is better suited for defining the user ratings for each place in the TripAdvisor dataset?** our conclusion is that **SVD** is the algorithm better suited for the TripAdvisor dataset.

Nevertheless, this algorithm works by optimizing an objective function that aims to increase the quality of prediction by reducing the error. Thus, the model misses more on ratings that are far from the average. Illustrating that, if a place has a rating near 4.5, the model will miss more on users that give bad ratings, under 3. Hence, we believe that by making a better parametrization of the RS algorithm can lead to better results. Lastly, we also believe that a content-based recommender system can achieve good results in TripAdvisor data, even more considering that these approaches would take advantage of the descriptive features available, thus, we could use data from other cities to provide good predictions in the target town.

6.2 Heuristics

In this section, we evaluate the results achieved by using different heuristics to solve the TTDP. First, we show the user templates that we select for each city. The users' selected aim is to evaluate the proposed methodology in the cold start and non-cold start scenarios to show how our algorithm performs in the selection of places and how our selection is often better than the one performed by users.

Next, we evaluate the utility of the generated routes, analyzing the impact of different economic profiles on fitness. In this assessment, we answer the **RQ2**, analyze and define the best algorithm in each scenario. Further, we answer the **RQ3** by evaluating the impact of the constraints on the routes generated and analyzing how POI, user, and systems constraints can change the tourist experience in several scenarios.

6.2.1 Users Selected

As described in Chapter 5, in this work we select two users with past experience in the city and two users with few experiences in a city (cold start). Table 6.2 presents users' history and predicted ratings. It is possible to notice that the newcomers (i.e. cold-start users) of each city do not have a standard deviation of history ratings, given that we select only users that have only one visit in the city. To complete the ratings of each user we use the **SVD** algorithm. The results show that the mean history rating is very close to the mean predicted rating.

		Experienced User 0	**Experienced User** 1	**New Comer** 0	**New Comer** 1
Tiradentes	# History Ratings	15	18	1	1
	Mean History Rating	3.466667	4.611111	5	4
	Std. History Ratings	1.245946	0.501631	–	–
	Mean Predicted Rating	3.939829	4.584593	4.355915	4.252029
	Std. Predicted Ratings	0.350917	0.356552	0.373764	0.377525
Ouro Preto	# History Ratings	17	83	1	1
	Mean History Rating	4.470588	4.409639	5	5
	Std. History Ratings	0.624264	0.781433	–	–
	Mean Predicted Rating	4.410237	4.348878	4.330977	4.29
	Std. Predicted Ratings	0.333199	0.349962	0.336659	0.338050
San Gimi.	# History Ratings	25	21	1	1
	Mean History Rating	4.040000	4.333333	5	5
	Std. History Ratings	0.888819	0.730297	–	–
	Mean Predicted Rating	4.254626	4.586697	4.435897	4.437126
	Std. Predicted Ratings	0.356542	0.340272	0.390403	0.357538
Cannes	# History Ratings	7	17	1	1
	Mean History Rating	4.571429	4.470588	5	5
	Std. History Ratings	0.534522	0.874475	–	–
	Mean Predicted Rating	4.324704	4.430897	4.199465	4.155052
	Std. Predicted Ratings	0.302214	0.310592	0.303201	0.312004
Ibiza	# History Ratings	13	11	1	1
	Mean History Rating	3.846154	4	4	4
	Std. History Ratings	1.214232	0.632456	–	–
	Mean Predicted Rating	3.932483	4.117522	4.072198	4.097940
	Std. Predicted Ratings	0.293937	0.295872	0.300013	0.299694

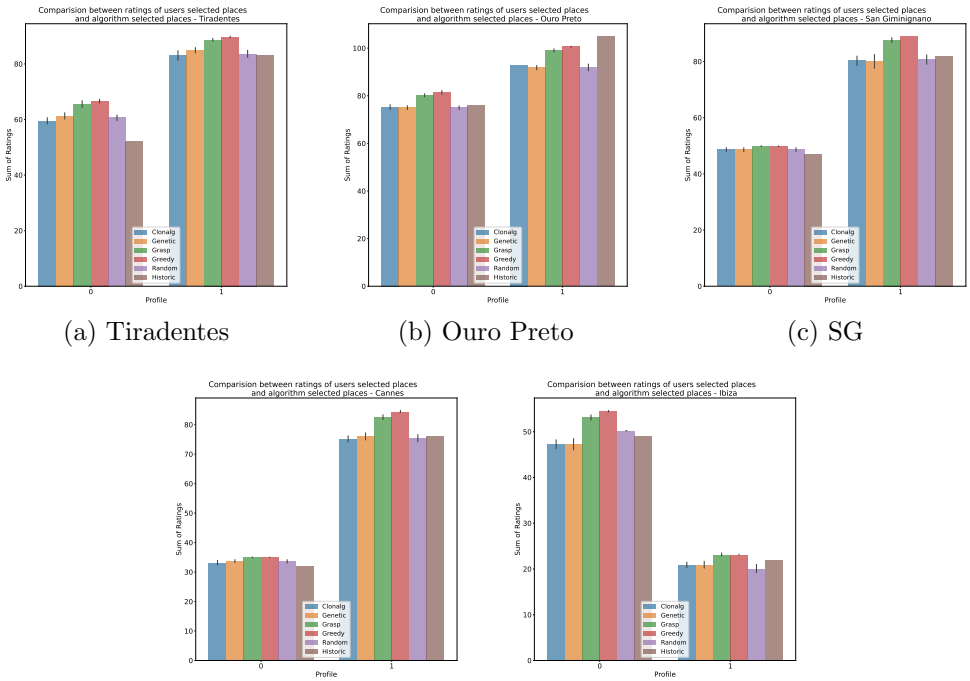
Table 6.2: Selected users' ratings description.

6.2.2 Place Selection

Before generating a route, we must select the places that will compose that route. Nevertheless, selecting these places is not a trivial task, due to places and users' constraints. We show the effectiveness of our proposal in the place selection task through two different evaluations: a **user history** and a **theoretical**.

In the **user history evaluation**, the POIs selected by each algorithm are compared with the POIs historically visited by the user. In the **theoretical evaluation**, it is evaluated how close is the places selected by the algorithm to the best places (top-k) in a city. In this second evaluation, place, and users constraints are not considered, making the best places selection a **theoretical upper bound**. It is noteworthy that in both cases we are not comparing the routes by utility (fitness), but by the sum of ratings of the places selected. Thus, place repetition is not considered to avoid the theoretical route to have only one place re-visited several times. For example, in the utility measure, the hotel rating is considered daily, while in the sum of the ratings, as we do not consider repetition, the hotels rating is only considered one time. In addition in the theoretical evaluation, we divided the sum of the ratings by the **theoretical upper bound**. Thus, in this analysis, as close is the value to 1, the closer is the place selection to the **theoretical upper bound**.

Figure 6.3, shows the results of the users' history evaluation, presenting a comparison between the users' history place selection and the selection made by algorithms. It is noteworthy that in most of the cities we had a better place selection than the ones performed by the users.



(a) Tiradentes (b) Ouro Preto (c) SG

(d) Cannes (e) Ibiza

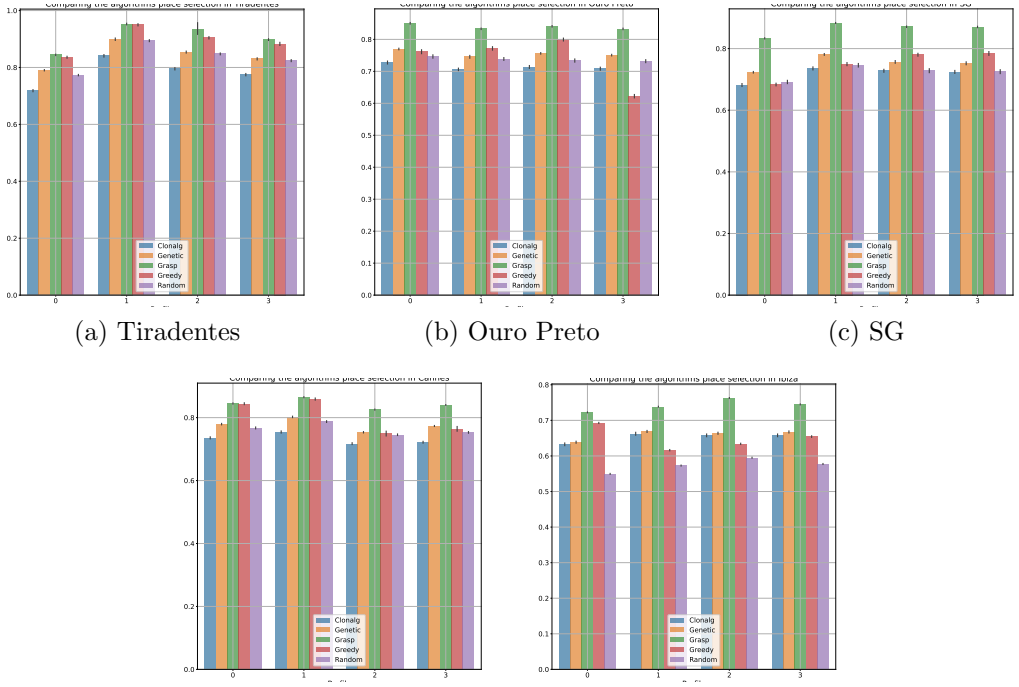
Figure 6.3: Comparison between the users' history place selection and the algorithmic selection.

One of the few scenarios where the user selection was better than ours is for the user 1 in Ouro Preto. In this case, the user selection is better due to the number of places previously visited by this tourist. Table 6.2, shows that the user 1 had previously visited about 83 places. As we did not generate routes with the same size as the users' history, to make a fair comparison, we reduce the size of the user history to k, considering that k is the size of the biggest route generated for this user. It was selected the best k places in the user history. As these k places visited by this user had a rate equal to 5 the sum of ratings was maximum. Thus, in this scenario where the users tend to give the highest rates to each place, we could not select better places to visit than the user. Nevertheless, in cases where the users are more critical, our proposal shows its potential to select better places for tourist visitation.

In addition, note that the **greedy algorithm** has a better place selection in several cases. This happens due that the **greedy algorithm** always selects the best places in the city to be visited. It is noteworthy, however, that this does not mean that the route generated will be feasible. Illustrating that, consider that the top places selected are all from the same category, then the route would not comport all POIs due to category

constraint. In this case, the construction heuristic would remove places aiming to make the route feasible, hence reducing the fitness. Thus, in some scenarios, the **GRASP** algorithm can have a better selection than the **greedy algorithm** due to the local search. The local search enables the **GRASP** to make feasible routes while selecting the places of the highest rating. In scenarios where the local search cannot guarantee the feasibility of the place swap, **GRASP** at least guarantee a feasible route.

Second, for the **theoretical evaluation**, corroborating with the previous results, the algorithms with the best selection (closest to upper bound) are the **greedy** and **GRASP**. The Figure 6.4 shows the theoretical evaluation results.



(a) Tiradentes (b) Ouro Preto (c) SG

(d) Cannes (e) Ibiza

Figure 6.4: Comparison between the best theoretical place selection and the algorithms place selection.

Genetic and **clonalg** algorithms have a poor selection, very close to the **random** algorithm. For these algorithms, their inadequate performance can be justified by the evolutive operators. As these evolutive operations did not consider a repetition constraint, the routes generated contain multiple repeated places, which were not counted in this analysis.

In Chapter A, we present the frequency of places selected by each algorithm. Through the figures, it is possible to notice how close are the selection of each algorithm. **Random** and **genetic** are very close with their selection spread through all Ibiza island, while the **greedy** and **GRASP** focus on specific regions that contain places with the highest ratings.

6.2.3 Route Evaluation

Algorithms Evaluation

In this section, we verify the routes generated by each algorithm and city. In this assessment, we aim to identify the best algorithm in terms of utility for each city. We also analyze several route quality indicators which are listed below:

1. **# Unique Places:** Shows the tendency of the algorithm to repeat places in the route
2. **# Visted Places:** Shows the full length of a route including the repetition of general places and hotels
3. **Fitness/Utility:** Is measured by the objective function, and can be considered the main indicator of the route quality
4. **Cost:** Shows the mean cost to perform that route
5. **Daily Distance:** Shows the mean length traveled (in meters) in each day.
6. **% Infeasible:** Shows the percentage of infeasible routes generated by each algorithm. Lower is this value, the better.

For these indicators is valid to point out some observations. To the first two indicators, the closer they are the better, given that routes with few repetitions are desired. For the fourth indicator, even though it is not explicitly expressed in our objective function, we consider that is better to generate cheaper routes. The fifth indicator can be interpreted in different ways. Given that real users have different thoughts towards how much they want to walk, longer routes will be better or worse according to the tourist. Nevertheless, if the user walks less and visits more places, there is more efficiency in the generated route. Thus, walking less is better due to the fact that the tourist has a limited time.

Table 6.3 shows the indicators presented above for each algorithm and city. Analyzing this table it is possible to categorize our results in two folds. First, cities where **GA** generates the best results 6.3, second, cities where **GRASP** generates the best results.

Tiradentes and Cannes were the cities that **GA** had the best results when evaluating the fitness. This algorithm fitness is majorly composed of ordinary places, given their poor selection of unique places, as we can see in Section 6.2.2. Given the high sensibility of our problem, where the change of one place can make a route infeasible, the mutation operator had almost no effect on the route, making the generated routes very similar to

the routes initially generated. Nevertheless, due to the selection and crossover operators, it was possible to generate bigger routes, that are as not as diversified as in other algorithms but hold more places. Thus, these two operators show a good performance in constructing the order of the places, nevertheless, it does not select the best places to compose a feasible route.

Also, note that these two are the smallest cities that we are evaluating. As these are small cities and easier to walk, the genetic operators were able to order the places to be visited, even though they are spread across the city, making smaller routes that are more efficient and allow to visit more places, not spending much time walking from one place to another.

In a second scenario, the cities of San Gimignano (SG), Ouro Preto, and Ibiza had the best results with **GRASP** when evaluating the fitness. These three cities can be considered a harder scenario than Tiradentes and Cannes for two main reasons: (*i*) there are more POIs in those cities, (*ii*) these cities have a larger area, and the POIs are spread through the city. The mean daily distance indicator illustrates that. While in Tiradentes and Cannes, the maximum distance is about 5 km, in SG, Ouro Preto, and Ibiza, the users' walk about 8 km daily in the best scenario.

In these cases, simple strategies such as **random** and **greedy** show a poor fitness performance, having a negative average in some cases. As these two algorithms are naive, it fails to properly order the POIs, thus generating routes that are not feasible.

GRASP stands out with good results given the local search performed over the selected solution. As this algorithm start solution is randomly defined as in the **GA** and **clonalg**, the initial solution has usually a poor selection. But, due to the local search, the final solution utility is highly increased by the replacement of places. Also, besides selecting the venue by rate, the algorithm also weighs the price and distance. Hence, in cities where the search space is bigger and can be better explored as these three cities, **GRASP** increases the fitness while reducing the distance walked and the costs.

Lastly, as **GRASP** is considered a sampling algorithm [22], we define the alpha parameter with a very low value, which makes this algorithm select majorly feasible instances. If we evaluate all cities, only in Ouro Preto (which we consider the hardest scenario) **GRASP** has infeasible instances. This happens due to the fact that the initially generated solutions (random) are not guaranteed to be feasible, and in Ouro Preto about 23% of the random instances are infeasible. **GRASP** reduces this value to 11% and increases the overall fitness of the random solutions. Nevertheless, even though **GRASP** presents the best results in many cities, the local search is costly. In tests with real users, it would be hard to provide good solutions in a small computational time.

City	Metrics	Clonalg	Genetic	GRASP	Greedy	Random
Tiradentes	Amount Unique	12.308373	16.922734	17.334195	15.437275	16.807208
	Amount Places	16.481741	24.009287	21.391064	19.952776	21.330301
	Fitness	94.816259	**96.773062**	90.017272	82.702224	83.756540
	Costs	2530.702094	2708.121741	2866.003539	3549.555595	2678.386029
	Mean. Daily Distance	5073.434548	3022.301678	2709.380246	4253.498443	3483.129762
	% Infeasible	0.000000	0.001486	0.000000	0.029560	0.004199
Ouro Preto	Amount Unique	9.879131	12.406743	15.224435	8.214011	11.426656
	Amount Places	14.090086	16.402377	19.350638	12.625609	15.610982
	Fitness	43.585937	55.249544	**74.304946**	13.300255	44.643456
	Costs	1956.174026	1844.068568	2080.544689	2528.121871	1929.814416
	Mean Daily Distance	16992.837467	10293.055100	9464.412100	15107.992458	12724.419673
	% Infeasible	0.275917	0.151589	0.116805	0.519374	0.235886
San Gimini.	Amount Unique	9.476674	11.668228	15.880041	7.264319	10.521059
	Amount Places	13.635064	16.788184	19.875256	11.762362	14.812523
	Fitness	35.610237	63.834090	**84.478462**	-1.603798	35.923901
	Costs	2999.071178	3228.818043	3672.440828	2654.376342	3272.813939
	Mean Daily Distance	17173.018857	8038.381857	8200.815059	16966.310436	12867.971830
	% Infeasible	0.303612	0.024135	0.000000	0.616151	0.294447
Cannes	Amount Unique	12.696958	17.486919	16.055349	12.142749	16.219401
	Amount Places	16.630116	24.114462	19.703195	16.933941	20.357711
	Fitness	85.755028	**95.523991**	84.168510	62.250286	77.572479
	Costs	3319.644453	3462.637457	3874.620829	5692.765121	3535.750553
	Mean Daily Distance	6374.472274	4756.525376	4684.295893	5145.746361	5370.746421
	% Infeasible	0.027413	0.000000	0.000000	0.115793	0.019704
Ibiza	Amount Unique	7.387737	7.286248	7.559630	5.065779	4.773061
	Amount Places	11.488052	11.055141	11.551852	9.024147	8.782709
	Fitness	5.713351	35.256576	**43.423459**	-7.971800	6.700410
	Costs	6565.766063	5935.662581	6729.339061	4317.664742	4972.001837
	Mean Daily Distance	18121.103611	10351.071215	3723.588404	13072.777898	7351.322356
	% Infeasible	0.508566	0.037346	0.000000	0.626145	0.385179

Table 6.3: Results achieved by each heuristic.

In conclusion, answering the **RQ 2: Which heuristic is better suited for solving the Tourist Trip Design Problem?**, it depends. Each algorithm presents different characteristics for each scenario, but **GA** and **GRASP** present the best results in most of the cases.

Clonalg tends to make less efficient routes, having the highest average daily distance covered in all scenarios. However, **Clonalg** is inefficient to convert the distance covered in the selection of places with higher rates. We believe that this is due to their intensive mutation that explores most of the search space, taking users to places that are far away from each other.

The **genetic algorithm** focuses on generating good routes with the places that are available on its feasible solutions. Due to the lack of a repetition constraint, the **genetic algorithm** tends to expand its route without diversification, which leads to a route with fewer unique places when compared to other techniques. However, the **GA** shows good results in small cities, making a good ordering of the places and increasing the efficiency in terms of the number of points covered and the distance covered.

GRASP shows the best performance in several scenarios, due to its resilience to generate infeasible start solutions and the fact that local search increases the utility of

solutions generated. Given that this is an aggregated analysis, **GRASP** stands out even more given that only a few solutions have a negative fitness. All these factors contribute to **GRASP** being the best algorithm in harder scenarios. Still, this algorithm is very costly, which directly impacts its use in real scenarios. This cost is mostly due to the local search which evaluates many options to identify feasible places that maximize the utility.

Lastly, **greedy** and **random** algorithms presented for the scenarios as Tiradentes and Cannes an acceptable answer, given their low computational cost. But in more complex scenarios these algorithms show poor performance in terms of fitness, presenting infeasible routes in more than 50% of the cases

6.2.4 Impact of constraints on the route

In the following analysis, we explore the impact of different constraints on routes generated, showing in detail scenarios where each algorithm performs better. First, we discuss the impact of different parametrization profiles on the routes. Following, we evaluate the generated routes performance under scenarios with mandatory places and appointments. Then, we analyze the impact of POI repetition on the routes. Lastly, we evaluate infeasible cases, assessing the cases of the most difficult scenarios.

Impact of different parametrization profiles

Due to the high variability of tourist profiles, users can have distinct experiences given their constraints. As these profiles can impact the users' experience we compare these profiles' impact on the different scenarios. Table 6.4 presents the mean fitness for each of the profiles evaluated in the cities. Recall that the first profiles (A and B) are more restrictive in terms of money budget while the other profiles are looser.

For the two Brazilian cities (Ouro Preto and Tiradentes), it is possible to verify that the money budget provided is more than enough to provide good results. Table 6.3 corroborates with that by showing that the mean amount of money expended in those cities is very close to the minimum amount considered in the profiles (2500). In this scenario, the daily walk budget is the one that most impacts the results. However, increasing the daily distance willing to be covered produces no effect in this scenario. While in Tiradentes the POIs are clustered near the city center, in Ouro Preto the POIs are scattered throughout all town, forming several groups. Thus, even though the distance willing to be covered increased, the speed that users moved from one location to another

was still the same. Hence, in Ouro Preto is better for tourists to stay in one of these clusters than move to faraway places.

City	Profile	Clonalg	Genetic	GRASP	Greedy	Random
Tiradentes	A	**92.42854**	92.418015	89.25969	69.443907	83.32768
	B	94.342156	**94.820783**	90.087321	73.17247	83.169617
	C	96.673572	**99.59273**	90.433259	90.912579	84.639805
	D	95.134736	**98.016287**	89.943181	83.997584	83.620664
	E	95.50323	**99.015021**	90.357863	94.041671	84.025009
Ouro Preto	A	43.468025	52.667993	**73.332366**	31.486657	46.367182
	B	35.551276	55.153533	**75.044577**	-15.048929	37.593487
	C	55.917492	58.07469	**75.042358**	33.770979	54.744676
	D	47.452635	53.665581	**73.133714**	31.734854	47.702454
	E	35.483475	56.689576	**74.96949**	-15.139253	36.804468
San Gimini.	A	19.582584	52.374826	**76.16189**	-9.17485	20.406295
	B	30.720421	60.771001	**81.711711**	3.864972	30.42681
	C	38.888689	67.065188	**87.213389**	-0.88968	36.65346
	D	30.122903	63.792014	**85.714884**	-6.994491	33.876209
	E	58.30154	75.063056	**91.547494**	5.195844	58.537052
Cannes	A	59.492331	**80.752526**	76.779567	34.892966	57.735125
	B	83.100285	**95.051469**	81.185221	43.784613	72.187747
	C	94.260153	**98.632286**	87.305156	68.635708	85.551195
	D	91.192676	**95.605382**	86.108306	68.905425	83.766893
	E	100.873132	**107.469136**	89.381646	95.397472	88.353971
Ibiza	A	7.506457	24.699154	**34.911007**	16.622047	7.111359
	B	7.743911	26.929396	**35.251528**	-5.577616	6.281348
	C	12.177483	37.935884	**44.231836**	17.313976	18.817271
	D	-1.223041	41.149692	**52.070052**	-16.623143	11.768818
	E	2.337238	45.480226	**50.652874**	-24.889985	-10.413799

Table 6.4: Impact of different parametrization profiles on the generated routes.

		Min	1st Quart.	2nd Quart.	3rd Quart.	Max	Mean	Std. Dev
Tira.	Min. Cost	0.0	15.0	60.0	265.62	637.0	136.91	133.24
	Max. Cost	0.0	50.0	150.0	458.22	999.0	284.39	229.11
OP	Min. Cost	0.0	9.0	18.0	196.75	600.0	92.52	119.11
	Max. Cost	0.0	48.0	144.0	343.5	955.0	198.56	184.19
SG	Min. Cost	0.0	118.0	118.0	502.5	859.46	268.61	229.49
	Max. Cost	0.0	254.0	558.0	624.89	1586.34	494.74	304.03
Can.	Min. Cost	0.0	58.0	116.0	116.0	572.0	141.08	110.51
	Max. Cost	0.0	182.0	546.0	546.0	2654.62	471.97	262.16
Ibi.	Min. Cost	0.0	85.0	170.0	170.0	481.0	143.15	79.21
	Max. Cost	0.0	622.0	1866.0	1866.0	2488.0	1366.99	744.34

Table 6.5: Price statistics.

For the European cities, it is possible to notice a different pattern. Given the low value of the Brazilian currency, the price of POIs in those cities is very high as we can see in Table 6.5. In this case, the change of users parameters has a big impact on the results. With exception of Ibiza, by increasing the daily walked distance it is possible to notice an increase in the fitness for all algorithms. In Cannes and SG fitness is highly related to user movement. As we can see, from the profile C to D, where the amount of money is increased but the user movements are restricted, there is a decrease in the fitness,

showing how dependent are the solutions from the distance. In Cannes, this decrease is more subtle given that this is a small town.

Ibiza presents different characteristics from the other cities. This city combines the high prices of the European towns with the POIs scattered through the map, as in Ouro Preto. In the first two evaluation profiles, A and B, the overall results are very poor due to these characteristics. Even if the daily distance is increased, the user still has no budget to visit many places. Table 6.3 shows that the average cost of a trip to Ibiza is at least 5000 reais. In the A and B scenarios, the budget available is smaller than the average. For the profile C, it is possible to verify an increase of fitness in all algorithms. **GA** and **GRASP** take advantage of the increased money budget and organize better routes, increasing the mean fitness. The other three heuristics have a poor performance in the last two profiles due to the expansion of the search space which makes available more places to be in a route. However, these techniques are not capable of properly ordering the routes, thus, generating infeasible solutions.

In conclusion, in Brazilian cities, it is possible to provide good routes with more economic profiles. In Ouro Preto, to generate better routes, it would be necessary to increase the transportation speed. Thus, it would be possible to visit far places spending less time moving from one place to another. Through these evaluations, it became more clear that naive strategies do not perform well even in extreme scenarios. While the **random** approach achieves better results by exploring more of the search space, this technique shows poor performance in more difficult scenarios like Ibiza and Ouro Preto.

Mandatory and Appointments

In this evaluation, we analyze the performance of the algorithms under the perspective of the mandatory and appointments constraints. The mandatory constraint, proposed in the work of [44], enables users to add obligatory places to be visited. The appointments constraint enables users to define places to visit with a fixed start and end hour.

To perform such evaluation, we select from 0 to 2 mandatory/appointments POIs per route. The places selected by us are in different regions of the cities. Also, we define that the POIs in the appointment setting should be visited during the afternoon period $(15:00 - 17:00)$ of any day.

Table 6.6 presents the fitness of the algorithm in different mandatory and appointments scenarios. Note that in most cases the algorithms are able to attend to both constraints. Overall, the naive approaches, **GA** and **Clonalg** present better results for the mandatory constraints than for the appointment constraint. This is due to the fact that mandatory places can be allocated at any time in the trip, while appointments have fixed start and end times. As the appointments are fixed in the solution and the POIs that will compose the route have been already selected, the places around the assignment do not minimize the distance between the last venue and the appointment, thus, generating

routes that tend to walk more and visit fewer places, decreasing the overall fitness. In this scenario, **GRASP** will be less affected due to the local search, which enables the selection of places around the appointments that decreases the distance between the last place and the assignment.

Finally, we conclude that **GRASP** performs better when allocating the mandatory places and appointments in more difficult cases. In cities such as Tiradentes and Cannes, the **GA** has a good performance in the allocation of mandatory places, however, the allocation of appointments has a negative effect on the route fitness, generating poor results when compared to the fitness of routes without appointments.

	# Mandatory	# Appointment	Clonalg	Genetic	GRASP	Greedy	Random
Tiradentes	0	0	110.664946	**135.580834**	89.831451	94.518527	96.500076
	0	1	96.037417	**104.916680**	87.353072	84.529189	85.950139
	0	2	102.848141	**107.870824**	82.926403	90.386309	91.821036
	1	0	104.298811	**114.032560**	94.463977	91.076884	93.238739
	1	1	88.017187	**91.480626**	91.181853	78.215928	79.503445
	1	2	78.295760	75.261381	**85.871124**	72.311905	72.625140
	2	0	94.738631	**102.398787**	98.357508	86.854177	86.321133
	2	1	87.966404	93.498006	**97.000034**	84.166566	84.143483
	2	2	**97.437349**	72.903374	84.708010	68.621211	68.558705
Ouro Preto	0	0	76.204324	74.602620	**88.234572**	46.738759	72.378063
	0	1	31.574628	45.006107	**72.608072**	-2.212039	31.747788
	0	2	10.572809	27.417087	**52.722601**	-14.26069	-7.231116
	1	0	66.979991	72.410836	**86.009360**	23.739517	65.357767
	1	1	50.067741	67.752275	**85.226235**	8.702525	56.486042
	1	2	12.045786	24.657818	**49.728812**	-8.502674	16.327432
	2	0	68.482688	75.995514	**87.499500**	23.248138	67.325434
	2	1	46.207281	72.986839	**91.924321**	9.383152	49.808685
	2	2	17.782386	33.424924	**70.090708**	-4.329278	15.823767
San Giminignano	0	0	45.444002	65.424502	**83.414270**	-0.940543	43.801437
	0	1	30.307076	64.435195	**83.261693**	-15.593241	33.683352
	0	2	43.439998	75.606773	**82.567668**	14.743342	34.746051
	1	0	34.466482	64.191335	**88.176005**	4.904539	34.269505
	1	1	30.412335	62.526407	**88.409039**	2.590955	37.763291
	1	2	41.355510	65.371021	**84.623197**	30.309926	46.671470
	2	0	41.066960	65.048367	**90.310231**	13.475187	35.139177
	2	1	21.396767	51.898670	**89.327427**	1.375460	26.511253
	2	2	34.259768	46.908721	**82.496408**	27.572302	21.301232
Cannes	0	0	91.532815	**103.106131**	88.699823	66.508929	79.501330
	0	1	87.689739	**96.571750**	85.416813	71.308851	78.004804
	0	2	80.632494	**87.122281**	76.427381	71.962504	73.357151
	1	0	88.101825	**101.319446**	89.187235	58.969587	80.223734
	1	1	84.327458	**92.928029**	83.177896	52.815098	76.237566
	1	2	73.004898	**80.395855**	72.538974	63.985745	69.294327
	2	0	88.321654	**100.313311**	88.113253	43.635737	82.992813
	2	1	82.305951	**92.197919**	82.506048	39.305902	79.412969
	2	2	76.741695	**79.442455**	73.367452	44.878472	74.006995
Ibiza	0	0	5.819541	29.392690	**38.571313**	3.034296	17.206308
	0	1	10.377486	35.184641	**38.072524**	7.613413	18.211744
	0	2	2.374472	36.995004	**37.989862**	5.399999	18.274696
	1	0	10.816626	32.981692	**44.241808**	-7.424231	10.345412
	1	1	7.006774	34.738549	**44.023145**	-10.007905	11.015540
	1	2	-2.452274	36.916545	**44.044328**	-10.305633	9.428715
	2	0	14.656988	39.324791	**48.204600**	-19.454371	-6.481651
	2	1	5.420071	37.089805	**47.938874**	-21.896976	-11.469650
	2	2	-2.009762	37.694074	**47.870732**	-19.788727	-6.851897

Table 6.6: Study of mandatory places and appointments on the routes generated.

Repetition Constraint

Recall that the definition of the TTDP presented in Chapter 4, does not define a constraint that restricts the number of times that a user can visit the same place on different days. Thus, a "perfect route" in terms of fitness can be composed by the repetition of places on different days. This does not occur, due to the fact that the algorithms used in this study are stochastic and each place can have different attributes per day in working hours and price. Besides, as we will see in Chapter 7, users usually are looking for diversity in their routes.

City	Algorithm	Amount Unique	Amount Places	Fitness	Costs	Mean Daily Distance	% Infeasible
Tiradentes	Genetic	18.88963	22.886667	91.250148	1936.394062	3221.51	0.00074
Ouro Preto	GRASP	15.224435	19.330499	74.172549	1490.420412	6763.17	0.0
San Gimignano	GRASP	15.879699	19.77136	83.855425	3111.107955	7919.18	0.0
Cannes	Genetic	17.481142	21.459905	84.071877	3097.124854	4919.21	0.0
Ibiza	GRASP	7.55963	11.550741	43.37798	5822.539002	3714.51	0.0

Table 6.7: Best algorithm per city results without place repetition.

It is noteworthy that this constraint is not present in our constraints set due to specific scenarios. For example, consider a vegan tourist in a town with only one vegan restaurant. In this case, the user would prefer to visit the same restaurant several times than visit one that does not attempt their needs. Nevertheless, this is an extreme case that will only happen to a few users. Hence, letting the users choose if they want to revisit a place is the best option.

Given this facts, we introduce to our model a **repetition constraint** that restricts the repetition of places in a travel. To verify the effectiveness of these algorithms with this new constraint, we evaluate the best algorithm of each city with similar indicators as before. Table 6.7 shows the results achieved in each city. Note that the results achieved do not decrease much of the fitness.

In the cities where **GRASP** had the best results, Ouro Preto, San Gimignano, and Ibiza, the results were not so different than the ones presented in Table 6.3. This is due to the fact that **GRASP** maintains its overall quality by reducing very little in fitness. In this case, the local search suggests the second-best place to visit, in cases where the first was already visited. Besides, **GRASP** routes are initially generated by a **random algorithm**, in which we remove the repetitions of the generated route. Hence, the routes generated are smaller in terms of length.

For the **GA** we observed a tendency of reducing the general size of the route, given that the algorithm tends to visit places that are farther. Thus, **GA** increases its search space to find new places, consequently, it increases the daily distance walked. However, the costs are reduced in both Tiradentes and Cannes.

In conclusion, we observe that both the best algorithms, **GA** and **GRASP**, are able to generate good routes while respecting the repetition constraint. In the case of **GA**, the techniques tend to explore more the search place aiming to find closer places that were not visited before. In this case, the algorithm makes the tourist walk more to find places that fulfill all constraints. For **GRASP**, the local search enables to walk less while reducing very few of the route fitness.

Infeasible Cases

In this evaluation, we analyze the constraints that made the results infeasible and how it is possible to avoid such results. To make such an analysis we categorize our results in 5 constraints: appointment, budget, daily distance, mandatory, and others. The last group

represents all other constraints presented in the problem formulation. It is noteworthy that a route can disrespect more than one constraint.

Table 6.8 shows the percentage in which each constraint is disrespected. Note that the daily distance and the mandatory constraint are the most difficult to meet. For naive algorithms as **greedy** and **random**, these constraints are violated even in simpler scenarios as Tiradentes and Cannes. For the daily distance constraint, the restriction is disrespected due to the fact these algorithms do not perform any enhancement operation on the route. Thus, it does not properly order the places, forming longer paths that do not satisfy the distance constraint. It is noteworthy that in these cities the **Greedy** algorithm presents the worst performance, showing 2% and 11% of infeasible instances in Tiradentes and Cannes respectively. As these are small cities the POIs are closer, making the impact of the distance constraint smaller.

Note that in more difficult cases, Ouro Preto, San Gimignano, and Ibiza, the daily distance is more violated than other constraints by all algorithms. Illustrating that, Ouro Preto is the only scenario where **GRASP** generates infeasible results.

The mandatory constraint is also disrespected in several cities. For the naive algorithms, this occurs more often due to the fact that their initial solutions are shuffled with all POIs that were selected to visit. As the constructing heuristic does not guarantee that all places will be allocated, some places are not in the final route. This problem does not occur with the POIs in the appointment set due to the fact that these are already fixed in the route.

As future work, we must evaluate evolutive operations focused on properly ordering the places, avoiding the daily distance constraint. The work of [44] tackles this problem by greedily sorting the places according to their distances in their route construction heuristic. Nevertheless, is noteworthy that this strategy cannot work in all cases due to the working hours window of each POI.

Lastly, answering **RQ3:What is the impact of the constraints in the routes generated?** the daily distance constraint is the one with the most impact on the routes and the most difficult to comply with. By not meeting such constraints the heuristic does not generate compact routes (users walk more and visit less), hence, the users tend to spend more time moving from one place to another. By visiting fewer places, consequently, the fitness of the route is decreased.

	Constraint	Clonalg	Genetic	GRASP	Greedy	Random
Tiradentes	Appointment	0.0	0.0	0.0	0.024390	0.166667
	Budget	0.0	0.0	0.0	0.000000	0.000000
	Daily Distance	0.0	0.0	0.0	0.219512	0.666667
	Mandatory	0.0	1.0	0.0	0.475610	0.166667
	Others	0.0	0.0	0.0	0.487805	0.166667
Ouro Preto	Appointment	0.463768	0.026316	0.0	0.254675	0.280899
	Budget	0.000000	0.000000	0.0	0.000000	0.000000
	Daily Distance	0.392915	0.776316	1.0	0.569012	0.474719
	Mandatory	0.003221	0.013158	0.0	0.008905	0.033708
	Others	0.141707	0.171053	0.0	0.170971	0.238764
San Gimini.	Appointment	0.117720	0.298507	0.0	0.144919	0.112658
	Budget	0.045849	0.000000	0.0	0.000000	0.000000
	Daily Distance	0.677819	0.417910	0.0	0.532910	0.635443
	Mandatory	0.013631	0.104478	0.0	0.004619	0.046835
	Others	0.138786	0.283582	0.0	0.310624	0.187342
Cannes	Appointment	0.000000	0.0	0.0	0.019672	0.096154
	Budget	0.547945	0.0	0.0	0.026230	0.000000
	Daily Distance	0.287671	0.0	0.0	0.324590	0.153846
	Mandatory	0.150685	0.0	0.0	0.403278	0.538462
	Others	0.013699	0.0	0.0	0.226230	0.211538
Ibiza	Appointment	0.112145	0.058824	0.0	0.000000	0.000000
	Budget	0.192376	0.241176	0.0	0.136968	0.180361
	Daily Distance	0.643174	0.600000	0.0	0.853723	0.782565
	Mandatory	0.031028	0.0647	0.0	0.00399	0.002004
	Others	0.021277	0.035294	0.0	0.005319	0.035070

Table 6.8: Percentage of infeasible instances.

6.3 Summary of Results

In this chapter, we presented the results achieved through the proposed methodology. First, we describe the results of the Recommender Systems for each of the cities. In those results, it was possible to verify that **SVD** had the best results in all the cities. Besides that, we also evaluate the behavior of LBSN recommenders in TSN datasets, our conclusion is that most of the data do not have the proper granularity for many of the algorithms, leading to poor results.

Besides, we also evaluate the results of heuristics, evaluating the best fitness cases and the constraints that most impact the generated routes. In this step, we could verify that the **Genetic Algorithm** (Tiradentes and Cannes) and **GRASP** (Ouro Preto, SG, Ibiza) are the heuristics that perform better in the cities we evaluated. In this scenario, we could evaluate that the **GA** tends to perform better in small cities where the places are closer to each other, while **GRASP** tends to get better results in more difficult scenarios due to the local search.

Chapter 7

Planet Caravan App.

Chapter 3 describes the literature of several works that model real users' constraints while traveling. In some of these works, authors evaluate their approaches with real users through an application (interface) that enables tourists to easily generate routes [74, 28, 12]. However, even though there are works that present an application, most of them are limited and do not allow users to fully experience constraints and personalization needed in one travel.

In Planet Caravan we tackle this problem by proposing an interface and an architecture that enables travelers to plan a personalized route. Figure 7.1 gives an overview of the proposed architecture.

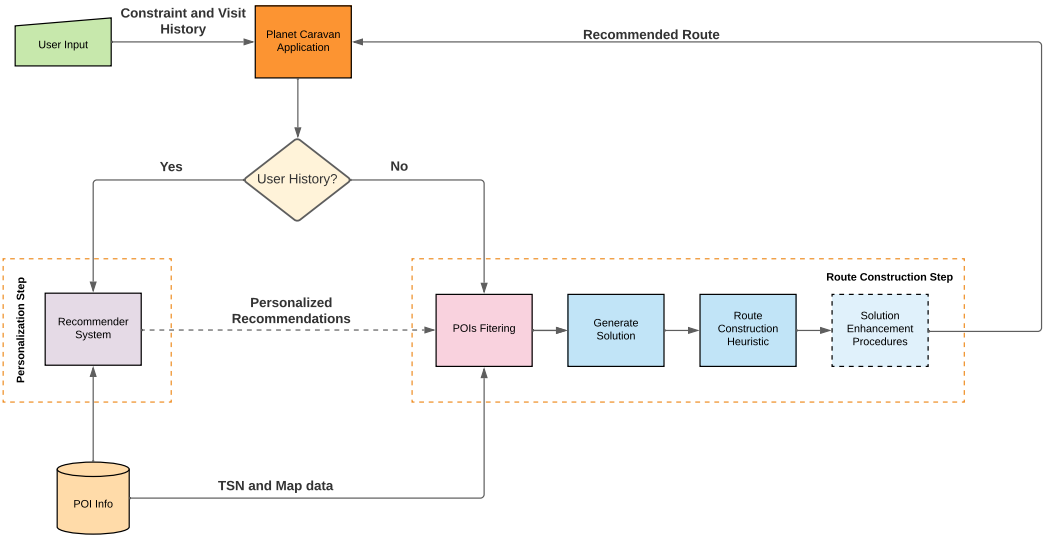


Figure 7.1: Overview of the complete architecture of Planet Caravan.

To provide personalized routes the architecture is highly dependent on the user's inputs. There are mandatory and optional parameters for the users. As mandatory inputs, there are the traveling dates, the money budget, the number of kilometers that the users plan to walk each day, and the activity hours. As optional inputs, there are mandatory places, fixed mandatory places, and users' historic ratings in the city. In cases where the

users inputs their past visits, a recommender system is trained and used to predict the places that were not visited yet by the user. In cases where no historic is provided, the average rating of each place is used as a predicted rating.

Besides the users' inputs, a database with information of each place is used, providing the locations restrictions. As described in Chapter 5, this data was collected from TripAdvisor, and each place has its geo-location, open and closing time, working days, and category. To measure the actual distance between the places was used the OpenStreetMaps API. From OpenStreetMaps we also identify the streets between the POIs, which were plotted using Folium library [1]. The front-end and design of the interface were implemented using Streamlit [2].

As intelligence algorithms responsible for personalization and route constructing were used **SVD**, for personalization, **GA** and **clonalg** as route heuristic. These algorithms were selected due to their good results for the selected cities, and the fact that they are less computational expensive than **GRASP**. In the heuristic step, **clonalg** was used in the city of Tiradentes, when the amount of appointments was bigger than 2 and the amount of mandatory was at least 1, which is the scenario that this algorithm performs better than the **GA**.

Figure 7.2 gives an overview of the proposed interface. Figure 7.3 presents all parameters available in the sidebar menu. On the right, it is displayed the menu. We enumerate each item according to what appears on the menu:

1. Allows users to change the city
2. Allows users to change the mandatory parameters. The last item on the second menu allows users to add mandatory places to be visited
3. Allows users to change the map visualization style
4. Allows users to add appointments in the city
5. Allows users to add places that were previously visited adding a rating for this place (history insertion)
6. Allows users to update files in two cases: First, visualization of previously generate routes. Second, allows uploading the user history
7. Allow users to change the preferences on the route visualization
8. Allow users to know more info about the generated route
9. Allow users to save the route as a CSV file or send it by e-mail [3]

[1] http://python-visualization.github.io/folium/
[2] https://streamlit.io/
[3] The three last options are only available after the route is generated.

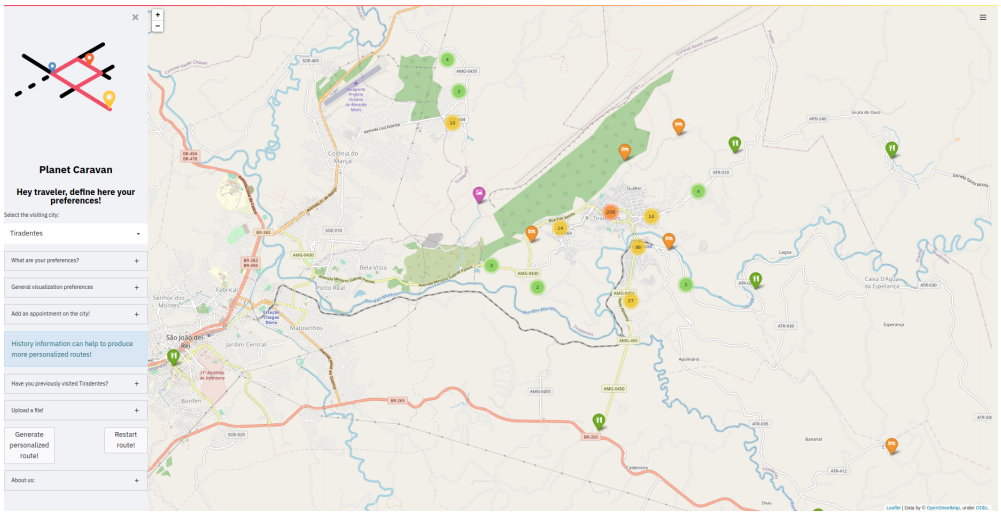


Figure 7.2: Interface Overview.

To evaluate the usability of our proposal in real scenarios, we test it with real users. To capture the users' opinions we provide a survey in which they evaluated our application. In our analysis, we selected Tiradentes and Ouro Preto as study cases, due to the fact that the survey was shared with Brazilian users. Thus, if the users visited one of the cities, the evaluation of the route could be based on their previous experience in the selected towns.

For our application was released for the users two different versions, named version A and version B. The version A is the one described above and in the version B, some improvements were implemented according to the users' answers on the survey. The versions were released in different periods of time. In total 56 different users evaluated our framework, where 28 evaluated the first version of the platform, and 28 evaluated the second version. We did not set for the users specific tasks, instead, it was only asked for them to interact with the platform and answer the questions.

7.1 Versions

In this section, we focus on describing the differences between the two systems that we evaluated. Version A was our first release and was evaluated by 28 users. As the users evaluated the A version and leave their comments in the survey, it was possible to notice their difficulties with the platform. Users complained of not knowing where the route would start and end, and they did not know the order to visit the places. Besides that, users complained about not knowing if the route was being generated, even though a progress bar is located at the side menu. Lastly, when an infeasible route was generated,

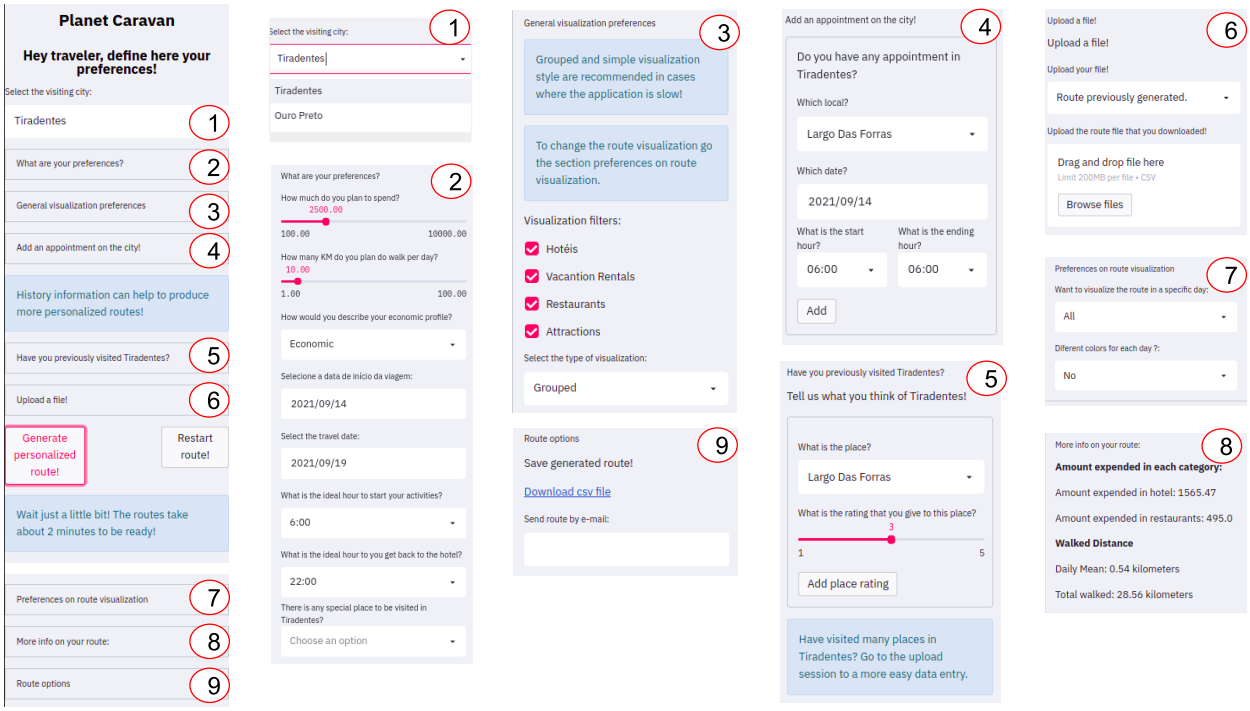


Figure 7.3: Sidebar of customizable options for the users.

a simple message was displayed that did not specify which constraint was disrespected.

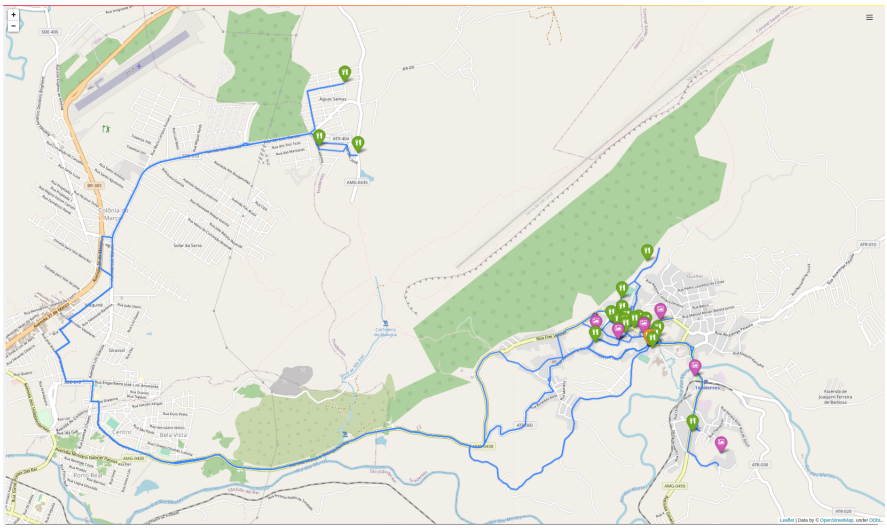
Given the problems pointed out by the users, in the B version we tackle these issues by showing the order of the places to be visited, a progress bar in the middle of the screen instead of the sidebar. Also, we specified to the users the constraint that made their route infeasible.

Figures 7.4a, 7.4b shows the routes generated by interface A and interface B for the same city. It is possible to notice that in interface B the POIs are enumerated and the hotel is more highlighted.

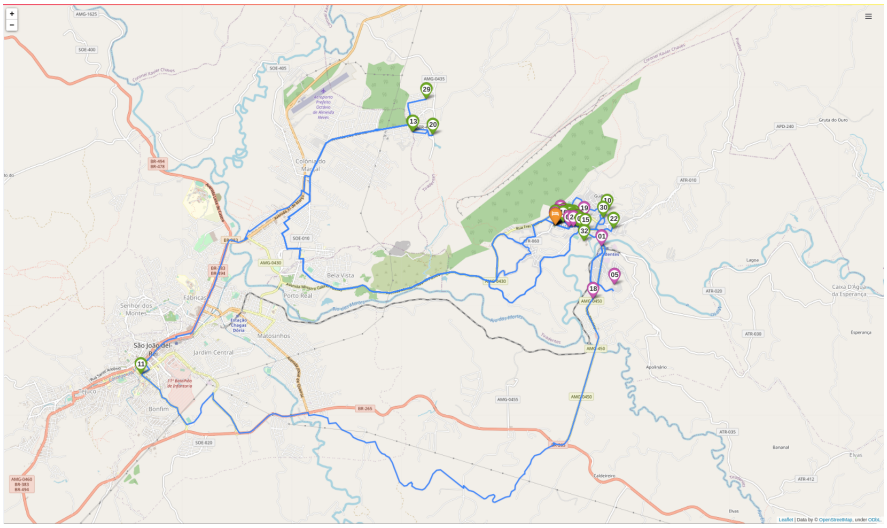
7.2 Survey Results

The survey was composed of 17 questions that aimed to characterize the user profile, identify the opinions about the route, and analyze the design and usability of the platform. The questions are presented in the list below:

1. What is your age?
2. What is your city?
3. Have you visited one of the cities available in the system?



(a) Version A - Interface



(b) Version B - Interface

4. What is your profession?
5. What do you most value when you are traveling?
6. How much do you agree with the phrase "Is better to visit new places that I do not know that will please me, that to visit that I already visited"
7. Would you like to have pre-defined hours to visit restaurants?
8. The generated route respected all your constraints?
9. In case of no in the last question, what were the constraints not respected?
10. What would you improve on your route?
11. How would you rate the route generated?
12. Did you have any problem using the system?

13. Something in the platform confused you?

14. How would you evaluate the platform design?

15. What would be more useful to you a mobile application or web?

16. What functionality do you miss in the platform?

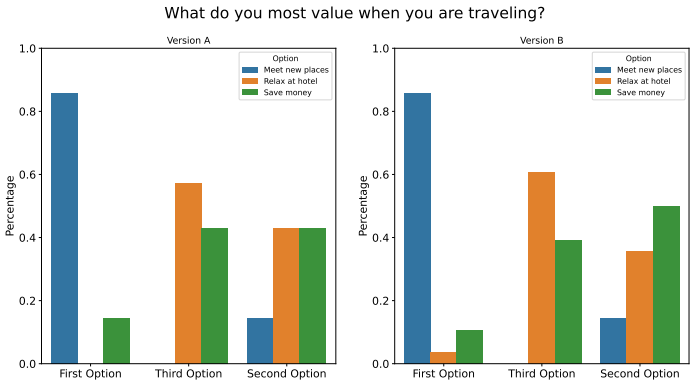
The first 7 questions aimed to make a profile of the users that interacted with the platform. In both versions, the age of the users was majorly between 18 and 30 years (about 74%), and in most of the cases, the users worked directly in the technology field as developers, data scientists, and computer science students, while about 18% worked in other areas. From the 56 persons that evaluated the framework only 9% were from outside the state of Minas Gerais, which is the state of the cities evaluated. In the survey about 90% of the users had already visited the cities, which made their opinion about the route more accurate given past experience in those cities.

Questions 5, 6, and 7 aim to understand the traveling preferences of the users. The fifth question aims to understand what the users most value while traveling, giving the traveler three options to be sorted. Figure 7.4a shows the options available and users' answers to the fifth question. In both versions, the users valued more to meet new places, followed by saving money and lastly relaxing at the hotel. These results show that users are more likely to avoid idle time, hence doing as many activities as possible. Complementary, the answers to the sixth question presented in Figure 7.4b, shows that besides having a complete schedule of activities users prefer to visit new places, even though these places are not between the best places to be visited in the city.

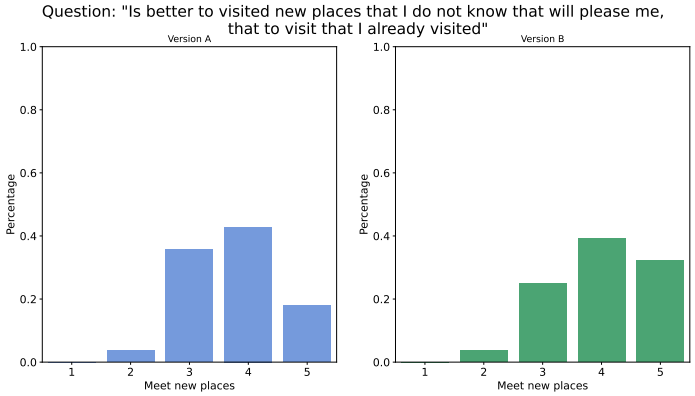
Question seven verifies if users would like to have pre-defined hours to visit restaurants and food-related POIs. Our results show in both versions that about 78% of the users would like to have pre-defined hours to visit restaurants. These results corroborate with the works of [74, 28, 12, 78] that have previously defined constraints related to meal-time windows. It is noteworthy that this could be easily adapted to our model through the appointments constraints.

Then, users were asked about the quality of the generated routes. In both versions, about 85% of the users did not report any constraint violation with the generated route. In the scenarios where the system did not return a feasible route, the two main reasons reported by users were an application crash in the A version, and infeasible mandatory constraint in the B version.

In the first case, the application crashed due to an implementation error. In the second case, users reported that their appointments and mandatory places were not present in the route. Evaluating these users' routes it was possible to verify that users had defined mandatory places and appointments that are too far away from the other POIs. Thus, to satisfy one constraint, the other one was infeasible. This problem can be avoided



(a) Users tends to want to visit more places than have idle time on the hotel or save money.



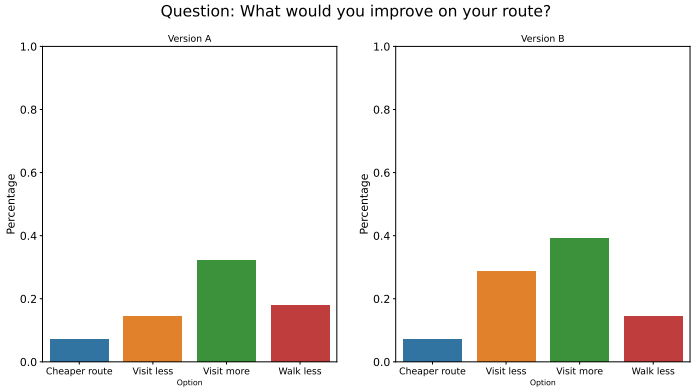
(b) A Likert scale was used to understand the users' preferences of meeting new places.

by increasing the number of daily kilometers to walk or picking mandatory places and appointments that are closer to regions with hotels.

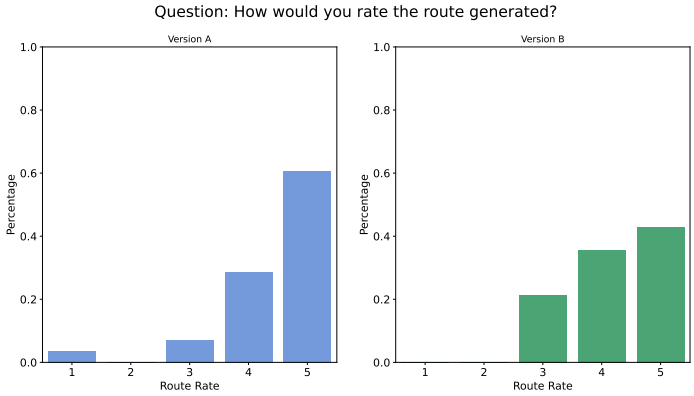
The tenth question is related to what the user would improve on the route and in the system. For this question we gave 4 options to the users (*i*) Make a cheaper route, (*ii*) Visit more places, (*iii*) Visit fewer places, (*iv*) Walk less. It is noteworthy that users could select more than one choice. In version A about 43% of the users answered that no improvements were needed in the route, while in version B about 78% of the users wanted at least one improvement in the route. Figure 7.4a shows the improvements that users would prefer. As we can see, most of the users would like to visit more places per day, while others would like to visit fewer places per day. Thus, as users showed different patterns towards the number of places visited per day, it would be from the user's interest to define the maximum amount of places that can be visited in one day.

The eleventh question asks the users to evaluate the quality of the route in a rating from 1 to 5, where 1 if the users did not like the route that was generated and 5 the user like very much the route. Figure 7.4b shows the rating distribution in each of the versions. As we can see, in version A about 85% of the routes were given a rating of 4 or higher, while in version B about 78% of the users gave a rating of 4 or higher. It is noteworthy

in the Figure that in version A we had a route with a rating equal to 1. Following the comments left by the user that gave such a rating, it was possible to verify that the user had an error that made unable the tourist complete the route. It is noteworthy however that only this user had this problem and this was fixed in the B version of the system.



(a) Improvements that users would make in their routes.

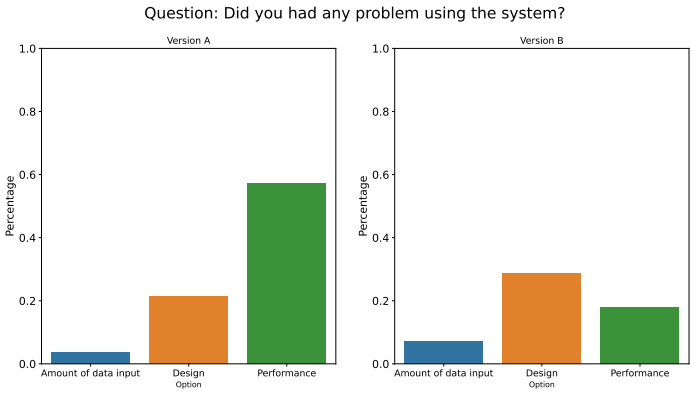


(b) Users evaluation of the generated routes.

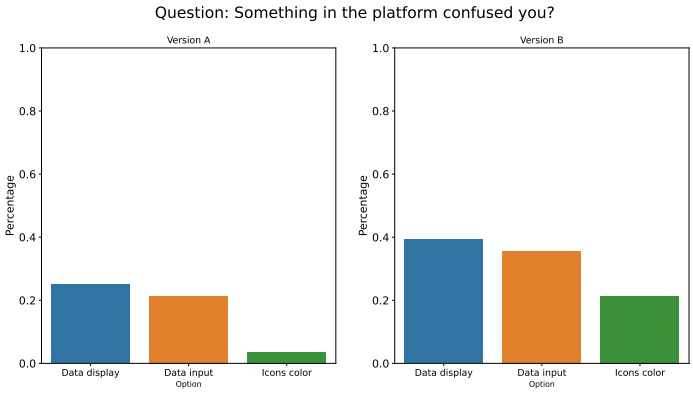
From the twelfth until the last question we focus on understanding the usability and the problems of our platform. The twelfth question asks if the users had any problem using the platform. We gave as options performance, interface, and the amount of data input. In the question, we had different results for version A and version B. While in version A, only 29% of the users did have any consideration to make about the platform, 50% of the version B users did not have any complaint. Complementary, Figure 7.4a shows that in version A, users pointed out that the performance was their biggest issue on the platform. In version B, we updated some aspects of the system design. This had a direct impact on the users' answers, given that the design is the biggest issue for the users on the B version.

As we did not set a list of tasks for the users and only gave them an overview of our project, the thirteenth question aim to understand if the user was confused at any point in the platform. On this question, we gave as options for the users the data display, data input, and the color of the icons. Figure 7.4b shows that users in versions A and B

had the same tendency, showing more confusion towards the way the data was displayed and how the data was inputted on the platform.



(a) Problems that users had with the system.



(b) Topics that most confused the users of the interface.

The fourteenth question asks the users to give a rate to the design of the application. Figure 7.4 shows that users on the B version were more critical about the design of the platform, having a wide range of grading when compared to version A users.

Focusing on future work, the fifteenth question asles about the user preferences towards the best platform to bare our system, mobile or web. In this scenario, we could verify that joining both versions, about 78% of the users would prefer a mobile version of Planet Caravan. We believe a mobile version would assist users to better keep up with the route given that could verify the paths and the places to visit at any point, while in the web version they can only verify once and send the route to the e-mail to verify later, without the assistance of an interface.

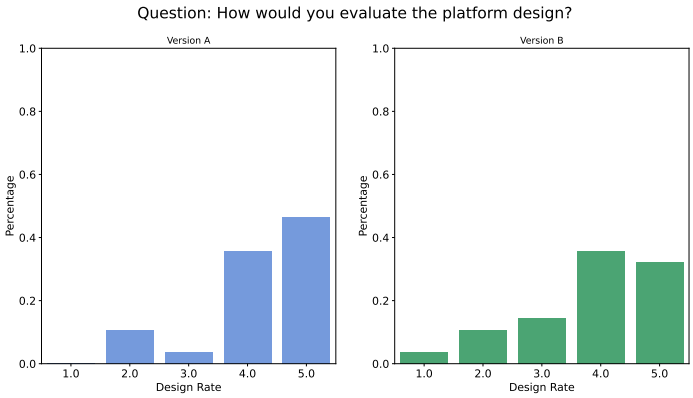


Figure 7.4: Users evaluation on the interface design

In conclusion, those two versions showed that Planet Caravan can be used in real scenarios. In 85% of the evaluations, no constraint violation was found. Besides that, our routes were well evaluated in both versions. However, through the users' comments, it was possible to notice some potential improvements in the performance and interface, which we intended to tackle as future work.

Chapter 8

Conclusions and Future Work

In this chapter, we summarize the main achievements of this thesis and present a discussion on future works. In section 8.1 we make a summary of our work and point out the differences between Planet Caravan and other works in the literature. Next, we answer each of the research questions presenting the obtained results. Finally, in section 8.3, we present future perspectives for our work.

8.1 Comparison

In this work, we propose new modeling for the Orienteering Problem, also known as, Tourist Trip Design Problem. This problem consists of a combination of the knapsack and the traveling salesman problem and has been widely studied in a tourism context. In this scenario, the objective is to find a route for maximizing users' preferences, while respecting environment, place, and user constraints.

In this work we propose a new definition for this problem, introducing a new constraint that enables users to allocate appointments and plan a trip around their schedule. We test our modeling in 5 different cities around the globe and evaluate the performance of 5 different heuristics for this problem.

Our results show that the **Genetic Algorithm** and **GRASP** have the best performance in different cities. Nevertheless, **GRASP** is prohibitive in real-time scenarios due to its high computational cost. Lastly, we also introduce an application for Planet Caravan. The study with real users shows promising results for the usage in real-time scenarios.

8.2 Research Questions

We guide our work by defining the following research questions.

Which type of Recommender System is better suited for defining the user ratings for each place in the TripAdvisor dataset? For this question, we verified that **SVD** is the model that shows the best performance between the algorithms evaluated. Also, results show that LBSN recommender systems do not usually work on TSN datasets.

Which heuristic is better suited for solving the Tourist Trip Design Problem? In this evaluation, it was possible to verify that the **Genetic Algorithm** and **GRASP** were the heuristics with the best fitness results in the scenarios that we evaluated. The **GA** shows promising performance for Tiradentes and Cannes. Nevertheless, it is noteworthy that this algorithm usually does not have a good place selection. The good results are usually achieved by the genetic operations providing bigger routes than other heuristics, which consequently impacts the fitness results.

In the most difficult scenarios **GRASP** presented the best result, was robust against infeasible cases, and had the smallest rate of disrespected constraints between all heuristics. However, due to the local search, this heuristic is computationally costly. This way, without good implementation, **GRASP** cannot be used in real-time scenarios. Hence, we believe that the best heuristic would be a merge between these **GA** and **GRASP**, balancing efficiency and quality in the generated solutions.

What is the impact of the constraints in the routes generated? In this evaluation, our results show that the daily distance constraint is the one with the most impact on the routes. To respect the daily distance the algorithms must generate compact routes, and the results point out that this is one of the biggest difficulties of the heuristics. The naive approaches do not use any kind of enhancement technique to properly order the places, hence, the user tends to walk more. In this scenario, only the **GA** and **GRASP** are able to generate more compact routes. Nevertheless, in more difficult cases, **GA** tends to reduce the overall size of the route, consequently having a smaller fitness.

8.3 Future Work

As for future work, first, we focus on better exploring the data collected. Besides having all information about the users in a specific city, the data presents the complete users' history in TripAdvisor. This way, we could use out-of-town users' data to enhance the quality of the models. Also, we only use memory and model-based collaborative filtering approaches in this work. We believe that more robust techniques would take more advantage of all the data available.

In the heuristic step, we can enhance the quality of the heuristics in several ways. First, finding the best parameters for each heuristic can lead us to better results. We also would like to test these heuristics as a multi-objective approach, focusing first on solving the knapsack problem by selecting good places and then the travel salesman problem finding good routes for the problem. Also, introduce different transportation types in our model, selecting the best transportation type in each situation.

Lastly, we would like to improve the quality of the proposed interface. The results point to several aspects in the design that make the user confused. Besides that, users also pointed out that a mobile application would be better for their needs than a web platform.

Bibliography

[1] Muhammad Afzaal and Muhammad Usman. A novel framework for aspect-based opinion classification for tourist places. In *2015 Tenth International Conference on Digital Information Management (ICDIM)*, pages 1–9. IEEE, 2015.

[2] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

[3] Francisco Amaral, Teresa Tiago, Flávio Tiago, and Androniki Kavoura. Comentários no tripadvisor: Do que falam os turistas? *Dos Algarves: A Multidisciplinary e-Journal*, 2(26):47–67, 2017.

[4] Rajesh Kumar Arora. *Optimization: algorithms and applications*. Chapman and Hall/CRC, 2019.

[5] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. Recommendations in location-based social networks: a survey. *GeoInformatica*, 19(3):525–565, 2015.

[6] Punam Bedi, Sumit Kumar Agarwal, Vinita Jindal, et al. Marst: Multi-agent recommender system for e-tourism using reputation based collaborative filtering. In *International Workshop on Databases in Networked Information Systems*, pages 189–201. Springer, 2014.

[7] Alejandro Bellogín and Arjen P De Vries. Understanding similarity metrics in neighbour-based recommender systems. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, pages 48–55, 2013.

[8] Dheeraj Bokde, Sheetal Girase, and Debajyoti Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015.

[9] Igo Ramalho Brilhante, Jose Antonio Macedo, Franco Maria Nardini, Raffaele Perego, and Chiara Renso. On planning sightseeing tours with tripbuilder. *Information Processing & Management*, 51(2):1–15, 2015.

[10] Jason Brownlee. Clonal selection theory & clonalg-the clonal selection classification algorithm (csca). *Swinburne University of Technology*, page 38, 2005.

[11] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524, 2017.

[12] Isabel Cenamor, Tomás de la Rosa, Sergio Núñez, and Daniel Borrajo. Planning for tourism routes using social networks. *Expert Systems with Applications*, 69:1–9, 2017.

[13] Buru Chang, Yookyung Koh, Donghyeon Park, and Jaewoo Kang. Content-aware successive point-of-interest recommendation. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 100–108. SIAM, 2020.

[14] Lei Chen, Lu Zhang, Shanshan Cao, Zhiang Wu, and Jie Cao. Personalized itinerary recommendation: Deep and collaborative learning with textual information. *Expert Systems with Applications*, 144:113070, 2020.

[15] Chen Cheng, Haiqin Yang, Irwin King, and Michael Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 26, 2012.

[16] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[17] L Nunes De Castro and Fernando J Von Zuben. The clonal selection algorithm with engineering applications. In *Proceedings of GECCO*, volume 2000, pages 36–39, 2000.

[18] Ali Divsalar, Pieter Vansteenwegen, and Dirk Cattrysse. A variable neighborhood search method for the orienteering problem with hotel selection. *International Journal of Production Economics*, 145(1):150–160, 2013.

[19] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. *Collaborative filtering recommender systems*. Now Publishers Inc, 2011.

[20] Absalom E Ezugwu, Verosha Pillay, Divyan Hirasen, Kershen Sivanarain, and Melvin Govender. A comparative study of meta-heuristic optimization algorithms for 0–1 knapsack problem: Some initial results. *IEEE Access*, 7:43979–44001, 2019.

[21] Gregory Ference, Mao Ye, and Wang-Chien Lee. Location recommendation for out-of-town users in location-based social networks. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 721–726, 2013.

[22] Paola Festa and Mauricio GC Resende. Grasp: An annotated bibliography. In *Essays and surveys in metaheuristics*, pages 325–367. Springer, 2002.

[23] Fedor V Fomin and Andrzej Lingas. Approximation algorithms for time-dependent orienteering. *Information Processing Letters*, 83(2):57–62, 2002.

[24] Zachary Friggstad, Sreenivas Gollapudi, Kostas Kollias, Tamas Sarlos, Chaitanya Swamy, and Andrew Tomkins. Orienteering algorithms for generating travel itineraries. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 180–188, 2018.

[25] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[26] Ander Garcia, Olatz Arbelaitz, Maria Teresa Linaza, Pieter Vansteenwegen, and Wouter Souffriau. Personalized tourist route generation. In *International Conference on Web Engineering*, pages 486–497. Springer, 2010.

[27] Michael R Garey and David S Johnson. Computers and intractability. *A Guide to the*, 1979.

[28] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, Nikolaos Vathis, and Christos Zaroliagis. The ecompass multimodal tourist tour planner. *Expert systems with Applications*, 42(21):7303–7316, 2015.

[29] Damianos Gavalas, Vlasios Kasapakis, Charalampos Konstantopoulos, Grammati Pantziou, and oNiklaos Vathis. Scenic route planning for tourists. *Personal and Ubiquitous Computing*, 21(1):137–155, 2017.

[30] Damianos Gavalas, Charalampos Konstantopoulos, Konstantinos Mastakas, and Grammati Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014.

[31] Mihai Gavrilas. Heuristic and metaheuristic optimization techniques with application to power systems. *Technical University of Iasi, D. Mangeron Blvd., Iasi, Romania*, 2010.

[32] Aristides Gionis, Theodoros Lappas, Konstantinos Pelechrinis, and Evimaria Terzi. Customized tour recommendations in urban areas. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 313–322, 2014.

[33] Jiqing Gu, Chao Song, Wenjun Jiang, Xiaomin Wang, and Ming Liu. Enhancing personalized trip recommendation with attractive routes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 662–669, 2020.

[34] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2):315–332, 2016.

[35] Qing Guo, Zhu Sun, Jie Zhang, and Yin-Leng Theng. An attentional recurrent neural network for personalized next location recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 83–90, 2020.

[36] Tong Guo, Bin Guo, Yi Ouyang, Zhiwen Yu, Jacqueline CK Lam, and Victor OK Li. Crowdtravel: scenic spot profiling by using heterogeneous crowdsourced data. *Journal of Ambient Intelligence and Humanized Computing*, 9(6):2051–2060, 2018.

[37] A Hanif Halim and IJAoCMiE Ismail. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem. *Archives of Computational Methods in Engineering*, 26(2):367–380, 2019.

[38] Qiang Hao, Rui Cai, Changhu Wang, Rong Xiao, Jiang-Ming Yang, Yanwei Pang, and Lei Zhang. Equip tourists with knowledge mined from travelogues. In *Proceedings of the 19th international conference on World wide web*, pages 401–410, 2010.

[39] Marisa G Kantor and Moshe B Rosenwein. The orienteering problem with time windows. *Journal of the Operational Research Society*, 43(6):629–635, 1992.

[40] Amir Khatibi, Fabiano Belem, Ana P Silva, Dennis Shasha, Marcos A Goncalves, et al. Improving tourism prediction models using climate and social media data: A fine-grained approach. In *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[41] Jinyoung Kim, Hyungjin Kim, and Jung-hee Ryu. Triptip: a trip planning service with tag-based recommendation. In *CHI'09 Extended Abstracts on Human Factors in Computing Systems*, pages 3467–3472. 2009.

[42] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):1–24, 2010.

[43] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[44] Serhan Kotiloglu, Theodoros Lappas, Konstantinos Pelechrinis, and PP Repoussis. Personalized multi-period tour recommendations. *Tourism Management*, 62:76–88, 2017.

[45] Yohei Kurata and Tatsunori Hara. Ct-planner4: Toward a more user-friendly interactive day-tour planner. In *Information and communication technologies in tourism 2014*, pages 73–86. Springer, 2013.

[46] Antônio HG Leite, Fabrıcio Benevenuto, and Mirella M Moro. Triptag: Ferramenta de planejamento de viagens baseada em experiências de usuários de redes sociais. In *28 TH BRAZILIAN SYMPOSIUM ON DATABASES*, page 37, 2013.

[47] Huayu Li, Yong Ge, Richang Hong, and Hengshu Zhu. Point-of-interest recommendations: Learning potential check-ins from friends. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 975–984, 2016.

[48] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 831–840, 2014.

[49] Zhixue Liao and Weimin Zheng. Using a heuristic algorithm to design a personalized day tour route in a time-dependent stochastic environment. *Tourism Management*, 68:284–300, 2018.

[50] Kwan Hui Lim, Jeffrey Chan, Shanika Karunasekera, and Christopher Leckie. Tour recommendation and trip planning using location-based social media: A survey. *Knowledge and Information Systems*, 60(3):1247–1275, 2019.

[51] Yiding Liu, Tuan-Anh Nguyen Pham, Gao Cong, and Quan Yuan. An experimental evaluation of point-of-interest recommendation in location-based social networks. *Proceedings of the VLDB Endowment*, 10(10):1010–1021, 2017.

[52] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[53] Eric Hsueh-Chan Lu, Ching-Yu Chen, and Vincent S Tseng. Personalized trip recommendation with multiple constraints by mining user check-in behaviors. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 209–218, 2012.

[54] Jie Lu, Dianshuang Wu, Mingsong Mao, Wei Wang, and Guangquan Zhang. Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32, 2015.

[55] Sean Luke. 1 essentials of metaheuristicsl. 1.

[56] John McCall. Genetic algorithms for modelling and optimisation. *Journal of computational and Applied Mathematics*, 184(1):205–222, 2005.

[57] Tien T Nguyen, Pik-Mai Hui, F Maxwell Harper, Loren Terveen, and Joseph A Konstan. Exploring the filter bubble: the effect of using recommender systems on content diversity. In *Proceedings of the 23rd international conference on World wide web*, pages 677–686, 2014.

[58] Prarthana Padia, Kwan Hui Lim, Jeffrey Cha, and Aaron Harwood. Sentiment-aware and personalized tour recommendation. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 900–909. IEEE, 2019.

[59] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135, 2008.

[60] Leif E Peterson. K-nearest neighbor. *Scholarpedia*, 4(2):1883, 2009.

[61] Tuan-Anh Nguyen Pham, Xutao Li, and Gao Cong. A general model for out-of-town region recommendation. In *Proceedings of the 26th International Conference on World Wide Web*, pages 401–410, 2017.

[62] Mauricio GC Resende and Celso C Ribeiro. Grasp: Greedy randomized adaptive search procedures. In *Search methodologies*, pages 287–312. Springer, 2014.

[63] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[64] Marc HJ Romanycia and Francis Jeffry Pelletier. What is a heuristic? *Computational Intelligence*, 1(1):47–58, 1985.

[65] Joy Lal Sarkar and Abhishek Majumder. A new point-of-interest approach based on multi-itinerary recommendation engine. *Expert Systems with Applications*, 181:115026, 2021.

[66] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[67] Richard Schaller. Planning and navigational assistance for distributed events. In *2nd Workshop on Context Aware Intelligent Assistance*, pages 44–56, 2011.

[68] SN Sivanandam and SN Deepa. Genetic algorithms. In *Introduction to genetic algorithms*, pages 15–37. Springer, 2008.

[69] Klaus Ten Hagen, Ronny Kramer, Marcel Hermkes, Bjoern Schumann, and Patrick Mueller. Semantic matching and heuristic search for a dynamic tour guide. In *Information and Communication Technologies in Tourism 2005*, pages 149–159. Springer, 2005.

[70] Curtis HK Tsang, Monde HC Woo, and Chris Bloor. An object oriented intelligent tourist advisor system. In *1996 Australian New Zealand Conference on Intelligent Information Systems. Proceedings. ANZIIS 96*, pages 6–9. IEEE, 1996.

[71] Berna Haktanirlar Ulutas and Sadan Kulturel-Konak. A review of clonal selection algorithm and its applications. *Artificial Intelligence Review*, 36(2):117–138, 2011.

[72] Julián Urbano, Harlley Lima, and Alan Hanjalic. Statistical significance testing in information retrieval: an empirical analysis of type i, type ii and type iii errors. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514, 2019.

[73] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290, 2009.

[74] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, and Dirk Van Oudheusden. The city trip planner: an expert system for tourists. *Expert Systems with Applications*, 38(6):6540–6546, 2011.

[75] Pieter Vansteenwegen and Dirk Van Oudheusden. The mobile tourist guide: an or opportunity. *OR insight*, 20(3):21–27, 2007.

[76] Hao Wang, Manolis Terrovitis, and Nikos Mamoulis. Location recommendation in location-based social networks using user check-in data. In *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 374–383, 2013.

[77] Wei Wang, Guangquan Zhang, and Jie Lu. Collaborative filtering with entropy-driven user similarity in recommender systems. *International Journal of Intelligent Systems*, 30(8):854–870, 2015.

[78] Budhi S Wibowo and Monica Handayani. A genetic algorithm for generating travel itinerary recommendation with restaurant selection. In *2018 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)*, pages 427–431. IEEE, 2018.

[79] Gerhard J Woeginger. Space and time complexity of exact algorithms: Some open problems. In *International Workshop on Parameterized and Exact Computation*, pages 281–290. Springer, 2004.

[80] Laurence A Wolsey. Mixed integer programming. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–10, 2007.

[81] Haoran Xin, Xinjiang Lu, Tong Xu, Hao Liu, Jingjing Gu, Dejing Dou, and Hui Xiong. Out-of-town recommendation with travel intention modeling. *arXiv preprint arXiv:2101.12555*, 2021.

[82] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik-Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the 34th*

international ACM SIGIR conference on Research and development in Information Retrieval, pages 325–334, 2011.

[83] Hongzhi Yin, Xiaofang Zhou, Bin Cui, Hao Wang, Kai Zheng, and Quoc Viet Hung Nguyen. Adapting to user interest drift for poi recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 28(10):2566–2581, 2016.

[84] Haochao Ying, Jian Wu, Guandong Xu, Yanchi Liu, Tingting Liang, Xiao Zhang, and Hui Xiong. Time-aware metric embedding with asymmetric projection for successive poi recommendation. *World Wide Web*, 22(5):2209–2224, 2019.

[85] Phatpicha Yochum, Liang Chang, Tianlong Gu, and Manli Zhu. An adaptive genetic algorithm for personalized itinerary planning. *IEEE Access*, 8:88147–88157, 2020.

[86] Kyung-Hyan Yoo, Marianna Sigala, and Ulrike Gretzel. Exploring tripadvisor. In *Open tourism*, pages 239–255. Springer, 2016.

[87] Chenyi Zhang, Hongwei Liang, and Ke Wang. Trip recommendation meets real-world constraints: Poi availability, diversity, and traveling time uncertainty. *ACM Transactions on Information Systems (TOIS)*, 35(1):1–28, 2016.

[88] Chenyi Zhang, Hongwei Liang, Ke Wang, and Jianling Sun. Personalized trip recommendation with poi availability and uncertain traveling time. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 911–920, 2015.

[89] Jia-Dong Zhang and Chi-Yin Chow. Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 443–452, 2015.

[90] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. Lore: Exploiting sequential influence for location recommendations. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 103–112, 2014.

[91] Pengpeng Zhao, Haifeng Zhu, Yanchi Liu, Jiajie Xu, Zhixu Li, Fuzhen Zhuang, Victor S Sheng, and Xiaofang Zhou. Where to go next: A spatio-temporal gated network for next poi recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5877–5884, 2019.

[92] Shenglin Zhao, Irwin King, and Michael R Lyu. A survey of point-of-interest recommendation in location-based social networks. *arXiv preprint arXiv:1607.00647*, 2016.

[93] Weimin Zheng and Zhixue Liao. Using a heuristic approach to design personalized tour routes for heterogeneous tourist groups. *Tourism Management*, 72:313–325, 2019.

[94] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Ji Zhang, and Fulong Chen. A tourism destination recommender system using users' sentiment and temporal dynamics. *Journal of Intelligent Information Systems*, 51(3):557–578, 2018.

Appendix A

Background

KNN Mean Prediction Equation - Considers the mean weight of the item to make the prediction

$$\hat{y}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} sim(i,j)} \tag{A.1}$$

KNN with Z-Score Prediction Equation - Considers the normalized mean of the item to make the prediction

$$\hat{y}_{ui} = \mu_i + \sigma_i \frac{\sum\limits_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - \mu_j)/\sigma_j}{\sum\limits_{j \in N_u^k(i)} sim(i,j)} \tag{A.2}$$

Variable	Description
i	Prediction item
j	Item in user history
u	Target user
$\hat{y}_{ui}$	Rating predicted
$r_{u,j}$	Historic rating
k	Amount of neighbors
$N_u^k(i)$	Set of nearest neighbors of item i that were rated by the user u
μ	Average rating of a item
σ	Standard deviation of a item
$sim(i,j)$	Similarity function

Table A.1: kNN prediction variables description.

Appendix B

Maps

In this appendix we present the maps of the cities evaluated by us: Tiradentes (Figure B.1), Ouro Preto (Figure B.2), San Gimignano (Figure B.3), Cannes (Figure B.4), and Ibiza (Figure B.5). In the maps is possible to see all POIs by category in the cities.

Lastly, we present four different Ibiza maps showing the frequency of places selected by different algorithms. Note that the behavior of the Genetic Algorithm is very close to the Random Algorithm, while GRASP has a selection more similar to the Greedy.

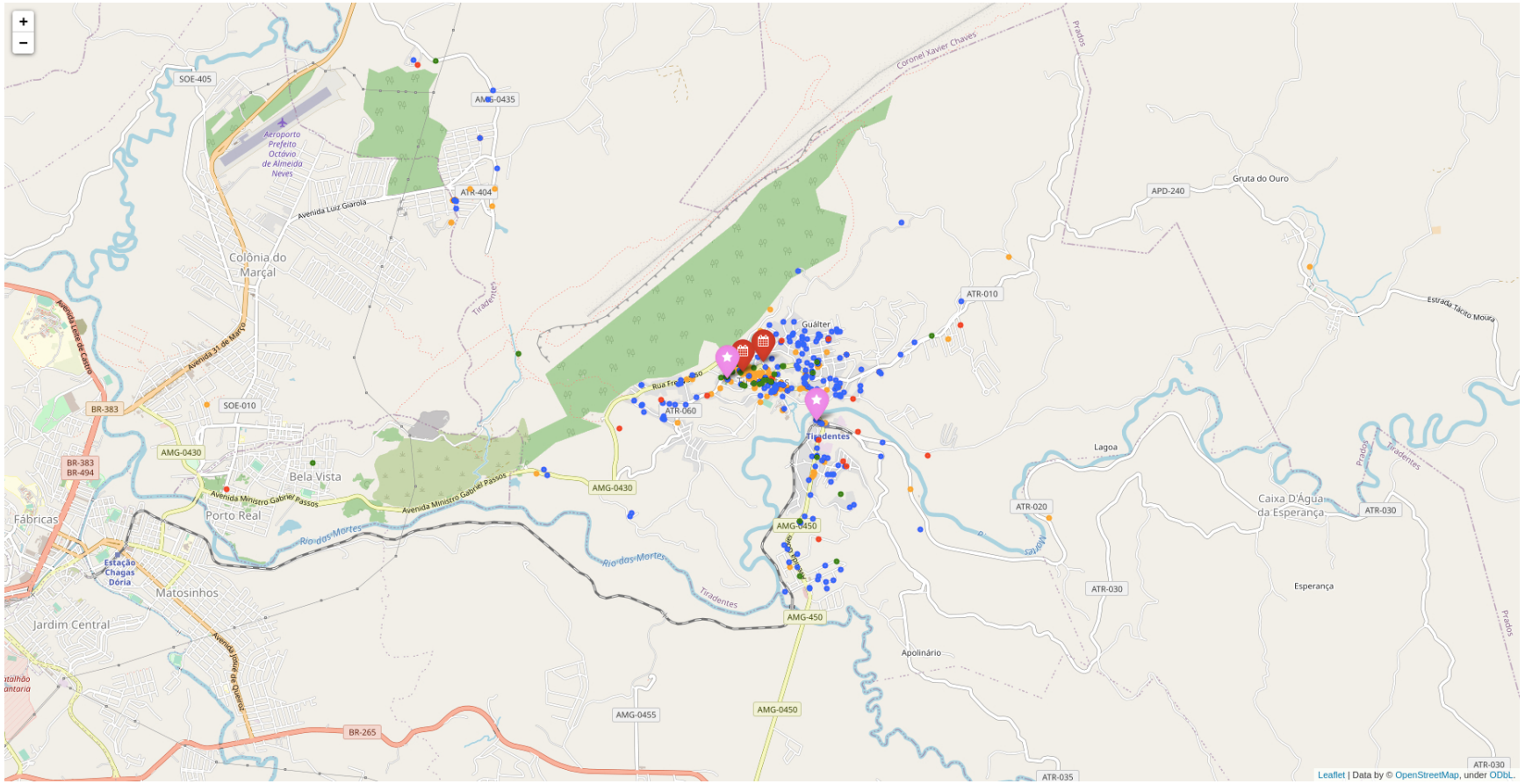


Figure B.1: Map of POIs in Tiradentes and places selected as mandatory and appointment. In green we have attractions, hotels are in blue, rental places are in red and orange are restaurants. With a start pink icon are mandatory places and in red with a calendar are appointments.

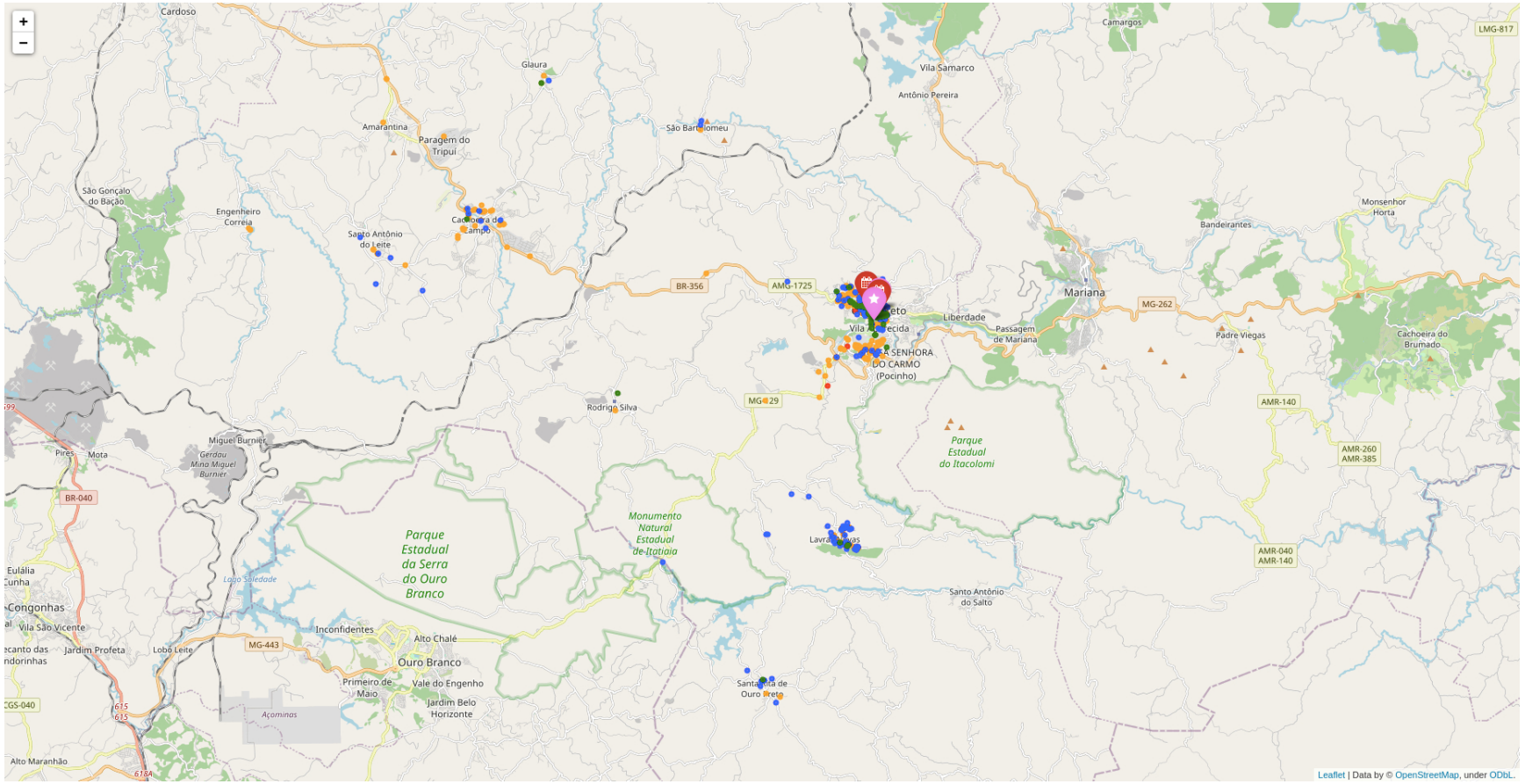


Figure B.2: Map of POIs in Ouro Preto and places selected as mandatory and appointment. In green we have attractions, hotels are in blue, rental places are in red and orange are restaurants. With a start pink icon are mandatory places and in red with a calendar are appointments.

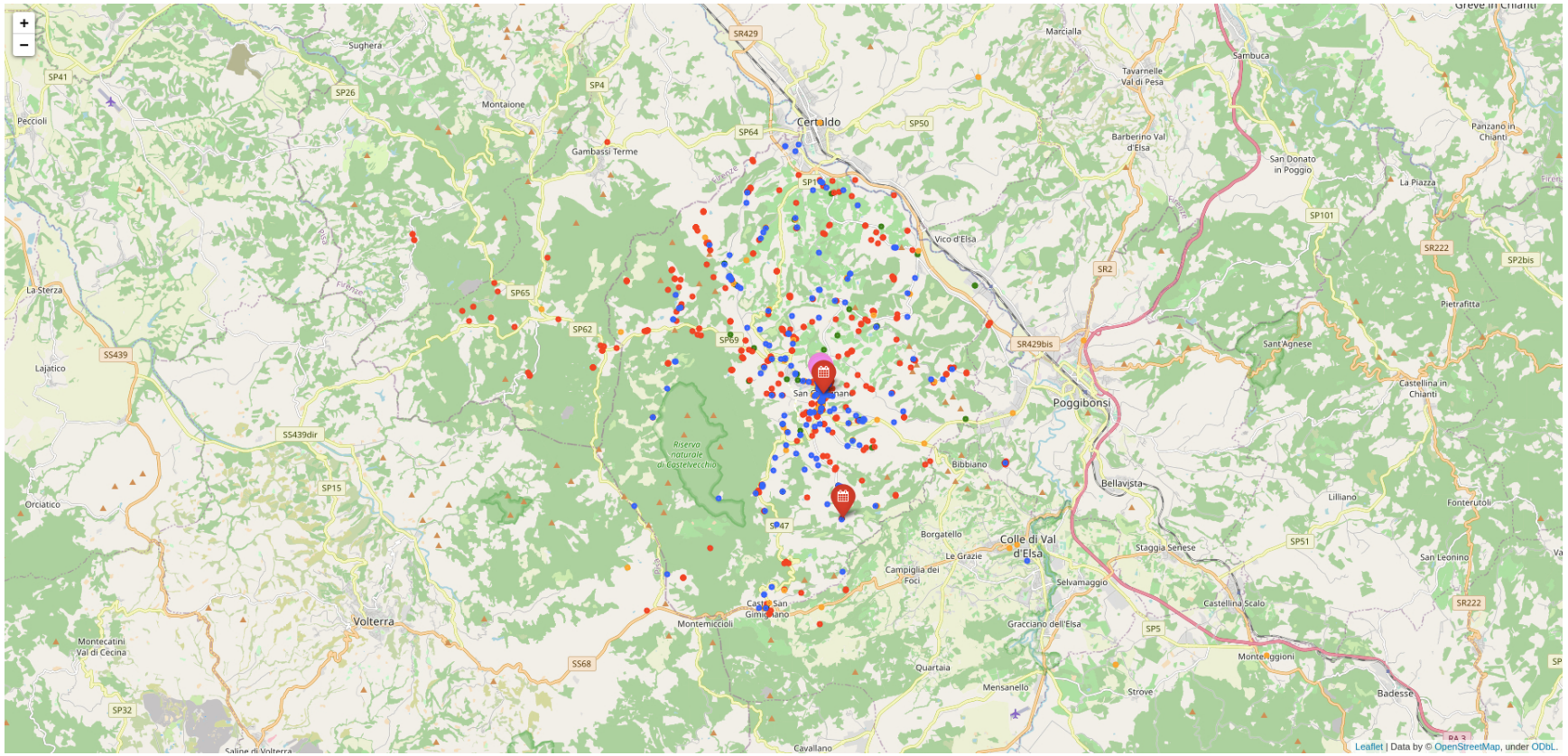


Figure B.3: Map of POIs in SG and places selected as mandatory and appointment. In green we have attractions, hotels are in blue, rental places are in red and orange are restaurants. With a start pink icon are mandatory places and in red with a calendar are appointments.

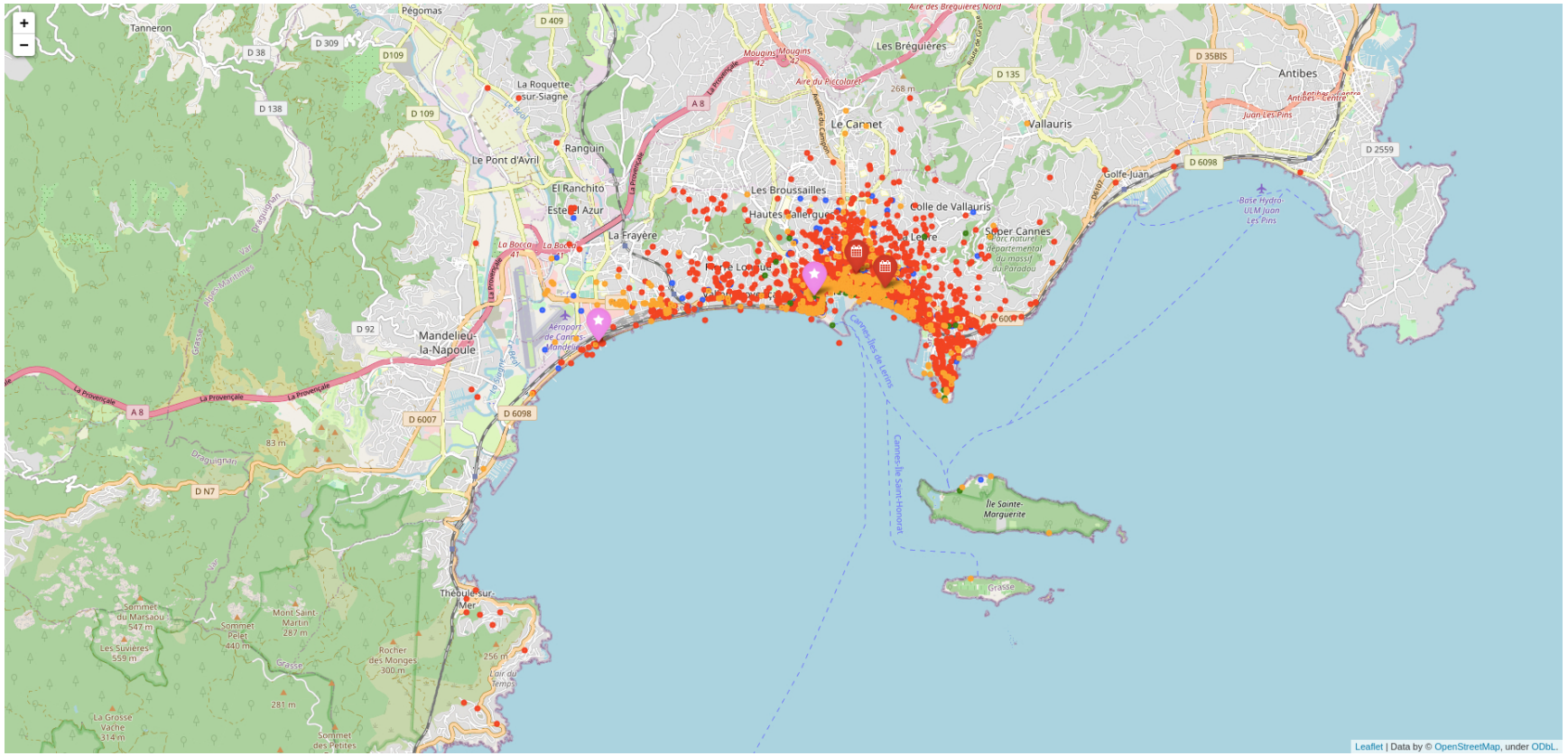


Figure B.4: Map of POIs in Cannes and places selected as mandatory and appointment. In green we have attractions, hotels are in blue, rental places are in red and orange are restaurants. With a start pink icon are mandatory places and in red with a calendar are appointments.

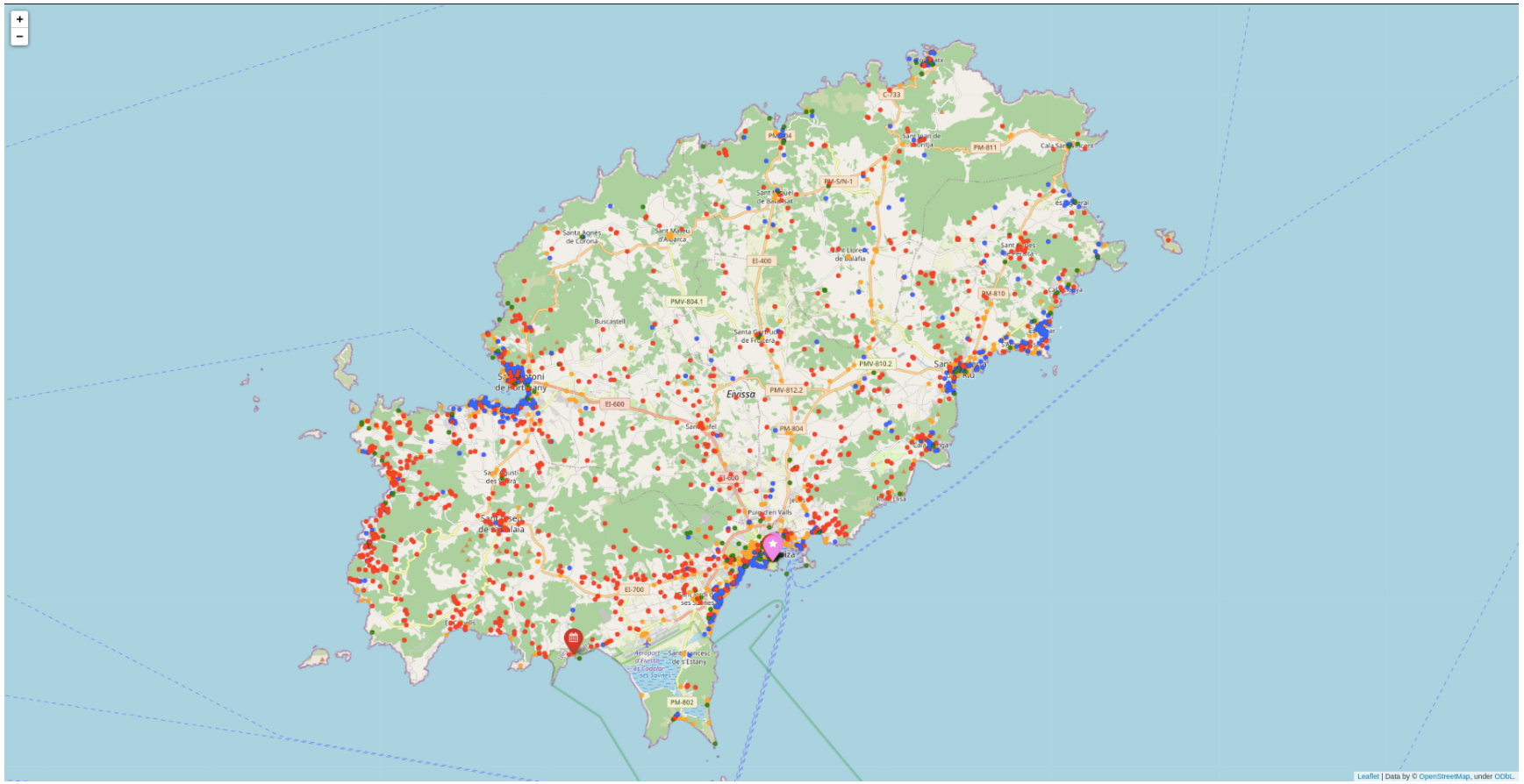


Figure B.5: Map of POIs in Ibiza and places selected as mandatory and appointment. In green we have attractions, hotels are in blue, rental places are in red and orange are restaurants. With a start pink icon are mandatory places and in red with a calendar are appointments.

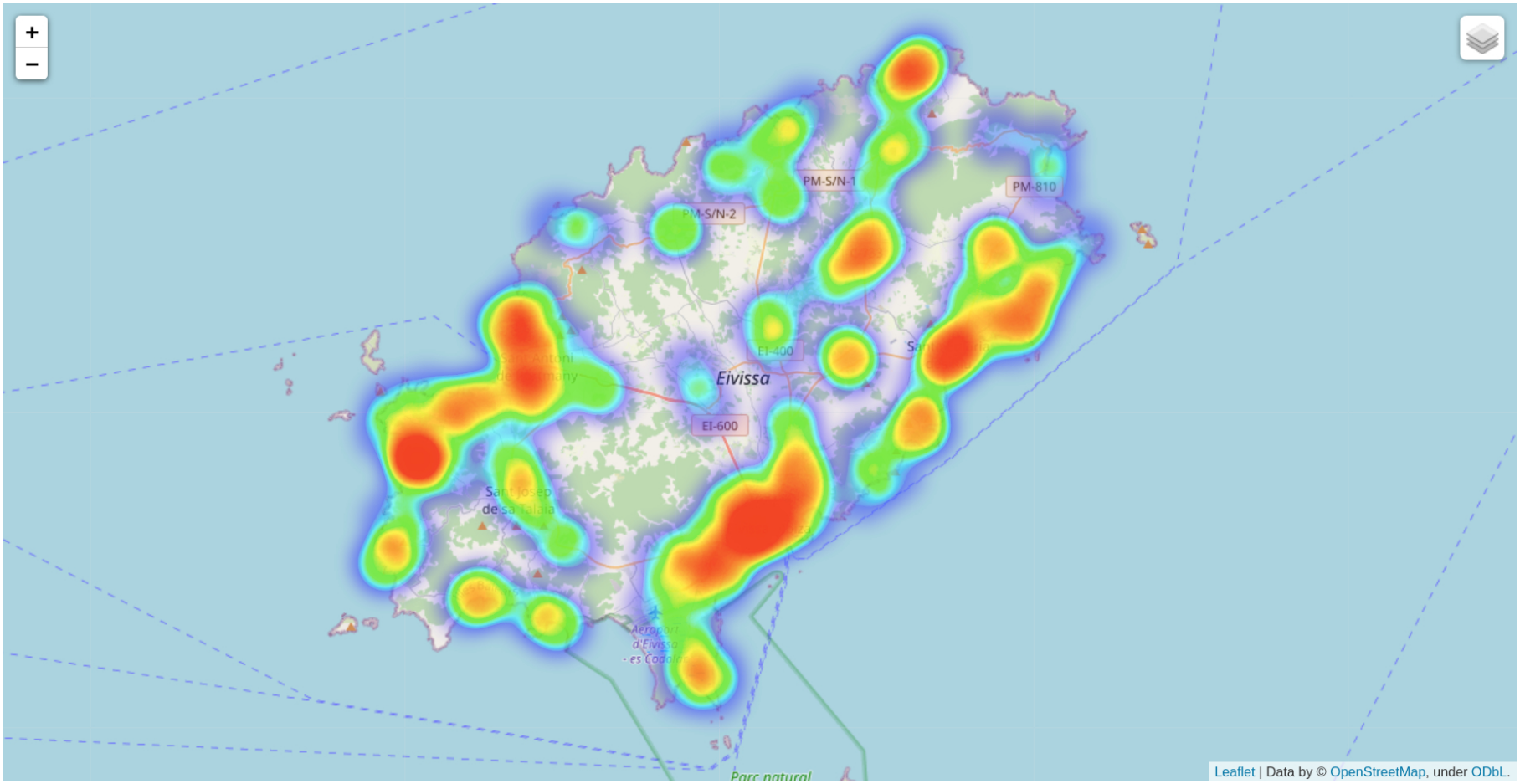


Figure B.6: Heatmap of most frequent places selected by the Random algorithm in Ibiza

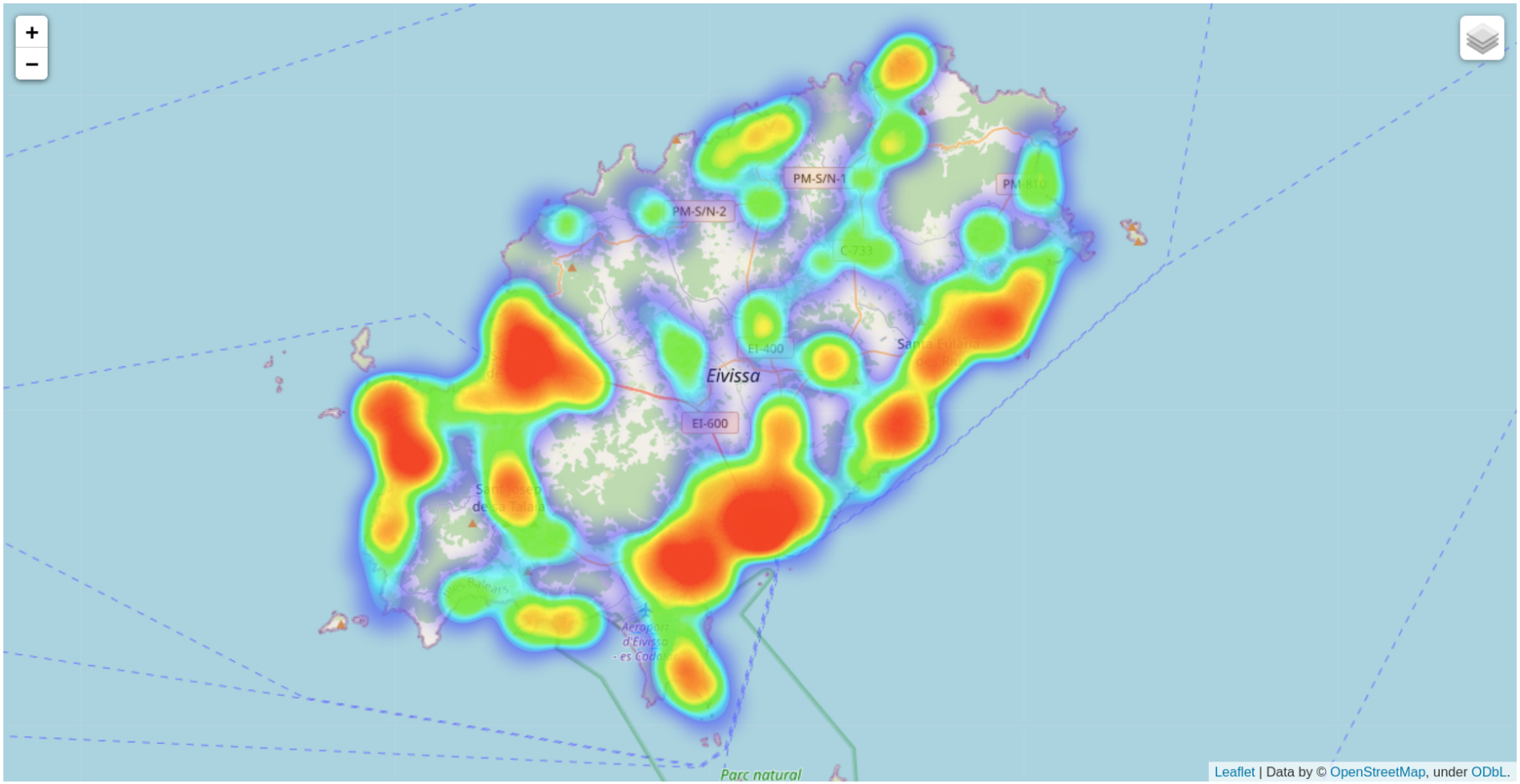


Figure B.7: Heatmap of most frequent places selected by the Genetic algorithm in Ibiza

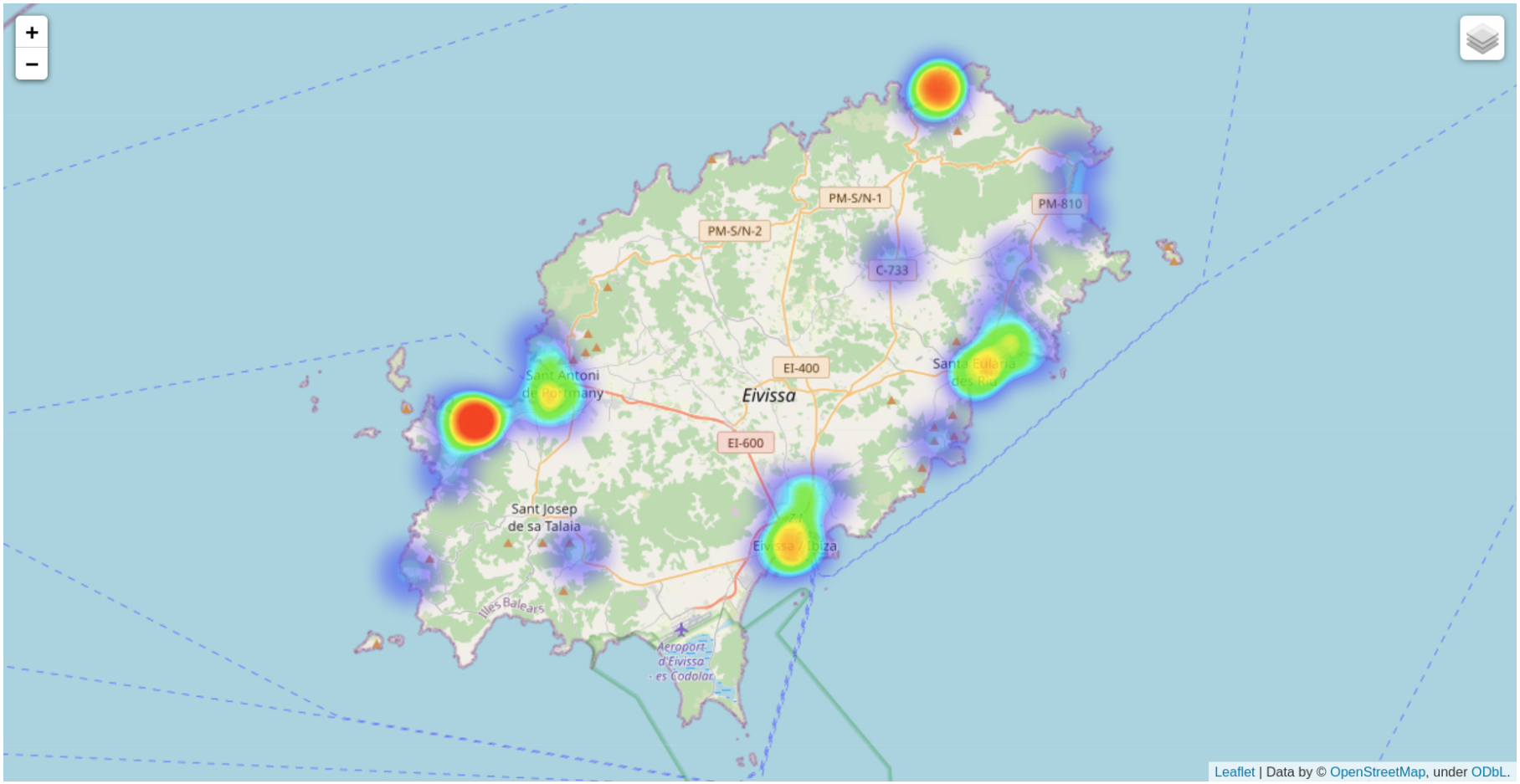


Figure B.8: Heatmap of most frequent places selected by the Greedy algorithm in Ibiza

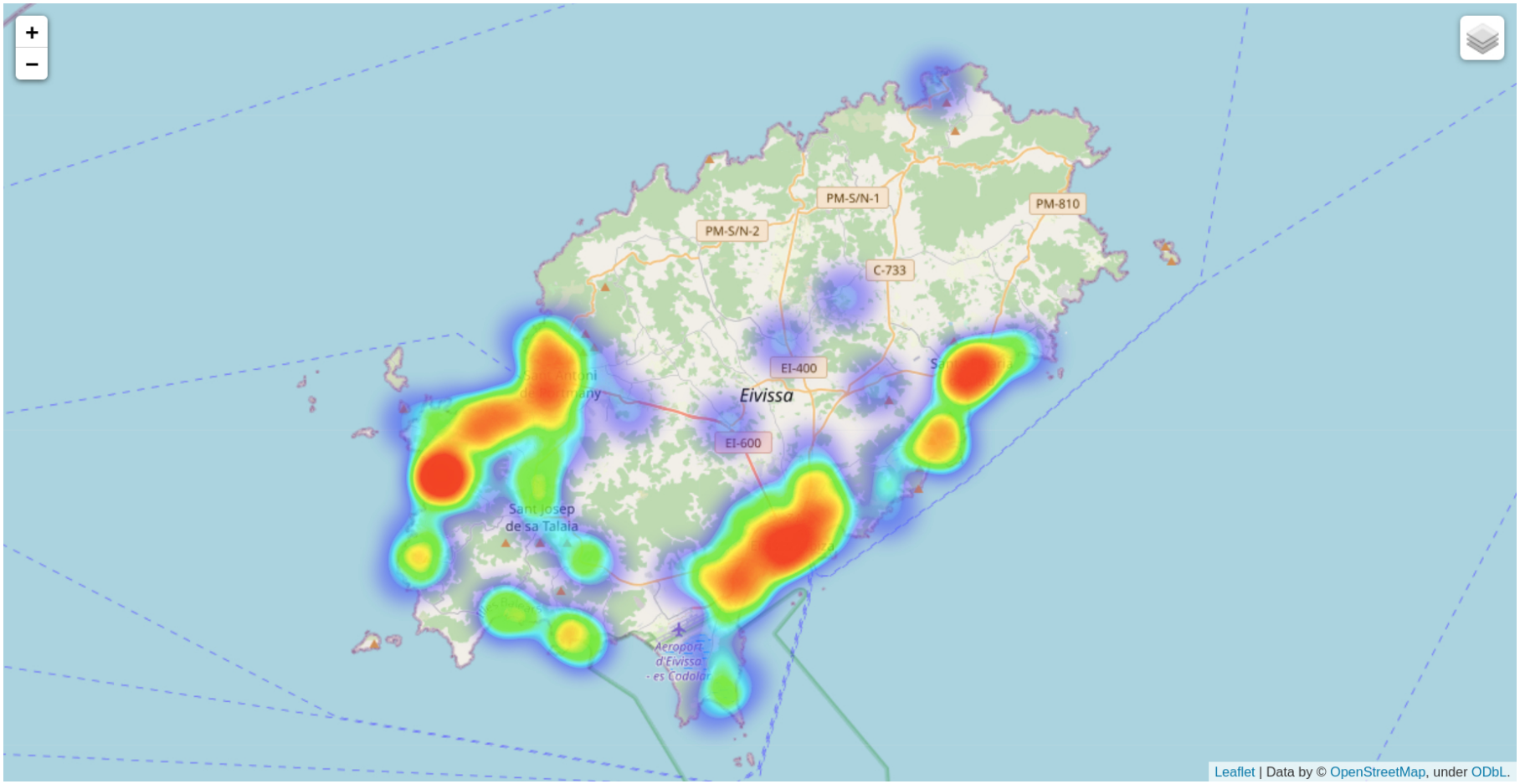


Figure B.9: Heatmap of most frequent places selected by the GRASP algorithm in Ibiza