E-TOURISM: CONTEXT-AWARE POINTS OF INTEREST FINDER AND TRIP DESIGNER

Hamzah Alghamdi

A thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the
Master of Applied Science in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

Many countries depend heavily on tourism for their economic growth. The invention of the web has opened new opportunities for tourists to discover new places and live new adventures. However, the number of possible destinations has become huge and even an entire lifespan would not be enough to visit all of these places. Even for one city, there are a significant number of possible places to visit. Nowadays, searching online to find an interesting place to visit is harder than ever, not because there is a lack of information but rather due to the vast amount of information that can be found.

Trip planning is a tedious task, especially when the tourist does not want to pick a preplanned itinerary from a traveling agency. That being said, even these preplanned itineraries need a lot of time and effort to be customized. Moreover, the set of itineraries that a tourist can select from is usually limited. In addition, there may be many places that tourists would enjoy visiting but that are not included in the itineraries. Thus, static planners do not always choose the right place at the right time. This is why the planning process should take into consideration many factors in order to give the tourist the best possible suggestions.

In this Thesis, we propose an algorithm called the Balanced Orienteering Problem to design trips for tourists. This algorithm, combined with a context-aware recommender system for tourism suggestions, create the infrastructure of the mobile application for the augmented reality tourism guide that we developed. We cover the background knowledge of tour planning problems and tourism recommender systems and describe the existing techniques. Furthermore, a comparison between the existing systems and our algorithm is completed to illustrate that our proposed algorithm yields better results. We also discuss

the workflow of our system implementation and how our mobile application is designed. Lastly, we address suggestions for future works and end with a conclusion.

Acknowledgements

First and foremost, I would like to express my gratitude to God for giving me the health, strength, and patience needed throughout this educational journey, allowing me to face all the ups and downs I encountered.

Secondly, I would like to express my sincere gratitude to my supervisor, Prof. Abdulmotaleb El Saddik, for the continuous support of my master's study, his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me overcome many crisis situations during this dissertation. I could not have imagined having a better advisor and mentor.

A very special thanks to Dr. Shiai Zho, who not only supported me during this research but also guided me in many life decisions; I am fortunate that he was always there when I needed him. I appreciate his extensive knowledge and skills in many areas and his collaboration on the subject. Additionally, I would like to show my gratitude to Dr. Mohamad Hoda who was a constant help throughout my thesis. I am also thankful to him for encouraging the use of correct grammar and consistent notation in my writings and for carefully reading and commenting on countless revisions of this manuscript.

I have been blessed with such a good and friendly group of colleagues in the Discover and MCR lab. AbdulRahman, Faisal, Basim, Majed, and Saeed were always there to listen to me on many difficult days and played an important supporting role during these past two years. I humbly thank Mr. Majed who provided me with great tips on Python and Computer Vision tools and much useful software.

Lastly, a special thanks go to my beloved parents, AbdulRahman and Faiza, for their faith in me and their belief that I could one day have as much success as I do today. They have always been my main source of love, encouragement, support, and strength. Everything that I have done in my life so far, I owe to them. A special thanks also goes to my siblings, for always being with me and granting me love, motivation, and care.

Contents

List of Tables

List of Figures

List of Acronyms and Abbreviations

ABRS	Author-Based Recommender Systems
API	Application Program Interface
APX-hard	Polynomial-Time Approximation hard
AR	Augmented Reality
AET	Average Execution Time
CBRS	Content-Based Recommender Systems
CF	Collaborative Filtering
CPTP	Capacitated Profitable Tour Problem
FR	Functional Requirements
GPS	Global Positioning System
JSON	JavaScript Object Notation
LBSNs	Location-Based Social Networks
NP-hard	Non-deterministic Polynomial-time hard
OP	Orienteering Problem
OPTW	Orienteering Problem with Time Windows
OS	Operating System
PCTSP	Prize Collecting Traveling Salesman Problem

PCVRP	Prize Collecting Vehicle Routing Problem
POIs	Points of Interest
PTP	Profitable Tour Problem
QR	Quality Requirements
REST	Representational State Transfer
RS	Recommender Systems
RSSI	Received Signal Strength Indicator
SQL	Structured Query Language
TDOP	Time Dependent Orienteering Problem
TOP	Team Orienteering Problem
TOPTW	Team Orienteering Problem with Time Windows
TSP	Traveling Salesman Problem
TSPP	Traveling Salesman Problem with Profit
TTDP	Tourist Trip Design Problem
VRPP	Vehicle Routing Problem with Profits
VRPP-TD	Vehicle Routing Problem with Profits and Time Deadlines

Chapter 1

1 Introduction

While tourists rely more and more on the Internet to find new places to visit, it is still a difficult task to search for interesting places, especially if the tourist is not familiar with the area. Even when the tourist has time to plan ahead for the tour, there are cases where planning ahead is infeasible; for instance when there is a sudden change of plans or if there is extra time before the next appointment. These cases require special attention as usually the tourist has a final destination to reach within a limited period of time. Moreover, they represent real world situations where the tourist, in general, has a place to return to (e.g. the hotel) at a particular time.

The planning problem is not new; in fact, it has been addressed since the 1980s [1]. Even with all the advancements in e-tourism, trip planning is still one of the hardest tasks to design. There exist different approaches to solve the tour planning problem (TPP), one of which is known in the literature as the Orienteering Problem (OP) [2]. Even though OP has been addressed and implemented several times during the last decade, it is still not widely used in the tourism industry [1], due to several factors like the complexity of the algorithm and the lack of the means to collect information about places in a massive scale (such as crowdsourcing).

In this thesis, we introduce a new variation to the orienteering problem algorithm. Although the proposed algorithm is similar to the well-known varieties of OP, it outperforms them, as we will see later, by proposing a solution to the TPP that reflects real life trips better by allocating time differently. To validate the proposed solution, a mobile

application has been designed and implemented. The application consists of three parts, namely the proposed balanced orienteering problem algorithm, a recommender system (RS) that ranks the places and gives suggestions to the algorithm, and a scheduler that keeps track of the trip and modifies it when necessary.

1.1 Motivation

The tourism industry is one of the biggest industries in the world [3], and many countries depend heavily on tourism for their economic growth. The invention of the web has opened new opportunities for tourists to discover new places and live new adventures. However, the number of possible destinations has become huge, and even an entire lifespan would not be enough to visit all of these places. Even for one city, there are a significant number of possible places to visit. Nowadays, searching online to find an interesting place to visit is harder than ever, not because there is a lack of information but rather due to the vast amount of information that can be found.

Trip planning is a tedious task, especially when the tourist does not want to pick a preplanned itinerary from a traveling agency. That being said, even these preplanned itineraries need a lot of time and effort to be customized. Moreover, the set of itineraries that a tourist can select from is usually limited. In addition, there may be many places that tourists would enjoy visiting but that are not included in the itineraries. Thus, static planners do not always choose the right place at the right time. This is why the planning process should take into consideration many factors in order to give the tourist the best possible suggestions.

Even with all the right suggestions at hand, planning the route for the trip is still difficult because the proposed plan has to respect the tourists' time and budget limits. Thus, evaluating the choices and finding the best way to enjoy the trip leads to more problems.

1.2 Thesis Goal

To solve the tourist trip design problem, we define a system that schedules a trip for the user with three key features:

- Must reflect user preferences.
- Does not only consider user preferences but also the popularity of places, among tourists, at trip creation.
- Provides the possibility of rescheduling if the plan can no longer meet the time limit T.

1.3 Thesis Contribution

In this thesis, we create a comprehensive approach to e-tourism and, more specifically, to trip designing and recommending. The aimed contributions for this work are as follows:

We develop a dynamic framework that will adapt to the user's behavior during the trip. Also, this framework increases the relevance of the recommendations by adding social networks mining, user preference tracking and Context awareness is in the form of weather evaluation, traffic conditions, and user history. As a part of our framework, we propose an algorithm called Balanced Orienteering Problem to design trips for tourists based on the

Orienteering Problem algorithm, which solves for the best trip. Our algorithm recommends routes based on different criteria than the original and also behaves differently. This is done by the introduction clustering into the problem, in order to optimize the speed. Lastly, since our algorithm behaves differently, we evaluate it against the original to show that it yields better results.

1.3.1 Publications

Based on the research we have done, we have published the following paper:

- H. Alghamdi and A. El Saddik, "Mobile Dynamic Tour Trip Planner," *2016 IEEE International Symposium on Multimedia (ISM)*, San Jose, CA, 2016.

1.4 Thesis Organization

The rest of this thesis is organized as follows:

- Chapter 2 contains the background details of both recommender systems and trip design. The background details are divided into two parts. The RS, which contain author-based and content-based approaches, and the trip design part with its two main components; the one trip model and the multi-trips model. Also, in this chapter, existing recommendation and trip design works done by the e-tourism society are discussed, and the comparison against the proposed work is shown.
- Chapter 3 first presents the formulation of the problem addressed in this thesis. Then, the proposed algorithm for trip design is described in detail, together with a discussion of the different parts of the system.

- Chapter 4 explains the system's implementation and technical details for the software application components, including the workflow of the framework and how different situations are handled.
- Chapter 5 presents the experiments, results and analysis in greater detail including data collection and the analysis of the different algorithms' prediction accuracy.
- Chapter 6 concludes this thesis and proposes future research directions.

Chapter 2

2 Background and Literature Review

In this chapter, a review of various trip planning recommender systems is provided in Section 2.1. Then, the background details of possible solutions to the Tourist Trip Design Problem are discussed in Section 2.2. Next, in Section 2.3, existing tourism frameworks created by the community are described. Finally, Section 2.4 presents the chapter summary and discusses the novelty of the work.

2.1 Recommender Systems

Recommender systems try to determine the user's interests (e.g. cars, music, and books) and show relevant data. In the tourism business, those systems filter out places of no interest and rank the rest for the user by predicting his ranking for those places [4]. The systems are mainly divided into two types. The first type is the author-based system, where the tourist is the focus, and the system provides very personalized suggestions. On the other hand, in a content-based system, the places are the main focus, and the system measures a number of attributes about each place. In the next sections, we provide a detailed explanation on how each of these systems work.

2.1.1 Author-Based Recommender Systems

The aim of the author model is to mine the places visited in each city by the main tourist, based on his travel history. Then, we find another tourist who has a similar travel history and has also visited the city that the main tourist wants to visit. The system then

suggests the travel history of the other user in that city to the main tourist [5]. A good example of this system is if we have user A who does not travel a lot and wants to visit a city X for the first time, the system will look into the user's history and find a city Y that the user A already visited. The system will then try to find a user B, who has visited both city Y and city X and has a similar traveling history to user A in city Y. The system will work better for frequent travelers, where the similarities in the history may include more than one city.

This model may improve further if the social influence, geographical influence, or temporal influence is considered in the ranking [6]. If social influence is considered, the system will consider what people with a similar behavioral trip to the user have done and will take their info into consideration when suggesting new places to the user. With the rise of social networks, the effect of social influences has increased more than ever, and a lot of recommender systems have used these networks to improve the overall quality of their suggestions by considering the interest of people with whom the user interacts [7].

The next type of influence is the geographical influence, since the user will most likely visit places near his most visited places, e.g. a restaurant near his favorite bar or near his hotel. Studies show that users will visit nearby locations more than those at a further distance [8]. That is why it is important to determine where the user currently is and to consider nearby places as better suggestions than faraway ones, at least in certain contexts.

The last influencer is the temporal one. According to Zhang et al. [6], the temporal influence can be divided further into five degrees of influence; the absolute time factor, the sequential time factor, the usage of time information for location predictions, the periodic time pattern discovery, and the periodic time pattern dedication. The absolute time factor

is the effect of the passage of time on the user's preferences, and how new experiences may be preferable to the user [9], e.g. a new favorite music band is preferred over the favorite music band from seven years ago. In the sequential time degrees of influence, the system will look for a temporal pattern in preferences. In other words, given that the user has done sequence $X_{(n-1)}$ in preferences, the user is most likely to do the $X_{(n)}$ preference, e.g. after watching the super bowl the user prefers to eat chicken wings. The third degree is the usage of time information for location predictions, where the system tries to find a geographic-temporal relation in the user's behavior history and then uses this relation to recommend the preference given the time [10], e.g. before work, the system will suggest only restaurants that are near the workplace. Fourthly, the periodic time pattern discovery, where the system analyzes when the user visits specific places [11]. E.g. user X usually visits bars after 9 pm; therefore the system will be more likely to recommend bars after 9 pm. Lastly, the periodic time pattern dedication works in the opposite way as the last one, where the system will divide the user time into units and analyzes what the user does during each time unit [12]. The units can range from hours of the day in a very simple system to an hour of a specific day of the week during a specific season in more complex systems.

It must be noted that those are not the only types of influences that can be used to improve the author based recommender system. Factors such as age, gender, demographics and personality treats can also impact the system's performance if not taken into consideration.

2.1.2 Content-Based Recommender Systems

Unlike the author model, the content model focuses on the places and how the tourist behaves toward them. This model can be used to dig out places that may not be found in any travel guide such as local sights that are popular among locals. This model is also good if the main tourist does not have a rich travel history [13]. A good example will be the temporal effect, when the tourist visits a certain place often. Another example is the geographical effect, where the tourist will go next or where they came from. Lastly, if we take the social effect, what kind of social assessments do tourists give this place, e.g. a honeymoon destination [14].

A content-based recommender system works by collecting information about a set of places. The set of places can cover only a part of a city or the entire globe. The system then uses the information collected to rank those places accordingly. The ranking criteria can be as simple as a "number of visitors" scale, or as complex as a scale with Different variables such as the age, gender, and ethnicity of visitors, time of the day and season of the year. An example of the later is a ranking scale, plus a category scale, plus a limitation scale, where each of the scales have a different weight when it comes to recommending the place. The category scale can inflate some category members such as parks and deflate another category such as religious places. The importance of the category scales is to give a fair comparison between category members without being affected by the other categories' members. Some places get deeply affected by some factors, like for example weather conditions or high-season times. If those factors are taken into consideration when suggesting places, the recommendations should be improved. E.g. if one of the best hot

springs in the city has a very long wait time during the high-season, most people will not consider it unless they have the spare time.

2.1.3 Approaches for Recommender Systems

Ideally, author-based recommender systems (ABRS) should yield better results than content-based recommender systems (CBRS). However, the limitations of ABRS are too many for it to be able to yield the best results. Nonetheless, CBRS are also far from being perfect. That is why most recommender systems use a hybrid solution between the two [5]. The next subsection provides an example of an influencer and how both systems react to it. Then, there is a discussion of the impact of social networks on those systems. Lastly, at the end of the chapter we revisit both systems to discuss their importance and their effect after seeing some of the works implementing them.

2.1.3.1 Recommender Systems Influencers

In subsections 2.1.1 and 2.1.2, we reviewed how each type of recommender system deals with influencers. As a demonstration, in this section, only one of those influencers will be considered to show the difference between the content-based systems, the author based ones and their hybrid.

The study of the effect of weather on tourist behavior [15] clearly shows how the weather affect places and people. It shows that outdoor places suffer from rainy days a lot, while indoor places get an increase in the number of tourists on those days. Good weather will also cause some places to have a spike in the number of visitors, e.g. water parks. However, the study shows that usually, an individual tourist's plans are not changed based

on the weather conditions. Even though they may not have changed their plans, a lot of tourists complain about the weather, and it definitely affects their level of satisfaction. By taking a hybrid approach to this and considering the tourist as well as the degree to which the place is influenced by the weather, we can obtain a hybrid score:

$$\text{Score}_{\text{hybrid}} = \text{Score}_{\text{author}} * \text{Score}_{\text{content}} \quad (1.1)$$

This hybrid score will be very low for a family visiting the park on a rainy day but will not be as low for a teenager going to a water park on the same day; those effects have been discussed more in [15]. By considering the influence on both the tourist and the place, the system will be more accurate when making suggestions, but this means that the data the system needs to collect and analyze increases significantly. For many years, it was very difficult to collect that much data, but that was before the rise of the social networks and crowdsourcing websites [16].

2.1.3.2 Impact of Social Networks on RS

Social networks provide an environment for users to share their data and experiences on platforms such as Twitter, Instagram, and Flickr. Designers of recommender systems make use of these platforms. In the author-based RS for tourism, the adaptation for social networks occurred with the introduction of Location-Based Social Networks (LBSNs) such as Foursquare, Gowalla, and Facebook Places. This shows that for each application, there is a platform that suits it the most. For example, if the system uses the user's Facebook account to get his tourism history, a fatal limitation is if the user does not have an account or valid information in his account. Also, due to the heavy reliance on social networks in RS, a new problem has arisen, known as the cold-start

problem [17]. The cold-start problem in RS is when a new user/place enters the system; there is no collected information about him/it. Usually, creating initial information that the system's designer believes will help provide a high-quality initial recommendation solves this problem. The system will then collect actual information about the user/place.

2.2 Tourist Trip Design Problem

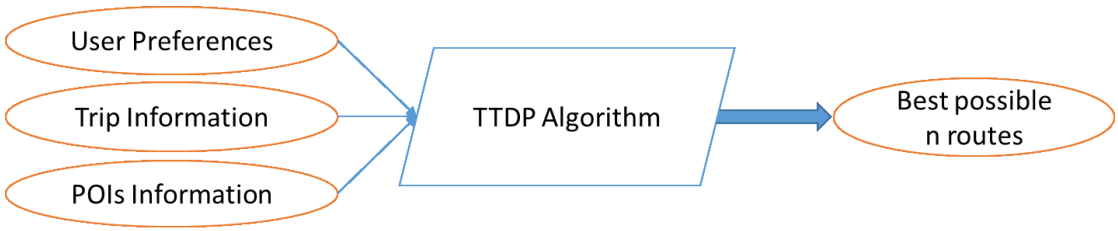

Figure 1. TTDP workflow: by getting the information about the user, trip, and POIs the algorithm will output the best possible n routes

One of the hardest questions to answer when traveling abroad for pleasure is where to go. This question is why traveling agencies have itineraries that tourists can choose from, which help to know what the most interesting places to visit are and how to get the most out of the trip. According to [18], the Tourist Trip Design Problem (TTDP) refers to a route-planning problem, solving the order in which the tourist visits the multiple Points of Interest (POI) in which he has interest.

When deciding on a day trip, the number of POI to be considered in that day are limited; there will be a lot of factors that affect the decision-making process. Those factors include but are not limited to the time required to visit each place, the opening hours of each place, the time spent getting from one place to another, and how eager the tourist is to go to a specific place [1].

According to [1], the TTDP is divided into two branches. The first is what we call a one trip design problem, where the aim is to find a route from A to B that provides the most satisfaction to the tourist, considering his limitations in time and budget. The second variation of this problem is the multiple trip version, where the tourist has more than one trip to visit a number of places. The aim here is to find the best multiple routes, where the tourist will visit each place only once.

In the next subsections, the two branches will be explored further, and their algorithms will be compared later on.

2.2.1 One Trip Problem

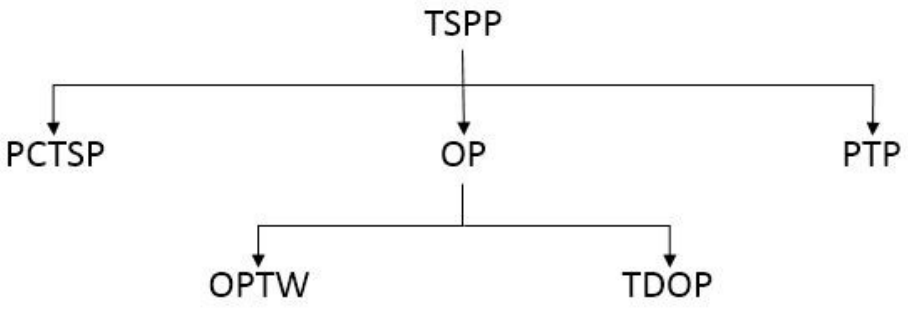

Figure 2. One Trip TTDP algorithm variants

The one trip design problem comes as a variant of the Traveling Salesman Problem (TSP), in the form of the two criteria model called the Traveling Salesman Problem with Profit (TSPP). In TSPP, the aim is to reduce the traveling time while maximizing the profit [19]. Currently, there are three models of TSPP, depending on how the algorithm handles the two criteria. The first one is the Profitable Tour Problem (PTP) introduced by [20], where the aim is to find the highest profit after subtracting the travel cost from the profits;

in other words, treating the two criteria as one criterion equal to the difference between the two. The second model is the Prize Collecting TSP (PCTSP) introduced by [21], where the aim is to have a profit threshold and find the shortest route while keeping the profit over the threshold. The last model is the Orienteering Problem (OP), firstly introduced by [22], where the aim is to have a time limit and find the route with the most profit while still within that time limit.

Out of the previous three models, the most wildly used in TTDP is the OP [1]. In the next subsection, we provide a more in-depth overview of OP and of its wide range of variants. This is not to undervalue the other models, however, in TTDP usually the time limit is known for the tourist, making it easier to work with OP instead of the other two.

2.2.1.1 Orienteering Problem

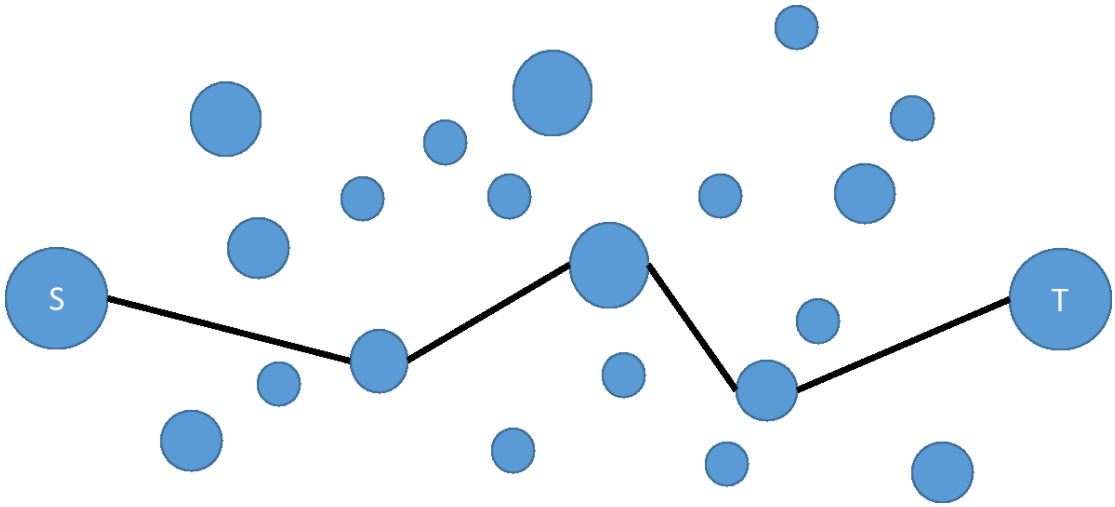

Figure 3. OP illustration - the size of the node reflects its profits

The orienteering problem is also known as the Selective Traveling Salesperson Problem [23] and the Maximum Collection Problem [24]. The OP problem can be

expressed as an edge-weighted graph G= (V, E), where each of its nodes has a profit value. Given a starting node s, a terminal node t and a positive time limit (budget) B, the goal is to find a path from s to t (or tour if s = t) with length at most B, such that the total profit of the visited nodes is maximized [1].

According to [25], an OP can also be expressed as an integer programming problem, as follows. Let N be the number of nodes labeled by 1, 2. . . N, where S = 1 and T = N, P_i be the profit of the visiting node i and $C_{i\,j}$ be the cost of traveling from i to j. For every path from 1 to N, if node i is followed by node j, we set the variable $X_{i\,j}$ equal to 1, otherwise equal to 0. Finally, U_i denotes the place of node i in the path. Equation (2.1) aims to find the route with the maximum total profit. We need a set of conditions that our equation needs to meet in order to be true. Constraint (2.2) is a constraint that the first node in the route is 1 and the last node is N. Constraint (2.3) ensures the path is connected and that each node along it is only visited once. Constraint (2.4) ensures the path is within the time budget. Lastly, Constraints (2.5), (2.6), and (2.7) ensure the path taken is the shortest one along nodes, and there are no sub tours. Figure 3 shows the OP after finding the path with the most profitable total within the time budget B.

$$\text{aimed route} = \max \sum_{i=2}^{N-1} \sum_{j=2}^{N} P_i \, X_{i\,j} \tag{2.1}$$

Such that:

$$\sum_{j=2}^{N} X_{1\,j} = \sum_{i=1}^{N-1} X_{i\,N} = 1 \tag{2.2}$$

$$\sum_{i=1}^{N-1} X_{i\,r} = \sum_{j=2}^{N} X_{r\,j} \leq 1, \forall\, r = 2 \ldots N-1 \tag{2.3}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} C_{i\,j} \; X_{i\,j} \; \leq B \tag{2.4}$$

$$2 \leq U_i \leq N, \forall\, i = 1,2, \dots, N \tag{2.5}$$

$$U_i - U_j + 1 \leq (N-1)\left(1 - X_{i\,j}\right), \forall\, i,j = 2, \dots, N \tag{2.6}$$

$$X_{i\,j} \in \{0,1\}, \forall\, i,j = 1, \dots, N \tag{2.7}$$

According to [23], an OP is a NP-hard problem, which means that for a solution to OP to be feasible, the number of nodes must be small. There exists a number of approximations for an OP, which are shown in Table 1. However, even though these time approximations are polynomial, which demonstrates how complex an OP is. That is why it is critical to limit the input of the problem when dealing with limited recourses (which is one of this thesis contributions).

Table 1. Approximation algorithms for the OP

Reference	OP Graph Type	Approximation Ratio	Time
[26]	Undirected	4	Polynomial
[27]	Undirected	3	Polynomial
[28]	Undirected	$(2+e)$	Polynomial
[29]	Directed	$O(\log n)$	Quasi-polynomial
[28]	Directed	$O\ (log^2\ O\ PT)$	Polynomial
[30]	Directed	$O(\frac{log^2 n}{\log\log n})$	Polynomial

There are a lot of extensions for the OP, however, only two will be discussed due to the other variants using other criteria than what is considered here (time and profit), and hence out of the scope of this thesis. The first is the Orienteering Problem with Time Windows (OPTW) [25]. In this extension, each node has a time window and it is only

possible to visit it during this window, where time windows could reflect hours of operation. The second extension is the Time Dependent Orienteering Problem (TDOP), where the time plays a factor when calculating the cost of travel [31]. TDOP is very effective when there is more than one option of transportation that can be used to move between nodes. In real life, this could be seen as deciding when the use of public transportation is faster than walking, or vice-versa. These extensions are driven from real-life needs and limitations. While they are considered direct extensions of the OP, there are a number of variants of the original OP. One of those variants is the Generalized Orienteering Problem (GOP), where instead of only having a profit value for each node, each one of them has a set of values that each represent a different aspect e.g. cleanness, taste, and friendliness [32]. The Multi-Objective Orienteering Problem (MOOP) is the multi-objective variant, where nodes are divided into categories e.g. malls, parks, and museums.

2.2.2 Multiple Trips Problem

Similar to the single trip problem, the original term for the multiple trip problem is Vehicle Routing Problem with Profits (VRPP), which is the multiple tours extension of TSPP [33]. The profit is collected through multiple vehicles, thereby ensuring that the routes provide the maximum profit under the constraint that no node should exist in more than one route. VRPP has three different implementation models. The most popular one is the Team Orienteering Problem (TOP), which is an extension of the OP to support multiple tours. The second one is the Capacitated Profitable Tour Problem (CPTP), which is also a multiple tours extension of PTP with the objective is to maximize the difference between the total collected profit (gained points, as each POI will have different points depending

on its importance) and the total travel cost. Next, the Prize Collecting VRP (PCVRP), which is again a multiple tours extension of PCTSP with the aim to minimize both the travel distance and the number of tours given a minimum profit threshold [34]. Lastly, the Vehicle Routing Problem with Profits and Time Deadlines (VRPP-TD) is a modified version of CPTP, where each node has a temporal deadline that needs to be met.

From the previous models, the most wildly used in TTDP is the TOP [1]. In the next subsection we provide an in-depth overview of TOP and its extensions, as it is the one that best illustrates TTDP.

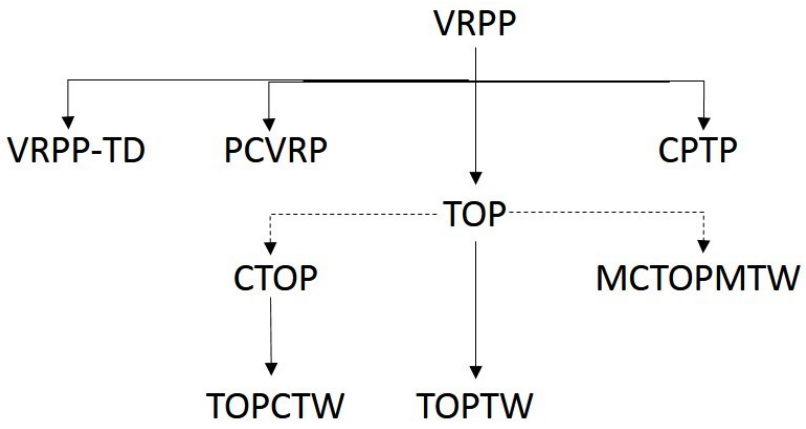

Figure 4. Multiple trips TTDP algorithms (dashed line denotes variants)

2.2.2.1 Team Orienteering Problem

TOP is an extension of the OP with an aim to find k most profitable tours, each within the time limit B, where each node in the graph must be visited at most once, except the start node S and the terminal node T. TOP is NP-hard and APX-hard since the OP is a special case of TOP. According to [25], an OP can be expressed as an integer programming problem by expanding the OP notation as follows: Given the integer k, let X_{ijm} be equal to

1 if node i is followed by node j in path m or equal to 0 otherwise. Let Y_{im} be equal to 1 if node i is visited in path m or equal to 0 otherwise, and let U_{im} be the position of node i in path m. With this notation we have the following equations:

$$\text{aimed route} = \max \sum_{m=1}^{k} \sum_{i=2}^{N-1} P_i Y_{im} \tag{2.8}$$

Such that:

$$\sum_{m=1}^{k} \sum_{j=2}^{N} X_{1jm} = \sum_{m=1}^{k} \sum_{i=1}^{N-1} X_{iNm} = k \tag{2.9}$$

$$\sum_{m=1}^{k} Y_{rm} \leq 1, \ \forall r = 2, \dots, N-1 \tag{2.10}$$

$$\sum_{i=1}^{N-1} X_{irm} = \sum_{j=2}^{N} X_{rjm} = Y_{rm}, \forall r = 2, \dots, N-1, m = 1, \dots, k \tag{2.11}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} C_{ij} X_{ijm} \leq B \ \forall m = 1, \dots, k \tag{2.12}$$

$$2 \leq U_{im} \leq N, \forall i = 1, \dots, N, m = 1, \dots, k \tag{2.13}$$

$$U_{im} - U_{jm} + 1 \leq (N-1)\left(1 - X_{ijm}\right), \forall i, j = 2, \dots, N, m = 1, \dots, k \tag{2.14}$$

$$X_{ijm}, Y_{im} \in \{1,0\}, \forall i, j = 2, \dots, N, m = 1, \dots, k \tag{2.15}$$

The objective function (2.8) is to maximize the total profit of the visited nodes. Constraints (2.9) and (2.10) ensure that each of the k paths starts at node 1 and ends at node N and that each non-starting, non-terminal node is visited at most once. Constraint (2.11) ensures that a flow of one unit can pass along each solution path connecting node 1 and node N, thereby ensuring that the path is connected. Constraint (2.12) ensures that the path meets the time budget. Finally, Constraints (2.13), (2.14), and (2.15) ensure that there are no closed sub-tours.

TOP with Time Windows (TOPTW) adds the temporal limitation to the nodes availability, as is the case with OPTW. Most of the TOP variations and extensions are only feasible with small graphs [1].

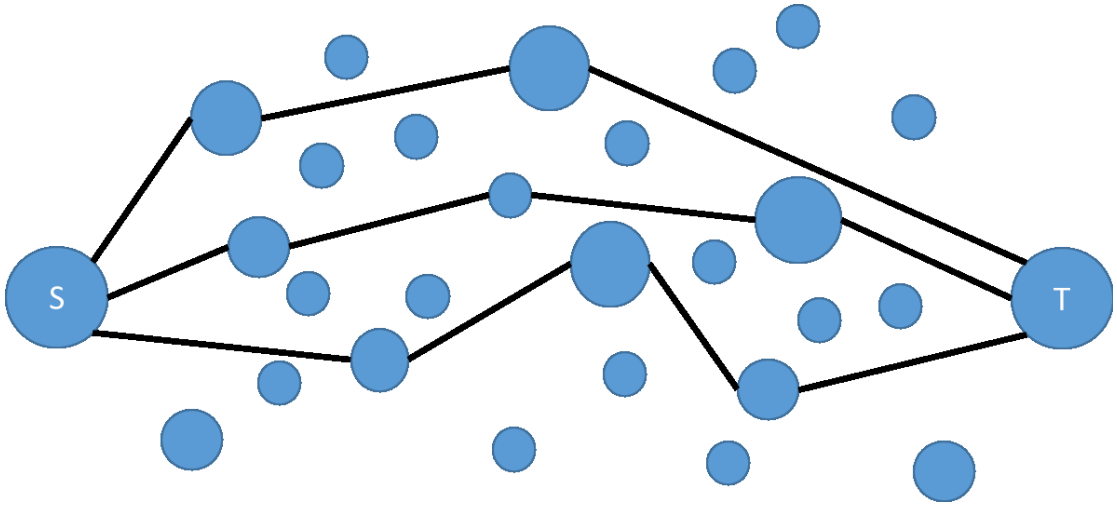

Figure 5. TOP illustration - the size of the node reflects its profits

2.3 Related Works

In this section we provide a detailed dissection of related works in both tourism RS and TTDP. The first section is an overview of current RS works, followed by sections covering three different implementations of TTDP. The focus will be more on TTDP since those applications are more related to the system presented in this thesis.

2.3.1 Current RS Approaches in Tourism

We begin with Flickr, the images sharing site. A number of papers have tried to analyze the website database to study tourist behavior. In [14], they try to cluster the images into time units and determine the tourist behavior during each time of each day of each season for each POI. Another work with the Flickr database [35] tries to rate POIs

according to the number of images and the time between images. It is noted that Flickr was not only used for CBRS but also for ABRS, as in both [36] and [5]. Both works try to mine the history of Flickr users.

The other type of location-based social networks (LBSN) that people have tried to analyze is the check-in social networks, e.g. Foursquare, Gowalla, and Facebook. In [12], Foursquare was mined to create a user-location matrix. In another work by the same author [11], Foursquare is used to find temporal preferences and temporal correlations for POIs. Furthermore, an in-town and out-of-town user-check-in matrix was done by [8] to study the behavior of tourists and locals around the same POIs. Lastly, [6] used data mining and machine learning algorithms to find the temporal influence correlations between LBSN users.

2.3.2 PERSTOUR

In [35], the authors proposed an algorithm called PersTour that recommends personalized tours using user preferences and POI popularity. Their algorithm depends on the information collected from Wikipedia and Flicker. This algorithm uses Wikipedia as a database for POIs and then uses the geo-tagged pictures of those POIs on Flicker to determine their popularity, based on the number of photos in that location which could be viewed as a simple CBRS. The algorithm uses the time stamp of photos to determine the average visit duration for each one of the POIs. The algorithm is not only a CBRS but a hybrid RS as it also collects the user's travel history from his time-stamped photos and uses visited POI time duration as a mean of measuring the user's interests in an ABRS way.

The algorithm can then be set manually to be CBRS or ABRS biased. After getting the hybrid RS score, the system will solve the OP integer problem to find the route.

In the evolution of their system, the authors in [37] compared CBRS and ABRS biases and demonstrated that ABRS biased recommendations outperform CBRS biased recommendations by using leave-one-out cross-validation. The argument is that the ABRS reflects the user more accurately. The second part of their evolution is to compare their OP system to baseline systems, namely Greedy Nearest (G-Near), Greedy Most Popular (G-Pop), and Random Selection (Rand). They have shown that, in general, their system outperforms all baselines in most cases, based on tour popularity, interest, precision, recall, and F1-score.

However, their work can be criticized in a number ways. The first is the usage of Wikipedia as a source of POIs, which is not a good choice since anyone can change Wikipedia pages. The second point is that their system depends heavily on user photos to determine the travel history of a user in greater detail, and the system fails if the user is not a heavy camera user. Even a small change in the user's picture taking behavior will affect their system. E.g. there may be a user who prefers to go to shopping malls but does not take photos there, however, his camera album is filled from the three times he went to a museum. In a more realistic sense, not all users have a rich traveling history. So, depending on only photos to get the user's preferences may not always be the optimal choice. Also, using the visit duration as a preference indicator is justifiable as people tend to stay more in places they like [35]. However, the means used in this paper are not reliable since the values were taken from the user's photos, therefore, those values are estimates.

2.3.3 ECOMPASS

In [38], the authors propose a variant to TDTOPTW called SlackRoutes. In their algorithm, they consider walking or taking public transportation as the possible ways of traveling between POIs. Their system intends to give the faster option between the two. They also implement the possibility of having lunch during the trip and the algorithm will therefore try to find a good restaurant along the route. Lastly, their system allows the user to choose any place as the start or end point of the tours. The algorithm will try to find the best k possible routes by first clustering the POIs; in their work, they use the global k-means algorithm [39] to get the clusters. The use of clustering is justified by two points: reducing execution time and identifying areas with high profits. The clusters will then be used as input for SlackRoutes algorithm, which starts by executing a local search to insert clusters in the routes. The algorithm then tries to escape local optima by shaking the routes (removing POIs that take a lot of time and generate little profits).

Forty-seven participants heuristically evaluated the mobile application created for the algorithm to determine usability. The mobile application was built to work with Android phones in a client-server configuration.

The system provides a comprehensive solution for TTDP for mobile users. While the system provides a multi-transportation solution, this only increases the complexity of the TTDP problem, as all possible modes of transportation must be calculated in real time, for each route between two POIs, in order to choose the best option. Also, the authors did not implement any kind of recommender system, but simply retrieved the recommendation for each POI from Trip Advisor. Lastly, their database only contains two cities, Athens and

Berlin, and they are using government databases, which are very detailed. Thus, it will be hard to find databases as detailed as those for every city in the world, which is the system's biggest drawback.

2.3.4 The City Trip Planner

In [40], the authors present a novel system and of the first applications to have a big scale recommender system that rely on a large database. However, recommender systems have evolved a great deal since 2011, which is when the paper was published. Also, the dataset is limited to five cities, which does not represent a real life solution but is appropriate for a proof of concept. The rating score for POIs is calculated through a questionnaire. Each POI has a location, opening hours, a visit duration, one out of five types and one out of five categories. The score is calculated based on the user type and the category preferences from the questionnaire, so depending on the questionnaire, each POI will have a score out of 36 (12 for type multiplied by 3 for category).

After calculating the score, the system then tries to find multiple tours by solving the TOPTW with a metaheuristic approach based on the Greedy Randomised Adaptive Search Procedure (GPASP) [41]. The system allows for the inclusion of lunch breaks by adding a phantom POI that will be used as a lunch break. The system does not allow for re-scheduling and only allows fixed start and end locations. The system only prints the plan for the user, and it is the user's responsibility to follow the plan or adjust it. Finally, once the trip is completed, the user completes a survey of six questions to share their degree of satisfaction with the system.

2.4 Summary and Comparison

Table 2. Related works comparison with our system

Work	RS sources	RS Criteria	TPP	TPP complexity Approximation	app
PERSTOUR [34]	Flickr + Wikipedia	# of images or time stamp	OP	O (C^N)	X
The City Trip Planner [37]	tourist offices databases	type, category, and the keyword search	TOP	O (KC^N)	Web
ECOMPASS [39]	gov. database + trip adviser	Trip Adviser score	SlackRoutes	O (KC^M)	Mobile
Our work	Flickr, google places, surveys	Popularity, context awareness	Balanced-OP	O (C^M)	Mobile

If we look at the current research done in the field of TTDP (Table 2), we can see that none has focused on the importance of the recommender system and its effect on the planning results that is why we needed to look further into what other people had done in e-tourism recommender systems even if they are not related to tour planning. Table 3 shows an overview of different implementations of e-tourism recommender systems with different complexities, inputs and types. We aim to make our system context-aware, to ensure that it outperforms even the state of art e-tourism RS (Table 3). Moreover, all their systems are suggesting pre-done routes to the user, which hinders any system that wants to be adaptive and robust. Lastly, due to the nature of the problem, the OP is always NP-hard, which that is why we want to modify it to make it much faster. This is done by clustering the input as we will see in section 3.3.1 and taking different criteria than what they have done thus far in section 3.2.2.

Table 3 Previous works in Recommendation comparison with our system

Work	CBRS	ABRS	RS complexity	RS complexity	LBSN	TTDP	application	RS Criteria	Remarks
(Han and Lee 2015)	√	X	O(n*m)	medium ter-2:	Flickr	X	X	# of images + day + time + season	
(Gao et al. 2013)	√	X	O(n)	simple ter-1	Foursquare	X	**Web**	user-location matrix.	
(Ference, Ye, and Lee 2013)	√	√	O(n*m)	medium ter-2	Foursquare + Gowalla	X	X	in-town and out-of-town user-check-in matrix.	
(Gao et al. 2013) 2nd paper	√	X	O(n*n)	medium ter-2	Foursquare + Facebook	X	X	temporal preferences + temporal correlations	
(Lim et al. 2015)	√	√	O(n)	simple ter-1	Flickr + wiki	OP	X	# of images or time stamp	start point must be POI
(Vansteenwegen et al. 2011)	X	X	O(1)	N/A ter-0	tourist offices databases	TOP*	**Web**	type, category, and the keyword search scores entered manually	not TOP, a Greedy Randomised Adaptive Search Procedure
(Gavalas et al. 2015)	√	X	O(1)	N/A ter-0	gov. database + trip adviser	TDTOPTW	**Mobile**	Trip Adviser score	not TDTOPTW but clustring and iterated local search procedure
(Zhang and Chow 2015)	X	√	O(n*k*d)	Complex ter-3	Foursquare + Gowalla	X	X	temporal influence correlations	
(Xu, Chen, and Chen 2015)	X	√	O(n*k*d)	Complex ter-3	Flickr	X	X	User History mining + and then build the user–user similarity matrix	User history must be rich
(Jiang et al. 2015)	X	√	O(n*k*d)	Complex ter-3	Flickr	X	X	author-topic model mining	similar to the previous
Our work	√	√	O(n*k*d)	medium ter-2	Flickr , google places, surveys	BOP	**Mobile**	popularity, weather, category , user preferences learning	the system take care of cold start

Chapter 3

3 Personalized Points of Interest Finder and Trip Designer

In this chapter, a detailed description of the algorithm and the system proposed in this thesis is presented. First, in Section 3.1, we formally state the research problem addressed in this thesis. Next, in Section 3.2, the proposed methodology is described in detail. The Augmented Reality interface is then described in Section 3.3. Other components of the system will be shown in Section 3.4, and finally, Section 3.5 presents the system flow as a summary of the chapter.

3.1 Objectives and Specifications

The main goal of the proposed system is to provide a software solution for the problem of trip design and scheduling. Taking into consideration the number of visitors and the duration of each visit, the system works as a recommender system with the aim of providing relevant suggestions for either places or entire trips. Furthermore, by analyzing the user's preferences and behavior, the system tries to make personalized recommendations. Another objective is that the system to be mobile and adaptive through context awareness, i.e. checking the weather conditions, traffic conditions, etc.

The contribution we want to make is that of an adaptive tool with automated responses and solutions for the user, as well as tracking and notifications of possible route changes. The accomplishments we want to fulfill are: the off-line and online availability of the system (database storage), the monitoring of trip states while generating analytics and

also the ability to notify the user when there is a need to reschedule the trip. We wish to accomplish all these things while also providing an improved reality interface.

3.1.1 Functional Requirements

The system must have a number of functional requirements for it to achieve our objectives. The list of functional requirements (FR) is as follows:

FR-1. The system should always be able to access the GPS module to get the user's current location.

FR-2. The system must keep a database of all nearby places to be able to access them offline.

FR-3. The system should mine online social networks for information about places.

FR-4. The system should search other e-tourism recommender systems to get more information for its places database.

FR-5. The system must group the places in each of the twelve directions relevant to the user's current location.

FR-6. The system should show the user a ranked list of the places in the direction he is facing.

FR-7. Given a destination and time limit, the system must generate a trip plan that finishes at the destination within the time limit.

FR-8. The system must keep track of the user during the trip to record the trip progress.

FR-9. If the plan can no longer be respected, the system must notify the user.

FR-10. The system must be able to modify the trip or generate a new one if the user can no longer meet the time limit constraint.

FR-11. The system must save trip stats and information to be able to restore it after a system restart.

3.1.2 Non-Functional Requirements

Our system also needs to meet certain Quality Requirements (QR) for it to achieve our goal:

QR-1. Performance: the system response time should be within a reasonable window.

QR-2. Availability: the system should be available at all times.

QR-3. Maintainability: the system must be easily restored to a working condition.

QR-4. Extendibility: the system should be easily integrated into other systems.

QR-5. Robustness: the system must cope with errors.

3.2 Proposed System

The system consists of four main parts: a recommender system, the proposed algorithm called the balanced orienteering problem, an augmented reality interface, and a

scheduler. We are going to discuss each one of them in the following subsections. We will then discuss the various components of the system.

3.2.1 Recommender System

To rate each POI, we need a ranking algorithm. The algorithm used in our recommender system rates POIs based on how many people have visited them as a first step. This can be done by extracting picture metadata from a social network that uses geolocation tags. Next, it links each POI with the pictures that have been taken there. Having this connection, as well as the metadata of the images, the system will re-create the trips that have been taken in that city [42]. By knowing how many users have visited a POI, the system can calculate how popular it is. In addition, to get an estimate of time spent in an individual POI, we calculate the time difference between the previous place visited and the next place visited in the same trip. Furthermore, we refine the obtained duration by subtracting the travel time between POIs.

$$POP_i = \sum_{u=1}^{n} V_u(POI_i) \tag{1}$$

$$V_u(POI_i) = {1, LOC_i \in \sum LOC_k \atop 0, otherwise} \tag{2}$$

$$D_i = \frac{1}{n} \sum_{u=1}^{n} T_u(POI_{i+1}) - T_u(POI_{i-1}) \tag{3}$$

In Equation (1), we calculate the popularity (POPi) of any POI as the number of visits (V) to that POI by all users, and Equation (2) is used to determine if the user visited that POI by checking if the POI location (LOC_i) is equal to the location of any of the user's photos (LOC_k). The duration (Di) of any POI is the average of the time of each trip (T_u) the user takes from the POI before POI_I to the one after it, in other words from POI_{I-1} To

POI_{I+1}. This time difference represents how much time the user has spent in POI_I, as in Equation (3). With popularity, duration along and user preferences, the algorithm has all the information needed to plan the trip.

3.2.2 Balanced Orienteering Problem

The proposed algorithm considers the time the tourist spent at each POI. It is important to note that such time has not been widely addressed in the implementation of various OP in the tourism industry as seen in section 2.2. In general, the time of a trip is divided into two categories: 1) the time spent moving from one POI to the other, and 2) the time spent sightseeing. Neglecting the time devoted to each individual POI sightseeing does not reflect real-life situations, where people spend a different amount of time at each place. One of the workarounds was done by [40] where The user will have slack time to spend sightseeing and the system will solve the transportation problem with the remaining time through orienteering problem algorithms.

Our variation of the original OP is called the balanced OP (BOP). Unlike the original OP, BOP takes into consideration both the time spent at each location and the traveling time. Moreover, BOP can work on mobile devices where resources are limited, while the original OP is an NP-hard problem.

BOP can be expressed as an integer programming problem, as follows: Let N be the number of POIs labeled by 1, 2. . . N, where the starting point S = 1 and the terminal point T = N, let P_i be the profit (Gained points) of visiting POI_i, D_i the average duration of the POI_i visit, and C_{ij} traveling time from i to j. For every path from 1 to N, if POI_i is followed by POI_j, we set the variable X_{ij} equal to 1 if the route goes from i to j or equal to

0 otherwise. Finally, U_i denotes the place of POI_i in the path. Equation (4) aims to find the route with the maximum total profit P_i (most satisfactory route) and the maximum time spent sightseeing D_i, it will start from i=2 since we consider the user current location have no profit since it is the starting location and the user will leave it as soon as the trip starts. Constraint (5) is a constraint that the first node in the route is 1 and the last node is N. Constraint (6) ensures the path is connected and each node along it is visited only once. Constraint (7) ensures the time spent moving from one location to the next and the actual sightseeing time are within the time budget. Lastly, Constraints (8), (9), and (10) ensure that the path is the shortest one along the nodes and that there are no sub tours. Equation (11) is how the profit is calculated, in this case meaning the popular places visited, in which the user is interested.

$$\text{aimed route} = \max \sum_{i=2}^{N-1} \sum_{j=2}^{N} P_i \, D_i \, X_{i\,j} \tag{4}$$

Such that:

$$\sum_{j=2}^{N} X_{1\,j} = \sum_{i=1}^{N-1} X_{i\,N} = 1 \tag{5}$$

$$\sum_{i=1}^{N-1} X_{i\,r} = \sum_{j=2}^{N} X_{r\,j} \leq 1, \forall\, r = 2 \ldots N-1 \tag{6}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} (C_{i\,j} + D_i) X_{i\,j} \leq B \tag{7}$$

$$2 \leq U_i \leq N, \forall\, i = 1,2, \ldots, N \tag{8}$$

$$U_i - U_j + 1 \leq (N-1)\left(1 - X_{i\,j}\right), \forall\, i,j = 2..N \tag{9}$$

$$X_{i\,j} \in \{0,1\}, \forall\, i,j = 1, \ldots, N \tag{10}$$

$$P_i = POP_i * User\ Preferences\ Matrix \tag{11}$$

Since OP is NP-hard, we propose a further improvement on our algorithm to decrease the processing time. While this enhancement impacts the precision of the result, it greatly decreases the processing time. This is applied by doing a global k-means clustering algorithm [39] with Equation (12) as a clustering criterion. We increase the number of clusters N until we achieve a Mean Absolute Distance (MAD) that is less than what the average human walks between any two POIs k and j. according to [43], the average preferred speed for humans is 4.8 km/hour and the average walking distance is 1.6 km, in this thesis we are using 5 km/hour and 1 km respectively for simplification reasons. Figure 7 shows the city of Ottawa divided into small clusters, where each represents a walkable area. The aim of this step is to group nearby POIs together in a cluster, with each cluster having a walkable distance between its points, so that the algorithm will visit them as a group and visit each cluster only once. Because of how clustering is done, we can assume that a human would prefer to use some type of transportation to get from one cluster to the next, and walk between POIs in the same cluster. We will discuss clustering with more details in 3.3.1.

$$MAD = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{k=1}^{M}\frac{1}{M-k}\sum_{j=k}^{M}|POI_k - POI_j| \quad (12)$$

To reduce the computational time of the algorithm we divide the problem in two steps instead. The first step is to solve the BOP for clusters instead of individual POIs, and the second is to solve BOP for the members of the clusters obtained from the first step. This leads to a decrease in the complexity of the algorithm, as it becomes an equation for the number of clusters instead of POIs, which is significantly smaller. For the algorithm to work with clusters, we redefine the profit of the cluster P_c and the duration of the cluster D_c in Equation (4) to be:

$$D_c = \sum_{i=1}^{N} D_i \quad (13)$$

$$P_c = \frac{1}{D_c} \sum_{i=1}^{N} P_i \, D_i \quad (14)$$

It is important to mention that our proposed approach route selection is based on a the metaheuristic Greedy Randomised Adaptive Search Procedure (GRASP) [44]. Which showed that it could fined the solution for the algorithm much faster by considering move valued routes first.

3.2.3 Scheduler

The scheduler will keep track of the user's movements during the trip by periodically identifying the user's location. The scheduler will check if the user is following the trip plan closely and will adjust the plan accordingly, to ensure that the user reaches his destination in time. This simple task is vital since people will not follow the plan, for many reasons. People might spend more time at a given POI than expected, while they might leave sooner if they do not like the place. Having this dynamic nature that changes with the user's actions will help the planner be more practical.

Alternatively, the scheduler works as a mean to obtain feedback concerning the user's preferences. Lastly, because of the limited resources, the scheduler will have to delete out-dated information and obtain missing information, e.g. when the user moves

from one city to the next, the scheduler removes the information about the old city and gets the new city data.

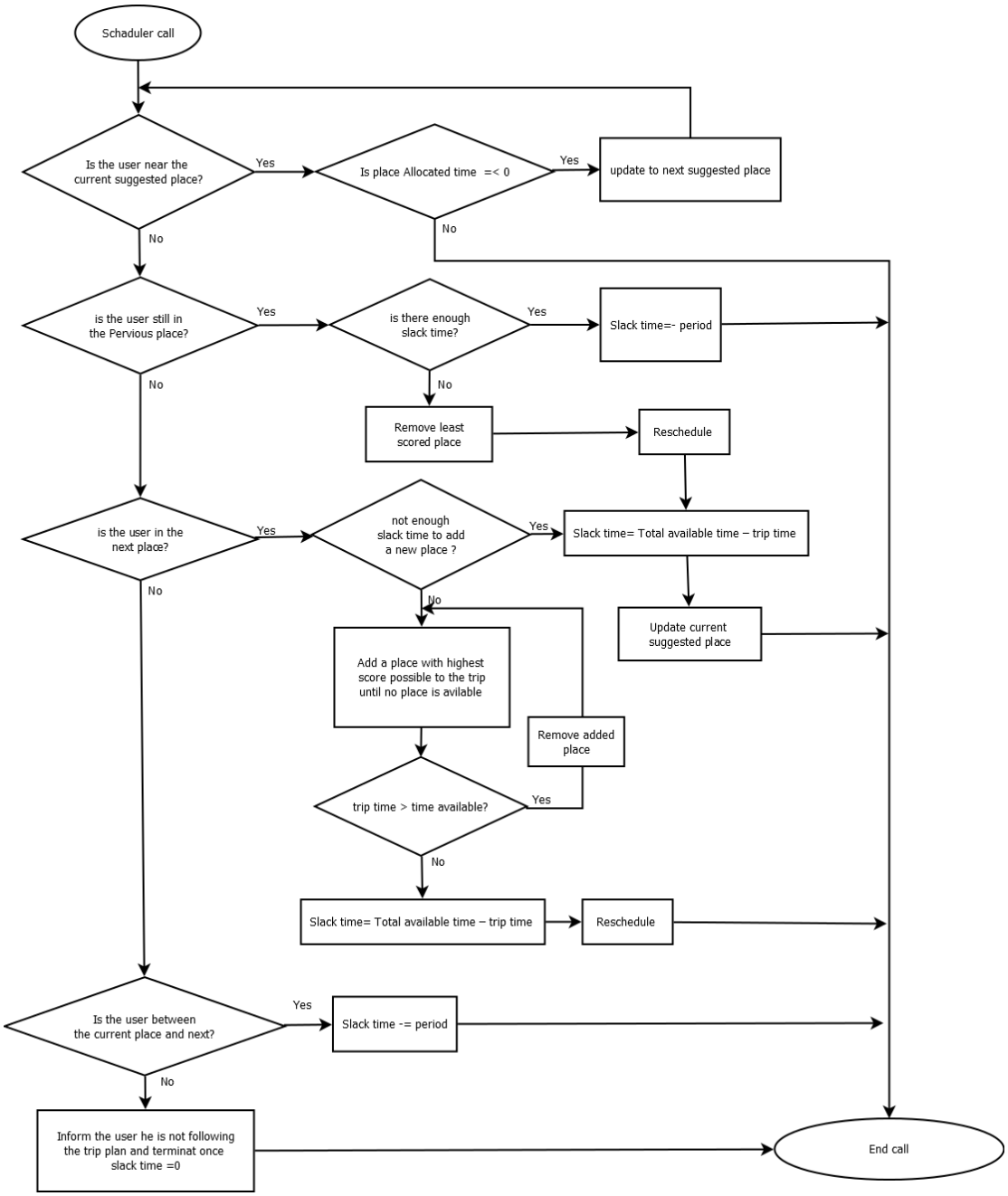

Figure 6. Scheduler flowchart

3.3 Components

In this section, we explain the different parts of the system that will work together to form the system.

3.3.1 K-means Clustering

The aim of this component in the system is to use an unsupervised learning algorithm to help the system break down the city map into small walkable neighborhoods. The k-means clustering is done by defining k number of central points, where each point is in the center of an area, in other words a cluster, and all the other points of the data set are connected to the nearest central point. Figure 7 shows how the City of Ottawa, Canada, is divided into small areas representing the walkable neighborhoods. This is done by loading a list of all the POIs of the city. This list then will be clustered into a neighborhoods list, with each neighborhood will have its own POIs.This will help the system find the most interesting walkable neighborhoods along the path instead of simply finding points.

The end user will benefit from this by a balanced trip, where the walking distance will be as short as possible, the number of times that transportation is needed is minimized and the time spent sightseeing is maximized.

In our system the centroids will be initialized randomly for each added cluster, then it will be reassigned for each clustering iteration (which equals the number of clusters). The assignment with the least mean error will be that cluster fixed point for the rest of clustering. We start with one cluster, then we add clusters one by one until MAD in equation (12) in section 3.2.2 become less than 1.

Once clustering is complete, the system will be able to end the offline cycle and the heavy processing; all that is left at this point is the online system. [39]. Appendix A shows the results of each step in the global k-means clustering for the cities of Ottawa, New York, and London.

3.3.2 Location Manager

The need for a location managing system is fundamental in e-tourism automation.

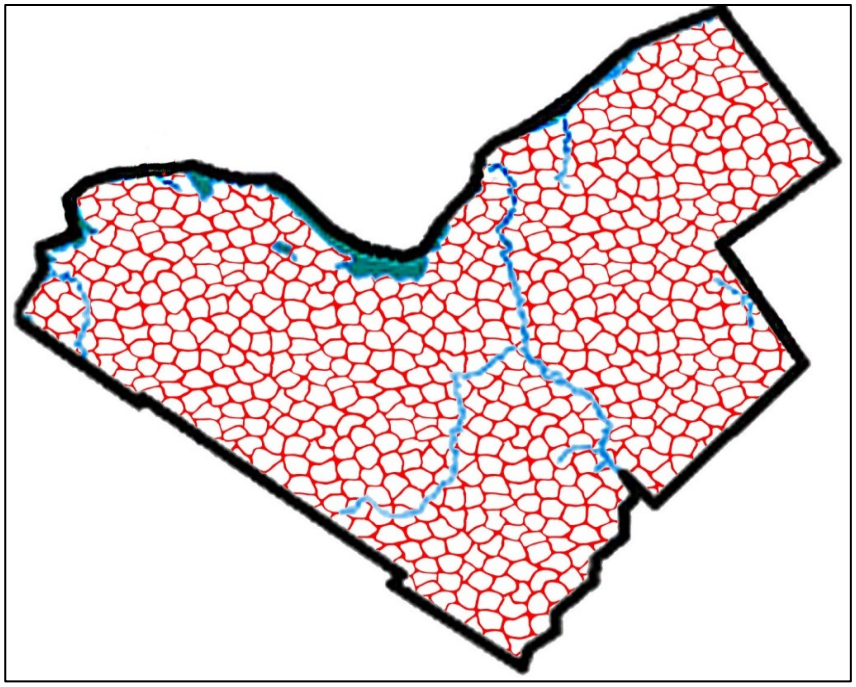

Figure 7. City of Ottawa map divided into multiple walkable neighbourhoods. Each neighbourhood is a cluster and each attraction belongs to the cluster with the nearest central point.

Instead of the user having to enter his current location, the manager can approximate the location of the user using the Global Positioning System (GPS). The system can also increase its precision by using the Received Signal Strength Indicator (RSSI) [45], which uses the cell towers or Wi-Fi routers as locators, instead of satellites.

The manager also needs to have a gyroscope so that the system can determine the direction in which the user is facing. While the user could hold the mobile device in different positions, the bearing reference that we are going to use is the direction of the back camera with respect to the true north.

After determining where the user is facing, the system will start by finding clusters that have the same bearing as the user. This is done by finding where the current location of the user is. And his destination. Then finding all clusters that the user could go through while reaching the destination within the time limit. The shape of the search area will be like an ellipse, where the middle will have more clusters than the two ends.

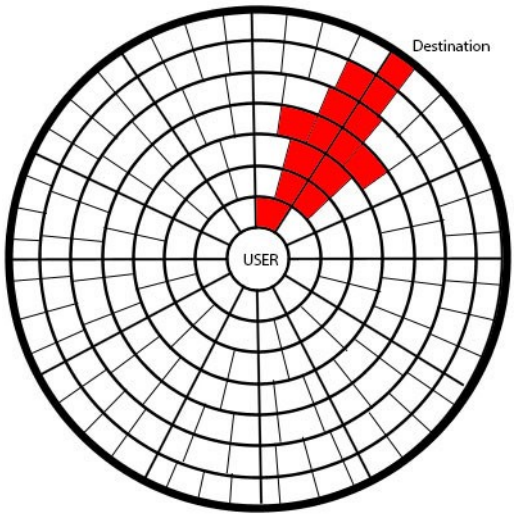

Figure 8. The search area for attractions that are in the direction the user faces. The red cells represent the clusters that the system will consider during the search.

3.3.3 Data Storage Management

We need to keep data in the system for three reasons. The first is to allow the system to work offline in the case where there is no Internet connection. The second is to keep the

system's current status if the system is terminated or interrupted. Lastly, we need to keep the user's information and use it for the next run. For each of these reasons, we use a different method to keep the data depending on the scale and sensitivity as follow; To have the possibility of using the data offline, a database solution is chosen based on light SQL to handle the large size of the data. To ensure that the system's status is saved, the host operating system handlers is used in order to manage the data since it is very sensitive and the operating system will take care of any error. Lastly, the user's information will be saved as comments in a text file, since it is relatively small and not so sensitive, so we are choosing the least expensive method.

3.3.4 API Handlers

Since our system retrieves a lot of data from different services and we want the system to be a specific-API independent, we have all the data retrieval handled in subsystems that talk to both our system and those APIs. Those handlers will initiate the call to the APIs and retrieve the data in their raw format. Next, they will process the data

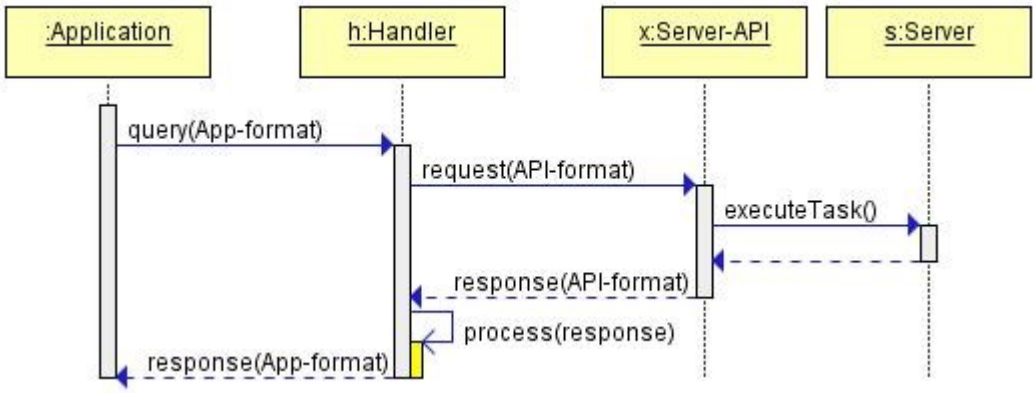

Figure 9. API handler interaction diagram

into our system format and pass it to our system. Figure 9 shows how handlers interact with APIs.

3.3.5 Preferences matrix and survey

Based on [15], we tried to use a survey to get the user preferences for different weather conditions where the scores are fractions of 1. This survey have the following questions:

- "Do you prefer to visit open-environment places or ones where you stay inside?" where it will give indoor places the score of question, and outdoor ones will get (1 – the score)
- "Will rain affect your trip planning?" if the score is less than 0.5, the score will be subtracted from each outdoor place during rainy days, if it is above 0.5 the (score – 0.5) will be subtracted from each indoor places.
- "Will snow effect your trip planning?" it will be as the rain effect but on snowy days.
- "Will humidity effect your trip planning?" it will be as the rain effect but on snowy humid days.
- "Will hot weather effect your trip planning?" it will be as the rain effect but on snowy hot days.
- "Do you prefer to stay long in one place or move around from one place to the next?" will multiply (score – 0.5) by the suggested duration found by equation (3) in section 3.2.1.

The indoors places list includes: art galleries, bowling alleys, museums, casinos and malls. The outdoors one includes: amusement parks, aquariums, campgrounds, parks, stadiums, and zoos.

The matrix will also be filled by reverse geo-tagging of the user camera album, to find which type of places the user prefer. Additionally, after each run of the system, a feedback loop will change the preferences depending on what user did in the trip.

3.4 System Flow

As an overview, Figure 10 shows the steps the system takes and how it moves forward to create the data and present it to the user. The system's architecture is described below:

1. The system will start by getting place preferences through either user tourism history mining or, if that is not sufficient (did not reach the threshold of 100 unique locations), with the help of a survey.
2. The system also will start finding nearby places with the use of an online maps service based on the user location.
3. We modify the nearby list by using the preferences we collected from step 1 to create a contextual POI list.
4. The list is then sorted based on their distance from the user and stored in a database.
5. The system will mine the social media by sending geo-tagged requests for images taken at each POI. This is done to find other tourists behavior around those POIs and create a rated list based on equations (1), (2) and (3).
6. The rated list will then be sorted in descending order.

7. The list will then be clustered to create a new clustered list that is sorted based on the clusters rating rather than the individual POIs. This clustered list is used in two main subsystems:
    a. Using the gyroscope to do a bearing analysis for the surrounding view augmented reality subsystem.
    b. Collect the trip information to plan a trip using BOP in the trip planner subsystem.

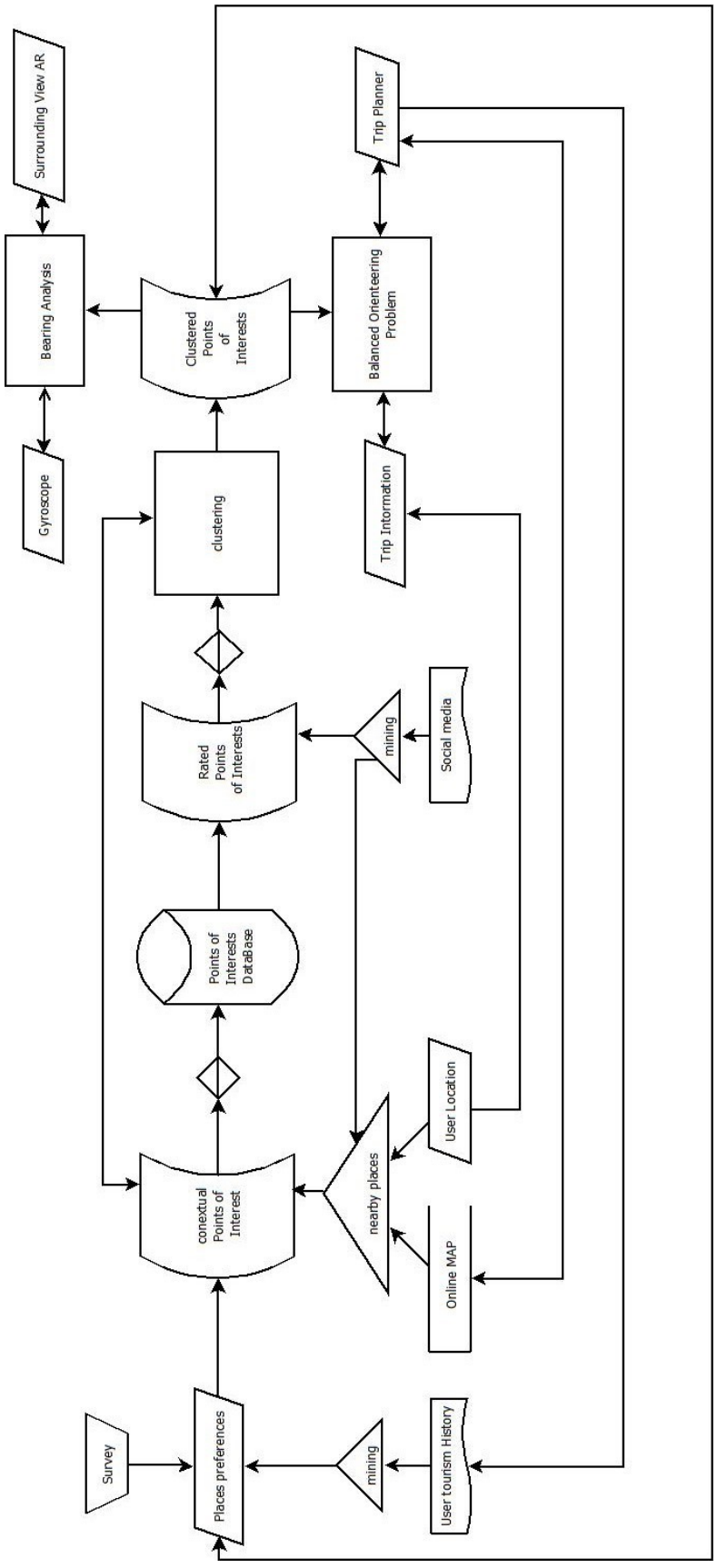

Figure 10. System overview

Chapter 4

4 System Implementation

We implemented our system into an android application with two parts, an online part and an offline part. The offline aspect handles the loading and clustering of the data; it will also handle the RS. The online aspect will handle the rest of the TPP, AR, and the scheduler. This workflow has been chosen because if there are two or more trips within the same city, we do not need to process the offline part again. It will also reduce the load to the online part and make it faster to respond.

4.1 Offline System

The offline subsystem will execute if the scheduler triggers that the database is out of date, which is done by simply checking the location of the user against the initial database location. If there is no database or the database is not the current city of the user, the system will start by downloading a list of POIs in the current city of the user from any POI provider. Currently, our system uses Google places.

Once the list is initialized, the system will then access Flickr and start mining the metadata of the pictures that have geotags equal to any of the POI list's geolocations. After getting the list of the associated images, the system will use Equations 1 to 3 to add the popularity and duration to each POI. Figure 14 shows the structure of our database.

The next step is the clustering of POIs, where the global k-means clustering is done. The system adds each POI to its cluster, then saves the list in a SQLite database and terminates itself by triggering that the database is up-to-date.

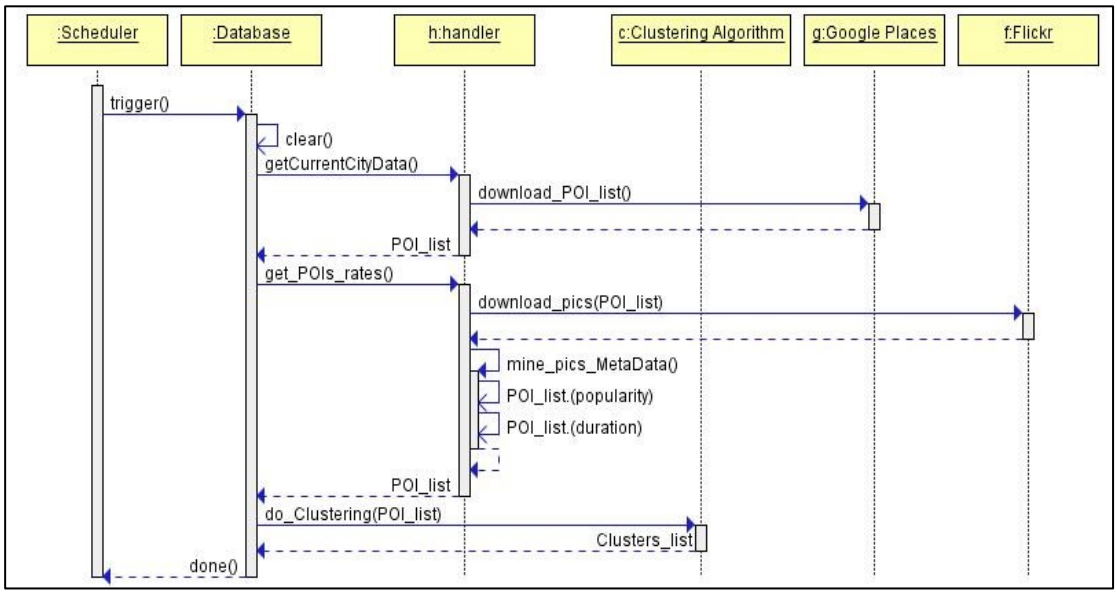

Figure 11. Offline system flowchart

4.2 Online System

The online system has two discrete parts. The first is the trip planner, which will be discussed in 4.2.1. The other part, which is an implementation of an augmented reality e-tourism recommender system, is described in 4.2.2.

4.2.1 Trip Planner

The online subsystem will first check if the database is up-to-date, determine where the user is currently, and see if there is a trip in progress. If the database is up-to-date, the system will load the SQLite tables to the POI list. Otherwise, it will trigger the offline subsystem.

If there are no trips in progress and the user wants to start a trip, the system will ask the user for the duration of the trip and his final destination, as shown in Figure 19-a. If the

destination is left empty, the system will assume that the user wants it to be a round trip, meaning that he intends to go back to where he currently is.

Once the time budget is known and both the starting and ending destinations are set, the system will calculate the cluster duration (Dc) and profit (Pc) for each cluster, using Equations 13 and14, and solve BOP for the clusters.

Once the BOP is solved for clusters, the BOP will be calculated again but this time with the POIs that existed in the clusters of the first run. Once done, the trip information will be saved, and the system will work in "trip in progress" mode.

During the trip, the system will schedule alarm for itself at a 10-minute intervals. Every time the alarm sounds, the system will check the user's location and determine the trip progress.

If the user is moving too fast, the system will prompt the user for the possibility of adding new POIs through the path. On the other hand, if the user is moving slower than what the system predicted, the user will be asked if he prefers to delete the least wanted POI (Pi = minimal) or re-plan the trip so that he can respect the time limit.

In the case of any of these situations, the scheduler will change the user's preferences matrix to increase or decrease the preference of this kind of POI in the future.

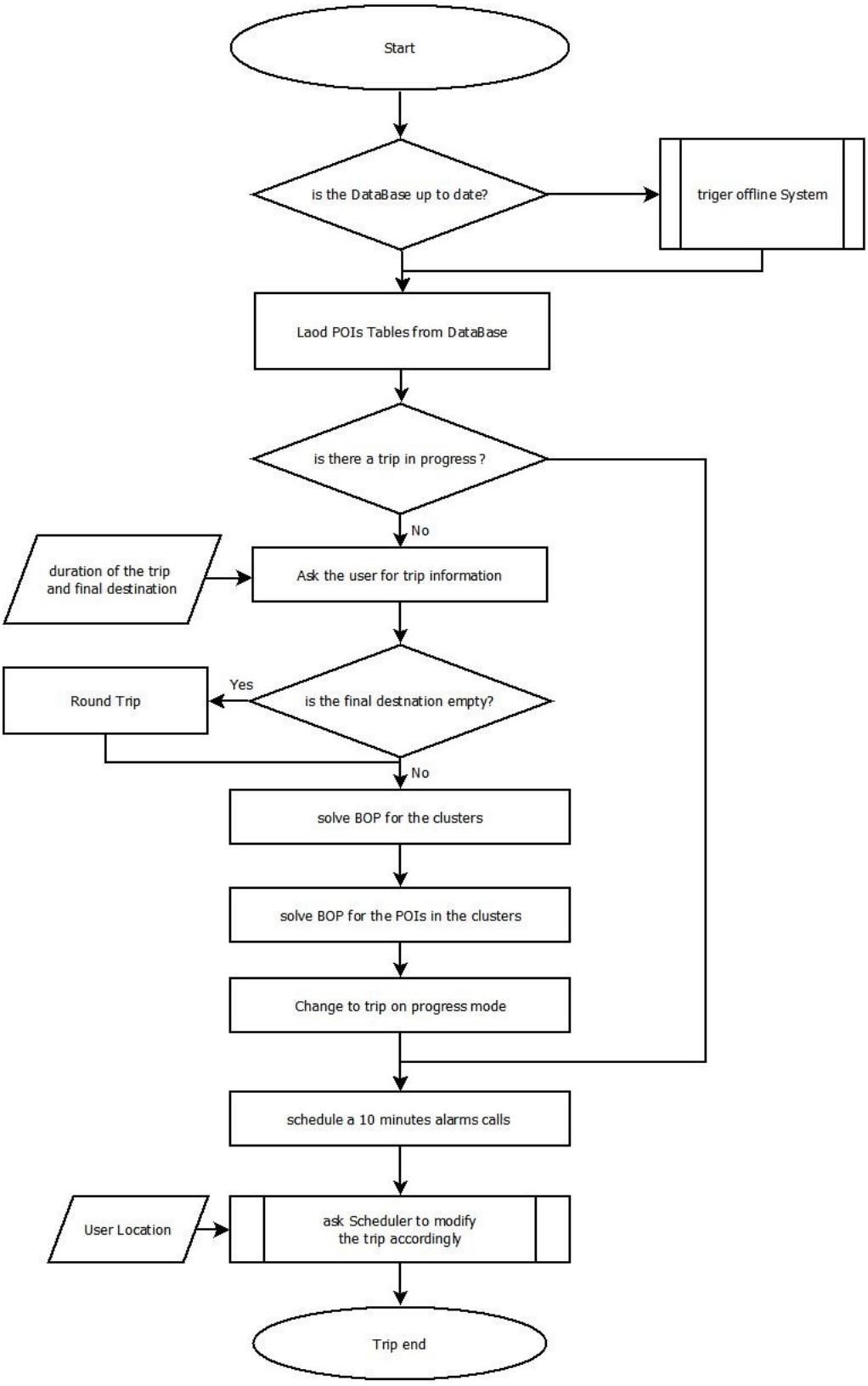

Figure 12. Trip planner online system flowchart

4.2.2 Surround View

In this part of the online system, we try to implement an augmented reality emulator. We use the sensory data from both the gyroscope and GPS to create a virtual surround view of our recommender system using images as explained in section **Error! R**

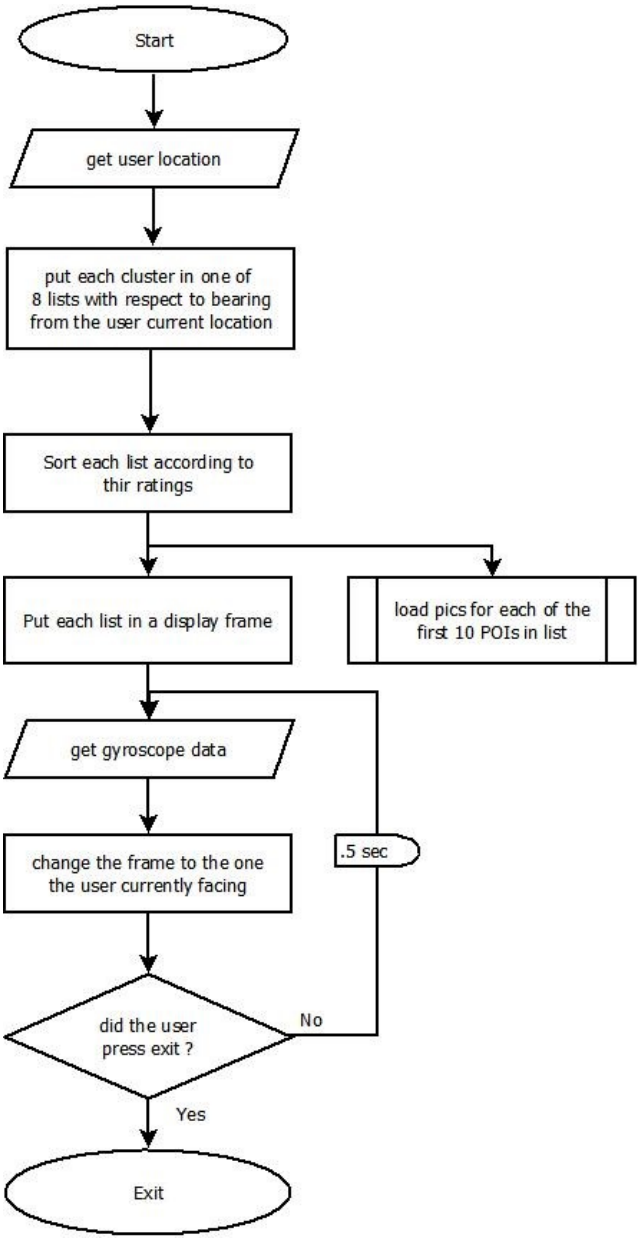

Figure 13. Surround view flowchart

eference source not found..

As shown in Figure 13, the system will start by identifying the user's current location. This is essential if we what to determine the distance and direction of each place with respect to the user. The next step is to analyze our database and add a bearing and distance information to each point of interest. We then divide them into eight cardinal directions, the four known ones e.g. north, east, west, and south and their intermediate directions. While being able to identify the exact direction in which the user is facing is ideal, it creates a lot of computational overhead as for every degree the user rotates, the system must regenerate the list again and recalculate the best rating. Having those eight cardinal lists give us a 12.5 degrees maximum error threshold, which may seem like a lot but if we consider the gyroscope error and the user's movement, the closer to the place the user gets the more the error decreases.

Once we have the lists we start multiple asynchronous tasks in order to download pictures for the top ten POIs in each list. This may take several seconds, depending on the speed of the Internet connection. The system will also create eight display frames, or XML files, each containing information about a specific location and a spot reserved for the picture to be inserted once it is loaded.

Finally, the system enters an infinite loop state, where it will get the gyroscope data to determine in which direction the user is facing and change the displayed frame to the correct one. It will wait for half a second and begin the loop again. We introduced this half-second delay to adapt to the user's movements and make the transaction between the frames smooth since the gyroscope is very sensitive to motion.

The system will exit the loop when he presses the back button. The system will go to the main screen again and wait for the user's next action.

Table 4. Example of a user preference matrix

Time	Condition	amusement park	aquarium	art gallery	bowling alley	campground	museum	park	casino	stadium	zoo
morning	clear	0.53	0.95	0.69	0.5	0.78	0.55	0.91	0.46	0.62	0.95
	rainy	0.37	0.47	0.85	0.42	0.39	0.75	0.08	0.19	0.37	0.36
	snow	0.2	0.12	0.81	0.39	0.2	0.87	0.3	0.19	0.23	0.5
	hot	0.98	0.66	0.94	0.46	0.83	0.87	0.98	0.22	0.83	0.71
	humid	0.97	0.66	0.7	0.24	0.6	0.62	0.96	0.44	0.92	0.83
day	clear	0.51	0.95	0.79	0.24	0.58	0.92	0.62	0.46	0.84	0.77
	rainy	0.04	0.27	0.21	0.01	0.13	0.28	0	0.36	0.13	0.06
	snow	0.38	0.48	0.78	0.34	0.18	0.56	0.43	0.4	0.33	0
	hot	0.68	0.88	0.09	0.47	0.58	0.49	0.6	0.41	0.94	0.96
	humid	0.95	0.83	0.29	0.14	0.75	0.38	0.87	0.2	0.89	0.86
night	clear	0.74	0.74	0.56	1	0.56	0.95	0.63	0.91	0.57	0.66
	rainy	0.07	0.36	0.87	0.61	0.08	0.88	0.29	0.66	0.12	0.31
	snow	0.1	0.24	0.93	0.69	0.08	0.66	0.01	0.66	0.34	0.27
	hot	0.69	0.58	0.85	0.63	0.54	0.86	0.65	0.8	0.93	0.62
	humid	0.88	0.58	0.78	0.77	0.67	0.89	0.7	0.91	1	0.51

4.3 User Preferences Matrix

This matrix consists of types of POIs versus weather conditions. Conditions include clear, rainy, snowy, hot, humid, and cold, and each condition has a subcategory of morning, day, and night. POI types include amusement parks, aquariums, art galleries, bowling alleys, campgrounds, museums, parks, casinos, stadiums, and zoos. Each cell in the matrix has a value between zero and one.

This matrix helps reflect the user's preferences regarding different types of POIs during different weather conditions [15]. This context awareness will increase the RS personalization and improve its suggestions as people's preferences change in different conditions. The matrix will be initialized when the user answers a simple survey as described in section 3.3.5. The scheduler will then update it with each trip to reflect the changes in the user's preferences.

4.4 Data Structures

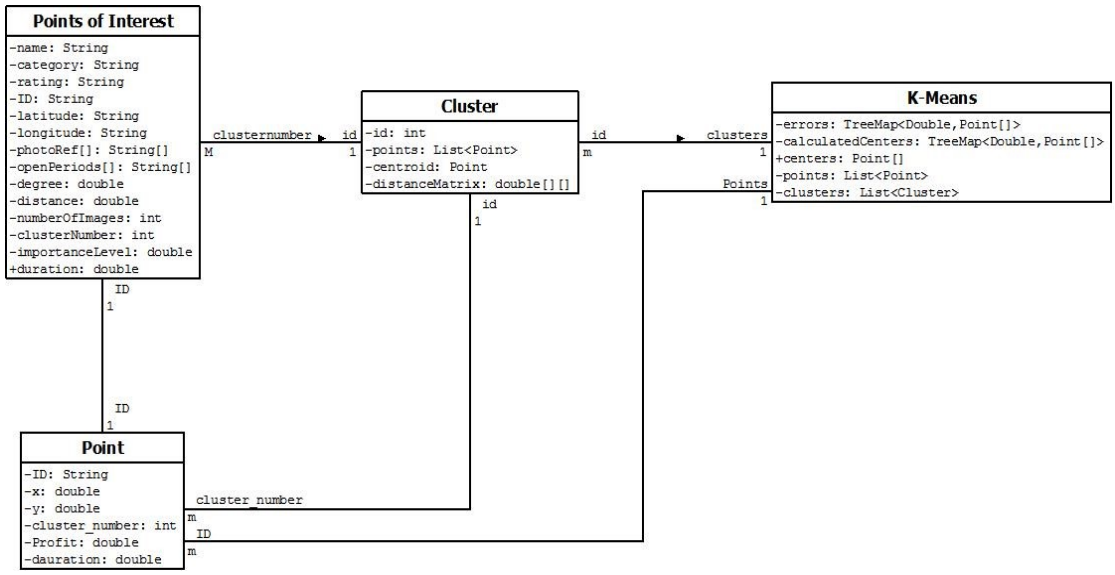

Figure 14 Relationship diagram of the data structures

In this section, we are going to provide a short description of each of the data structures in our application. Figure 14 shows the relationship diagram for the different objects used in this research.

4.4.1 Cluster

A cluster is a representation of points with a minimum distance to one of the means of the k-means algorithm. Table 5 displays the information that each cluster holds from its centroid, which represents the mean to the distance matrix that contains the distance between each point in the points list and the centroid.

Table 5 Cluster information

Name	Type	Description
ID	Int	The cluster id
points	List<Point>	A list of all points belonging to the cluster
centroid	Point	The location of the center of the cluster.
distanceMatrix	double[][]	The matrix that contains the distance between the centroid and every point in the cluster.

4.4.2 Point

Points are used to represent each point in our x-y plane, where all the calculations for the k-means clustering and route planning happen. A point can represent a POI or a centroid. It also helps us to simply delete the representing point for any POI to exclude it from the algorithms without deleting its information. Table 6 shows the simple structure of a point.

Table 6 Cluster point information

Name	Type	Description
ID	Int	A point's identifier. Usually the same as the POI's ID
x	double	The longitude coordinate of the point
y	double	The latitude coordinate of the point
Profit	double	The profit of visiting this point if it is a POI.
Duration	Double	The expected duration of stay in the point if it is a POI. Through the usage of equation 3 in section 3.2.1

4.4.3 Point of Interest

The points of interest information database contains a lot of information about each and every place. Table 7 shows the variables of the database together with a brief description of each variable.

Table 7 Point of interest information

Name		Type	Description
Name		String	Name of the point of the interest.
Category		String	The category the POI belongs to such as: museums, parks, casinos, stadiums, and zoos..
Rating		String	The rating of the POI from 0 to 5 where 0 is dissatisfactory And 5 is most satisfactory
ID		String	The unique identification of the POI
Latitude		String	Location coordinates
Longitude		String	Location coordinates
photoRef[]		String[]	An array of URLs of image of the POI fetched from Flickr
OpenPeriods[]		String[]	An array of the daily opening hours
Degree		double	The degree of bearing from the north
Distance		double	The distance between the user and the POI
NumberOFIamges		int	The number of the Flickr images that are taken in the location of the POI
ClusterNumber		int	The id of the cluster the POIs belongs to.

4.5 APIs

The application communicates with different web services through their application programming interfaces (APIs). For this we implement multiple handlers to be the links between other APIs and our system. In this section, we discuss each API we are using as well as the APIs' responses. Most the APIs respond with a JavaScript Object Notation (JSBN) file, which we decode to get the information we need. Appendix B shows all the responses to the calls in the next sections.

4.5.1 Google Places

The Google Places API is essential for our application to work each time our database become irrelevant to the user's current location. At the start, the handler will ask the API for all the locations within a 50 kilometers radius from the user. It will then ask for the information of each individual place.

An example of a search call to find the Ottawa Parliament is:

https://maps.googleapis.com/maps/api/place/radarsearch/json?location=45.4244041,-75.6785358&radius=50000&keyword=Parliament&key=**GoogleAPIkey**

In this call, we ask the API to respond in JSON format. The location is the center of the search circle (in this example the University of Ottawa). The radius here is in meters. The keyword refers to a keyword search with Parliament as the keyword. The key value is the Google API key that is associated with our application.

The response will contain a unique ID for each place that meets the search criteria. The handler will again call the APIs to ask for each individual place. An example of the second call will be:

```
https://maps.googleapis.com/maps/api/place/details/json?placeid=ChIJ75TkT_8Ezkw
Rbp_CYE-1awI&key=GoogleAPIkey
```

Where again we ask the API to respond in JSON and we give it the place's unique ID and our Google API key.

4.5.2 Google Maps

Unlike the Google Places API, the Google Maps API is not essential. We only use it to retrieve information about the route between two places if they are far away, and the user may need to use transportation to travel. The code below is used to get the route information:

```
https://maps.googleapis.com/maps/api/directions/json?origin=location1&destination=
location2&key=GoogleAPIkey
```

Location1 is the coordinates of the first location and location2 is the coordinates of the second location.

4.5.3 Flickr

We use the Representational State Transfer (REST) API from Flickr. We use this API to get metadata about the pictures that were taken in a specific place. We use the longitude and latitude coordinates of the place, together with a textual searcher, to filter unrelated pictures that are usually the name of the POI itself. A call to the Flickr API is shown below:

```
https://api.flickr.com/services/rest/?method=flickr.photos.search&api_key="FlickrKey
"&"+poi.getLatitude()+"&lon="+poi.getLongitude()+"&text="+URLEncoder.encode(
poi.getName(), "UTF-8")+"&per_page=5&format=json
```

4.5.4 Yahoo Weather

Yahoo Weather uses an YQL interface that is similar to a SQL statement. For example to get the weather forecast in Ottawa we sent this query to the API:

```
select * from weather.forecast where woeid in (select woeid from geo.places(1) where
text="ottawa, on")
```

This will give us all the weather forecast information for Ottawa in a JSON file. This information will help our system better suggest places to the end user.

4.6 Android OS Services

In this section, we are going to discuss the four main operating system (OS) services we use in our application. Those services keep track of the application and provide an easy interface to the device hardware. Those services are as follows:

4.6.1 Location Manager

The Android Location Manager is a service provided by the OS that will provide parameters such as what to use to get the location and the location update frequency. The application then uses the location listener object to get the current location. There are three types of location provider modes: GPS_PROVIDER, NETWORK_PROVIDER, and PASSIVE_PROVIDER. While the first two were discussed in Section 3.3.2, the last one will use the software to get the location instead of the hardware.

4.6.2 Sensor Manager

The Android Sensor Manager is a service provided by the OS that will give the type of sensor it wishes to use and the frequency of updating the location. The application then uses the sensor listener object to get the current bearing. Despite the big range of sensors available, we only use the ACCELEROMETER sensor and the MAGNETIC_FIELD sensor to get the bearing of the device from its rotation vector.

4.6.3 Alarm Manager

It is important to take countermeasures to the fact that the user will usually not have the application running in the foreground. This is why our application will schedule itself to run periodically. As was discussed in detail in Section 3.2.3, it will be achieved by using the OS alarm manager, which is an internal service that can call methods in the application if the application registers them.

4.6.4 Geocoder

We use Geocoding and reverse geocoding to translate from the longitude/latitude coordinates of the place to the name of the place. While advising someone to go to the University of Ottawa is understandable, telling them to go to 45.4244041, -75.6785358 is meaningless unless he uses a device that will show where those coordinates are. On the other hand, for a computer to grasp the location of something it needs the coordinates. That is why we need to translate where the user wants to go to the application and translate the places the application is suggesting to the user, in a language the user can understand.

4.7 System Interfaces

In this section, we will cover each of the system interfaces. In total, there are six interactive interfaces and a loading screen. The graphical user interfaces of the system are summarized in Figure 15.

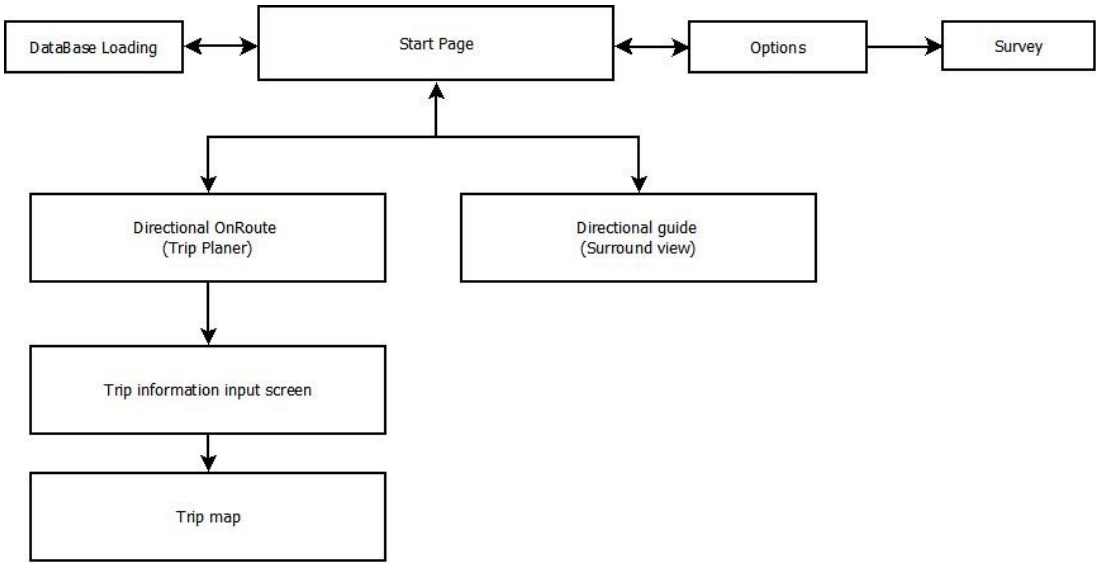

Figure 15. System interface architecture

4.7.1 Start Page

The start page of the system supports three actions, as shown in Figure 16. Upon the first startup of the application, the user will have to grant the application a number of permissions such as Internet access and camera roll access among many others.

If the “directional guide” is pressed, the application will load the POIs database and the subroutine for the directional guide will be executed. This subroutine will be explained in 4.7.3. The “Directional OnRoute” button will execute its associated subroutine as described in section 4.7.4. The last button is for the options and configurations as described in 4.7.1

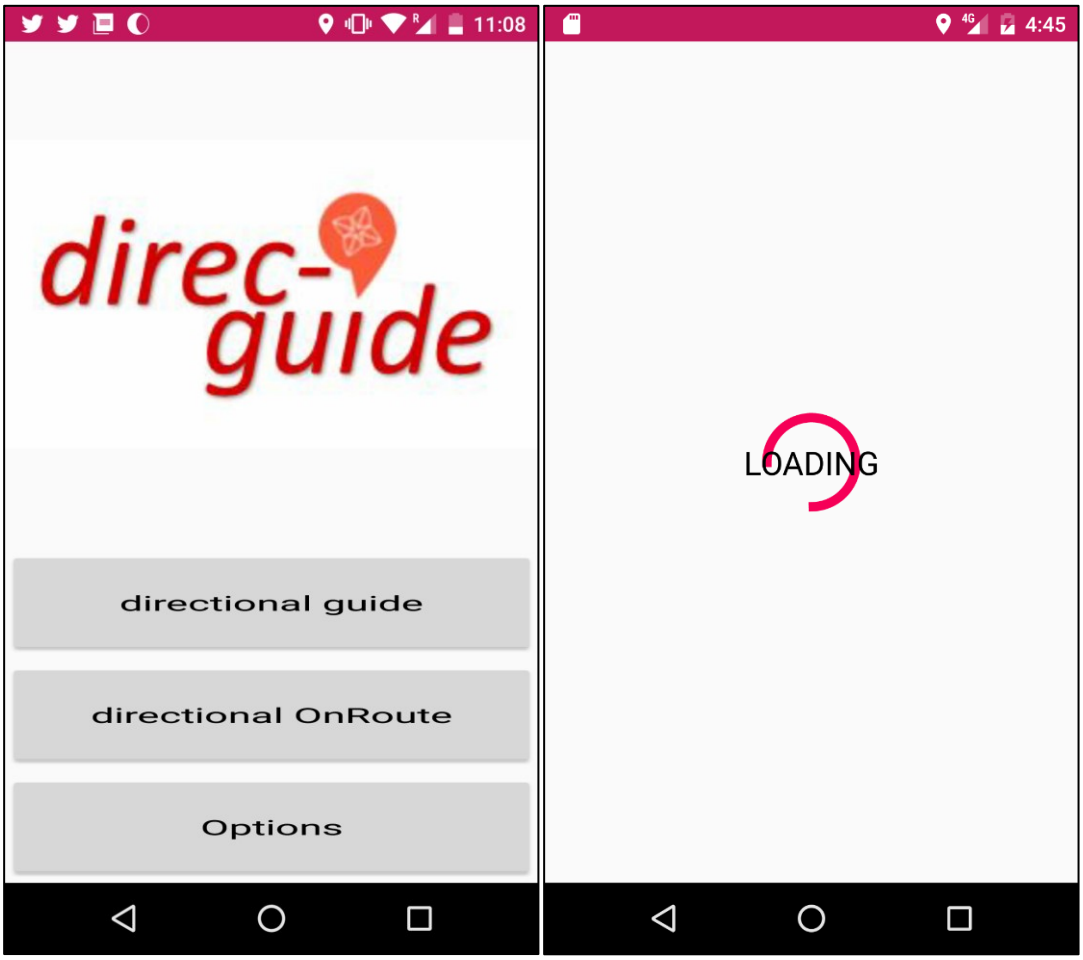

Figure 16. Main menu and loading screen

4.7.2 Options and Survey

The first thing the user needs to do is press the Options button to go to the options menu. In the menu, the user can specify the radius of the search. He can also exclude some types of POIs from the search results. Within the options, the user can access the survey, where the system will ask some questions related to the user's context in order to increase context awareness. Figure 17 shows both screens.

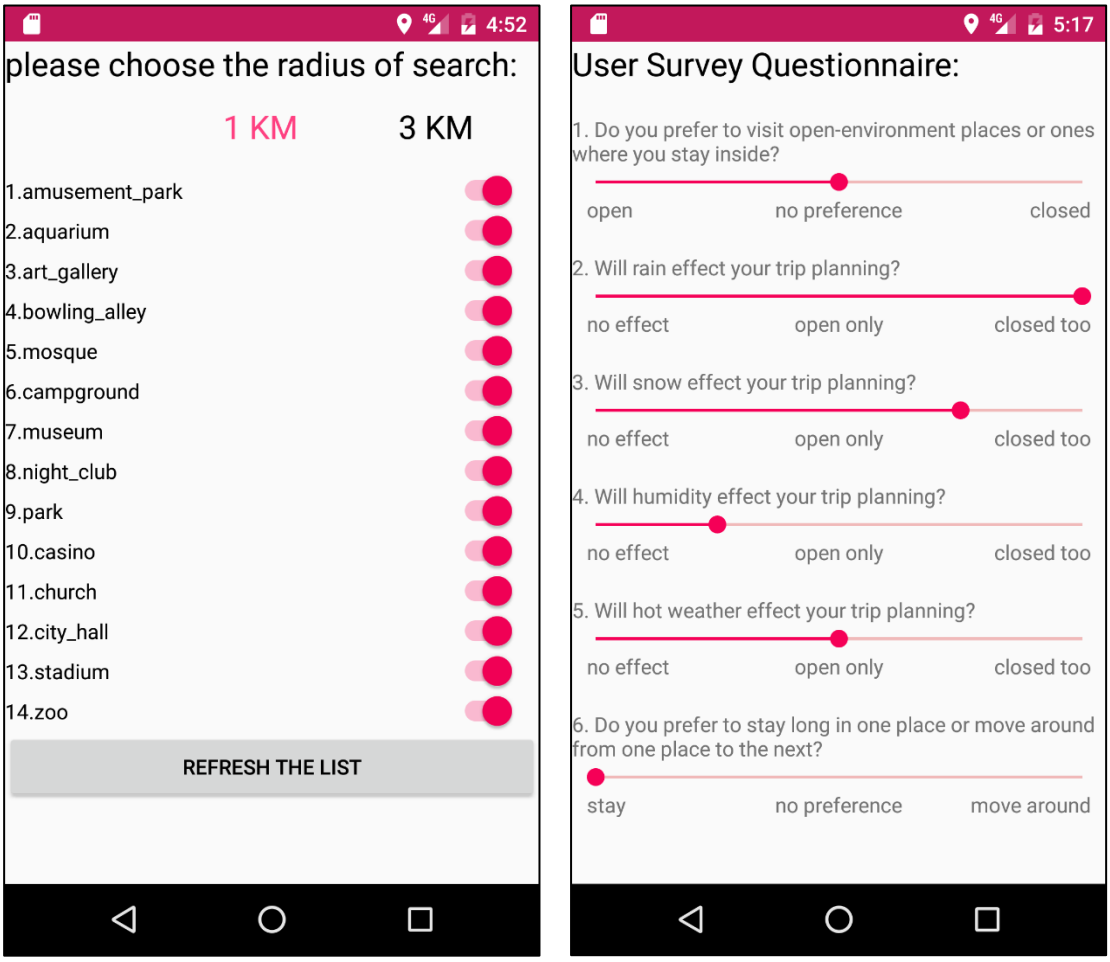

Figure 17 a) Options screen, b) user survey

4.7.3 Directional Guide (Surround View)

On this interface, the application will interact with the user in an augmented reality mode. For instance, if the user moves around, the system will change the display accordingly. If the user, for example, holds the device while facing north, the system will show suggestions for places that are both north of the user and nearby, as shown in Figure 18-a. If the user changes his location and now faces south, the system will show suggestions that are south, as well as nearby, as seen in Figure 18-b. This change is done in real time, and any movement the user does affects the results shown in the display.

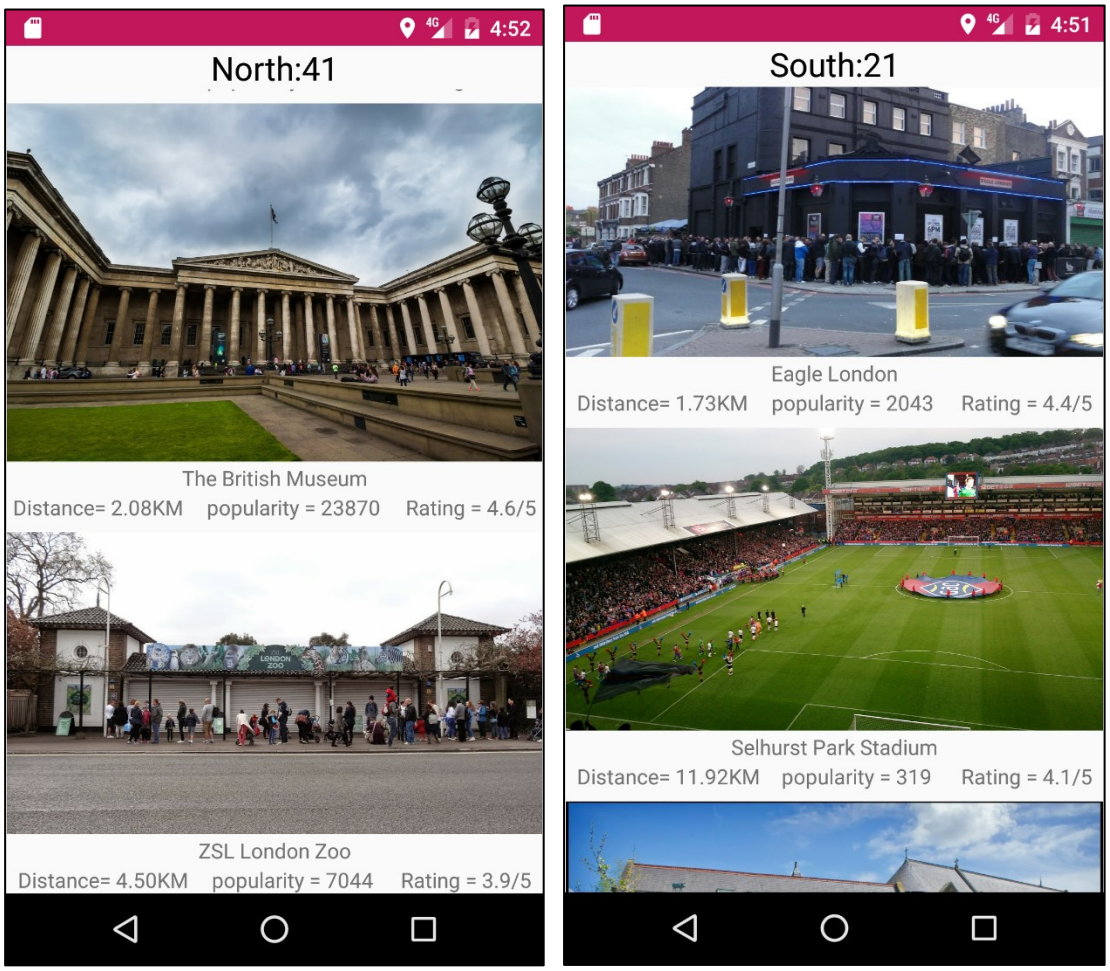

Figure 18 a) North, b) south views based on London's big ben.

4.7.4 Directional OnRoute (Trip Planner)

This interface is where the user will start trip planning. As shown in Figure 19-a, the user will start by entering his destination address (in this case the user's home address) and will then chose the trip destination and end. The user will then see the map view (Figure 19-b) in which he will see a blue pointer representing his current location. A black pointer shows his destination and a pink pointer indicate possible place to visit. If the user clicks on any of them, the system will display information about the place as well as a number

representing the order in which the user should visit it. After choosing one of the pointers, the user can then click on the Directions button on the bottom right of the screen to start the navigation to that point.

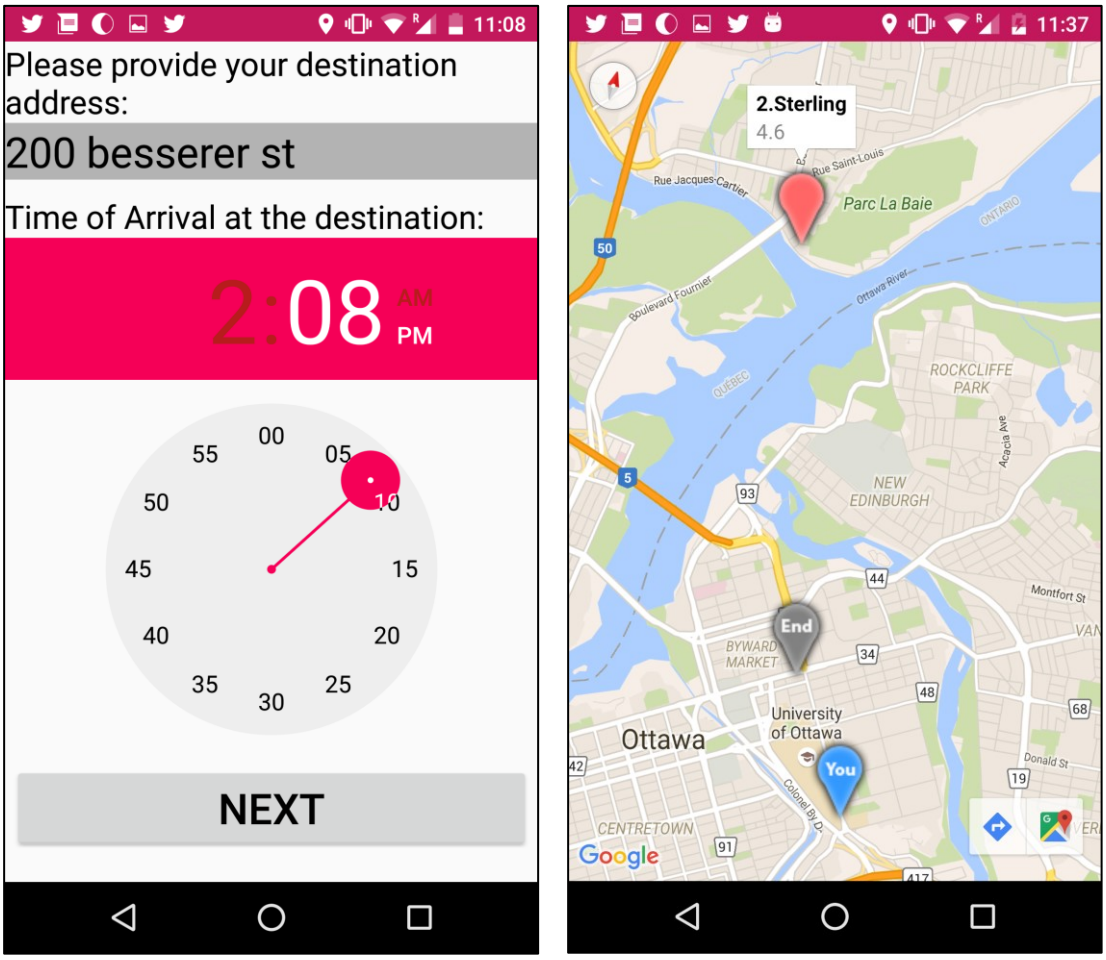

Figure 19 a) Information input b) Map view

Chapter 5

5 System Evaluation and Experimental Results

In this chapter, we evaluate the prediction performance of the BOP algorithm compared to the ground truth of the real-life data using well-known evaluation metrics. Moreover, the evaluation process includes an evaluation against the baseline algorithm in the field of trip planning, which is the OP algorithm. We will also show how accurately each of the algorithms can predict the ground truth using the YFCC100M database [46].

The chapter starts with the evaluation hypothesis in Section 5.1. The experimental setup follows in Section 5.2. Next, in Section 5.3, the experimental results of the prediction performance will be presented. Finally, in Section 5.4, the evaluation of the requirements will be presented.

5.1 Evaluation Hypothesis

If we can accurately predict the trips a user has taken in a city X by obtaining accurate travel history contextual data from other cities, we can say that our algorithm BOP can indeed plan trips in other cities as accurately as the city X plans.

5.2 Experimental Setup

The setting of the evaluation of BOP is done by taking random real-life trips and letting our system predict those trips by cross-validation [37].

We are going to use two metrics in cross-validation evaluation, which are recall and precision. We are also going to use their harmonic mean, F-score.

Here are the equations for the three:

$$Precision = \frac{True\ positive}{True\ positive + False\ positive}$$

$$Recall = \frac{True\ positive}{True\ positive + False\ negative}$$

$$F1 - Score = \frac{Precision * Recall}{Precision + Recall}$$

To limit the effect of the recommendation error we will only consider POIs that were included in the real trips. To get those trips, we use YFCC100m Dataset [46] by looking for high accuracy geo-tagged images, then searching images by the same user on the same day to recreate the trip done that day.

We manually extracted 40 trips that go through 17 POIs in Ottawa, Canada. We were very critical when considering trips in order to eliminate unwanted errors. In this case, the profit equals the number of visits to that POI (equation 1 in section 3.2.1) as well as how the user behaved in other cities (equation 2 in section 3.2.1). Duration is calculated using Equation 3 in section 3.2.1. Table 8 shows the total recall and precision of our algorithm compared to the original OP. Furthermore, it shows the impact of clustering on the results.

Table 8 Comparison between OP, balance OP, and BOP with clusters in terms of recall, precision and F1-score

Algorithm	Recall	Precision	F1-Score	AET
OP	0.718	0.6230	0.667	19.6 min
BOP	0.694	0.849	0.764	13.4 min
BOP+Cs	0.690	0.825	0.751	2.4 min

We can see the effect of the complexity reduction our clustering causes on the average execution time (AET) of each algorithm on a LG Nexus 5 mobile phone recorded

by the system clock. While we experience a 1 percent loss in the F1-score, we gain almost 5 times reduction in execution time. This is due to the algorithm's complexity relying on the number of clusters instead of the number of POIs, the former having a fraction of the latter's number.

5.3 Evaluation of the Experimental Results

As seen in Table 8, the OP obtains better recall results than our system because the time budget limit is traveling times and not time spent in POIs. While it is safe to assume the loss of precision may come from different factors, our algorithm yields better results than the original, which is sufficient to justify the usage. When we use our algorithm with clustering, we mildly sacrifice the precision, but by doing so the performance and complexity of the system are heavily impacted.

Chapter 6

6 Conclusion and Future Work

Planning trips is a difficult task still not because there is a lack of information but rather due to the vast amount of information that can be found. Thus, evaluating the choices and finding the best way to enjoy the trip will save the tourist a lot of time if done properly.

In this thesis, we presented our algorithm balanced orienteering problem to solve the planning aspect of tour planning. We also implemented a recommender system to suggest places visit and to rank those places through the mining of the Flickr database. Additionally, we built our mobile application with a scheduler to allow the system to adapt to the changes that the user may go through. In addition, while our system did not get a high prediction score, it provided better results than the standard algorithm.

We evaluated the performance of our proposed algorithm against the baseline algorithm using three accuracy measurements namely; precision, recall, and F-measure.

We also tested our system behavior in different parts of the world through emulation and mock locations, due to traveling to those places will be resources consuming process.

One of the limitations of our work that it does consume a lot of unneeded resources. An example of that is downloading the city POIs list multiple times each time the user re-enter the city. This process generates a lot of unwanted internet traffic.

In the future, we would like to implement a more advanced RS that could produce better suggestions. This will help to improve the precision of the planner and make it more

robust against biased recommendations. This improvement can be done by improving our user preferences by introducing personality treats or social influences. Or increasing the RS criteria to include the relationship between different POIs types.

Secondly, as a future improvement, we could make our algorithm run on multi threads to increase the response time further. Since most of mobile devices nowadays have mutable cores. We believe this is a very critical point to the end user as it will impact how the program will response. However, we did not consider it here as our aim was to showcase the algorithm without introducing the overhead of multithreading.

Lastly, by creating a centralized server we could remove the need to have the offline sub-system in the mobile application. While keeping the requests to the APIs, especially the non-free ones, low. Also, we could upload users' behaviors to the server and analyze it further to improve our system.

References

[1] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, “A survey on algorithmic approaches for solving tourist trip design problems,” *J. Heuristics*, vol. 20, no. 3, pp. 291–328, Jun. 2014.

[2] T. Tsiligirides, “Heuristic Methods Applied to Orienteering,” *J. Oper. Res. Soc.*, vol. 35, no. 9, pp. 797–809, 1984.

[3] M. Mowforth and I. Munt, *Tourism and Sustainability: Development, Globalisation and New Tourism in the Third World*. Routledge, 2015.

[4] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, “Mobile recommender systems in tourism,” *J. Netw. Comput. Appl.*, vol. 39, pp. 319–333, Mar. 2014.

[5] S. Jiang, X. Qian, J. Shen, Y. Fu, and T. Mei, “Author Topic Model based Collaborative Filtering for Personalized POI Recommendation,” *IEEE Trans. Multimed.*, pp. 1–1, 2015.

[6] J.-D. Zhang and C.-Y. Chow, “TICRec: A Probabilistic Framework to Utilize Temporal Influence Correlations for Time-aware Location Recommendations,” *IEEE Trans. Serv. Comput.*, pp. 1–1, 2015.

[7] Y.-C. Chen, W.-Y. Zhu, W.-C. Peng, W.-C. Lee, and S.-Y. Lee, “CIM: Community-Based Influence Maximization in Social Networks,” *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 2, pp. 1–31, Apr. 2014.

[8] G. Ference, M. Ye, and W.-C. Lee, “Location recommendation for out-of-town users in location-based social networks,” 2013, pp. 721–726.

[9] Y. Koren, “Collaborative filtering with temporal dynamics,” *Commun. ACM*, vol. 53, no. 4, pp. 89–97, 2010.

[10] J. J.-C. Ying, W.-C. Lee, and V. S. Tseng, “Mining geographic-temporal-semantic patterns in trajectories for location prediction,” *ACM Trans. Intell. Syst. Technol.*, vol. 5, no. 1, pp. 1–33, Dec. 2013.

[11] H. Gao, J. Tang, X. Hu, and H. Liu, “Modeling temporal effects of human mobile behavior on location-based social networks,” 2013, pp. 1673–1678.

[12] H. Gao, J. Tang, X. Hu, and H. Liu, “Exploring temporal effects for location recommendation on location-based social networks,” 2013, pp. 93–100.

[13] Y. Gao, J. Tang, R. Hong, Q. Dai, T.-S. Chua, and R. Jain, “W2Go: a travel guidance system by automatic landmark ranking,” in *Proceedings of the international conference on Multimedia*, 2010, pp. 123–132.

[14] J. Han and H. Lee, “Adaptive landmark recommendations for travel planning: Personalizing and clustering landmarks using geo-tagged social media,” *Pervasive Mob. Comput.*, vol. 18, pp. 4–17, Apr. 2015.

[15] B. McKercher, N. Shoval, E. Park, and A. Kahani, “The [limited] impact of weather on tourist behavior in an urban destination,” *J. Travel Res.*, vol. 54, no. 4, pp. 442–455, 2015.

[16] A. Doan, R. Ramakrishnan, and A. Y. Halevy, “Crowdsourcing Systems on the World-Wide Web,” *Commun ACM*, vol. 54, no. 4, pp. 86–96, Apr. 2011.

[17] M. Braunhofer, M. Elahi, and F. Ricci, “User Personality and the New User Problem in a Context-Aware Point of Interest Recommender System,” in *Information and Communication Technologies in Tourism 2015*, I. Tussyadiah and A. Inversini, Eds. Cham: Springer International Publishing, 2015, pp. 537–549.

[18] P. Vansteenwegen and D. Van Oudheusden, “The Mobile Tourist Guide: An OR Opportunity,” *Insight*, vol. 20, no. 3, pp. 21–27, Jul. 2007.

[19] C. P. Keller and M. F. Goodchild, “The Multiobjective Vending Problem: A Generalization of the Travelling Salesman Problem,” *Environ. Plan. B Plan. Des.*, vol. 15, no. 4, pp. 447–460, Dec. 1988.

[20] M. Dell’Amico, F. Maffioli, and P. Värbrand, “On Prize-collecting Tours and the Asymmetric Travelling Salesman Problem,” *Int. Trans. Oper. Res.*, vol. 2, no. 3, pp. 297–308, Jul. 1995.

[21] E. Balas, “The prize collecting traveling salesman problem,” *Networks*, vol. 19, no. 6, pp. 621–636, Oct. 1989.

[22] T. Tsiligirides, “Heuristic Methods Applied to Orienteering,” *J. Oper. Res. Soc.*, vol. 35, no. 9, pp. 797–809, Sep. 1984.

[23] G. Laporte and S. Martello, “The selective travelling salesman problem,” *Discrete Appl. Math.*, vol. 26, no. 2, pp. 193–207, Mar. 1990.

[24] S. Kataoka and S. Morito, “An Algorithm for Single Constraint Maximum Collection Problem,” *J. Oper. Res. Soc. Jpn.*, vol. 31, no. 4, Jan. 1988.

[25] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, “The orienteering problem: a survey,” *Eur. J. Oper. Res.*, vol. 209, no. 1, pp. 1–10, Feb. 2011.

[26] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, “Approximation algorithms for orienteering and discounted-reward TSP,” in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings*, 2003, pp. 46–55.

[27] N. Bansal, A. Blum, S. Chawla, and A. Meyerson, “Approximation algorithms for deadline-TSP and vehicle routing with time-windows,” in *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, 2004, pp. 166–174.

[28] C. Chekuri, N. Korula, and M. Pál, “Improved algorithms for orienteering and related problems,” *ACM Trans. Algorithms TALG*, vol. 8, no. 3, p. 23, 2012.

[29] C. Chekuri and M. Pal, “A recursive greedy algorithm for walks in directed graphs,” in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS’05)*, 2005, pp. 245–253.

[30] V. Nagarajan and R. Ravi, “The Directed Orienteering Problem,” *Algorithmica*, vol. 60, no. 4, pp. 1017–1030, Mar. 2011.

[31] F. V. Fomin and A. Lingas, “Approximation algorithms for time-dependent orienteering,” *Inf. Process. Lett.*, vol. 83, no. 2, pp. 57–62, Jul. 2002.

[32] Q. Wang, X. Sun, B. L. Golden, and J. Jia, “Using artificial neural networks to solve the orienteering problem,” *Ann. Oper. Res.*, vol. 61, no. 1, pp. 111–120, Dec. 1995.

[33] C. Archetti, A. Hertz, and M. G. Speranza, “Metaheuristics for the team orienteering problem,” *J. Heuristics*, vol. 13, no. 1, pp. 49–76, Dec. 2006.

[34] L. Tang and X. Wang, “Iterated local search algorithm based on very large-scale neighborhood for prize-collecting vehicle routing problem,” *Int. J. Adv. Manuf. Technol.*, vol. 29, no. 11–12, pp. 1246–1258, Oct. 2005.

[35] J. C. Kwan Hui Lim, “Personalized Tour Recommendation based on User Interests and Points of Interest Visit Durations,” 2015.

[36] Z. Xu, L. Chen, and G. Chen, “Topic based context-aware travel recommendation method exploiting geotagged photos,” *Neurocomputing*, vol. 155, pp. 99–107, May 2015.

[37] R. Kohavi, “A Study of Cross-validation and Bootstrap for Accuracy Estimation and Model Selection,” in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, San Francisco, CA, USA, 1995, pp. 1137–1143.

[38] D. Gavalas, V. Kasapakis, C. Konstantopoulos, G. Pantziou, N. Vathis, and C. Zaroliagis, “The eCOMPASS multimodal tourist tour planner,” *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7303–7316, Nov. 2015.

[39] A. Likas, N. Vlassis, and J. J. Verbeek, “The global k-means clustering algorithm,” *Pattern Recognit.*, vol. 36, no. 2, pp. 451–461, Feb. 2003.

[40] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, "The City Trip Planner: An expert system for tourists," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 6540–6546, Jun. 2011.

[41] T. A. Feo and M. G. C. Resende, "Greedy Randomized Adaptive Search Procedures," *J. Glob. Optim.*, vol. 6, no. 2, pp. 109–133, Mar. 1995.

[42] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera, "Personalized Tour Recommendation based on User Interests and Points of Interest Visit Durations," in *24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, 2015.

[43] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram, "Effects of obesity and sex on the energetic cost and preferred speed of walking," *J. Appl. Physiol.*, vol. 100, no. 2, pp. 390–398, Feb. 2006.

[44] T. A. Feo and M. G. C. Resende, "A probabilistic heuristic for a computationally difficult set covering problem," *Oper. Res. Lett.*, vol. 8, no. 2, pp. 67–71, Apr. 1989.

[45] M. Ibrahim and M. Youssef, "CellSense: A Probabilistic RSSI-Based GSM Positioning System," in *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*, 2010, pp. 1–5.

[46] B. Thomee *et al.*, "YFCC100M: The New Data in Multimedia Research," *Commun. ACM*, vol. 59, no. 2, pp. 64–73, Jan. 2016.

Appendix A

The GK-means algorithm calculation of K steps (each step increases the clusters by 1). In this Appendix we jumped steps 6-69 as city of Ottawa reach MAD of <1 at 79.

city of ottawa clustring with max clusters of 112		new york city clustring with max clusters of 140		city of london clustring with max clusters of 140	
meanSqErrors in km	halt at 1km avrage error	meanSqErrors in km	halt at 1km avrage error	meanSqErrors in km	halt at 1km avrage error
571.6307576118991 | 1	571.6307576118991 | 1	660.129240960317 | 1	660.129240960317 | 1	708.226934083057 | 1	708.226934083057 | 1
356.61475313501103 | 2	356.61475313501103 | 2	393.7224081894844 | 2	393.7224081894844 | 2	458.5608806300225 | 2	458.5608806300225 | 2
205.2192863996313 | 3	205.2192863996313 | 3	220.52473385201762 | 3	220.52473385201762 | 3	290.72945101595604 | 3	290.72945101595604 | 3
168.2838230315637 | 4	168.2838230315637 | 4	174.2671265611034 | 4	174.2671265611034 | 4	223.2313898762384 | 4	223.2313898762384 | 4
138.28053731678853 | 5	138.28053731678853 | 5	134.21742597213682 | 5	134.21742597213682 | 5	179.1363064029315 | 5	179.1363064029315 | 5
1.4662749182863375 | 70	1.4662749182863375 | 70	3.098818468603126 | 70	3.098818468603126 | 70	3.6416822363580037 | 70	3.6416822363580037 | 70
1.4122154909512445 | 71	1.4122154909512445 | 71	2.9594937340384813 | 71	2.9594937340384813 | 71	3.531375710347088 | 71	3.531375710347088 | 71
1.3482614474679022 | 72	1.3482614474679022 | 72	2.8727711391332327 | 72	2.8727711391332327 | 72	3.4053418677987777 | 72	3.4053418677987777 | 72
1.28638157826795 | 73	1.28638157826795 | 73	2.760476029954345 | 73	2.760476029954345 | 73	3.2962814328508894 | 73	3.2962814328508894 | 73
1.2289958984356193 | 74	1.2289958984356193 | 74	2.6741107739830143 | 74	2.6741107739830143 | 74	3.1999030611636075 | 74	3.1999030611636075 | 74
1.177636568467697 | 75	1.177636568467697 | 75	2.5842795905620375 | 75	2.5842795905620375 | 75	3.06097515665275 | 75	3.06097515665275 | 75
1.122038946554055 | 76	1.122038946554055 | 76	2.4395306090789983 | 76	2.4395306090789983 | 76	2.9487396265623618 | 76	2.9487396265623618 | 76
1.0669754756017147 | 77	1.0669754756017147 | 77	2.347800695515661 | 77	2.347800695515661 | 77	2.813084571449764 | 77	2.813084571449764 | 77
1.0210279956207589 | 78	1.0210279956207589 | 78	2.2659138932789866 | 78	2.2659138932789866 | 78	2.7109161778130355 | 78	2.7109161778130355 | 78
0.9737285426825356 | 79	0.9737285426825356 | 79	2.1821025907549743 | 79	2.1821025907549743 | 79	2.6151817674113302 | 79	2.6151817674113302 | 79
0.9257303980558037 | 80		2.0962141981008267 | 80	2.0962141981008267 | 80	2.5262999228477523 | 80	2.5262999228477523 | 80
0.8841175804743304 | 81		2.020938682391569 | 81	2.020938682391569 | 81	2.435793551431935 | 81	2.435793551431935 | 81
0.8459632942339103 | 82		1.945063886770087 | 82	1.945063886770087 | 82	2.3398982602414193 | 82	2.3398982602414193 | 82
0.80086342875128 | 83		1.87840578815529 | 83	1.87840578815529 | 83	2.2694752550056467 | 83	2.2694752550056467 | 83
0.7586321293739413 | 84		1.8045807891333092 | 84	1.8045807891333092 | 84	2.1875518431098526 | 84	2.1875518431098526 | 84
0.7195944592750662 | 85		1.7470180102876993 | 85	1.7470180102876993 | 85	2.110751840855046 | 85	2.110751840855046 | 85
0.684712614648636 | 86		1.6779401786350892 | 86	1.6779401786350892 | 86	2.039710455401032 | 86	2.039710455401032 | 86
0.6500658099500359 | 87		1.6214042104576503 | 87	1.6214042104576503 | 87	1.9709532143367512 | 87	1.9709532143367512 | 87
0.6195028573672733 | 88		1.5697417600914114 | 88	1.5697417600914114 | 88	1.900375054638278 | 88	1.900375054638278 | 88
0.5928584244372953 | 89		1.5333249895152352 | 89	1.5333249895152352 | 89	1.8390695077955241 | 89	1.8390695077955241 | 89
0.5659148399803002 | 90		1.4804706061611455 | 90	1.4804706061611455 | 90	1.7676625077963437 | 90	1.7676625077963437 | 90
0.5441281280242485 | 91		1.4342298061905838 | 91	1.4342298061905838 | 91	1.7076758006768598 | 91	1.7076758006768598 | 91
0.5233872939906067 | 92		1.3919468850203875 | 92	1.3919468850203875 | 92	1.651494043986978 | 92	1.651494043986978 | 92
0.49895184166073137 | 93		1.3477327079307402 | 93	1.3477327079307402 | 93	1.5971094210685215 | 93	1.5971094210685215 | 93
0.4773628649056797 | 94		1.3035552119387341 | 94	1.3035552119387341 | 94	1.5376804784425175 | 94	1.5376804784425175 | 94
0.4592850926719291 | 95		1.25696631727392 | 95	1.25696631727392 | 95	1.4757758656346451 | 95	1.4757758656346451 | 95
0.44034150311035425 | 96		1.217314244340056 | 96	1.217314244340056 | 96	1.4212624626503265 | 96	1.4212624626503265 | 96
0.42236625282950224 | 97		1.1802935892922017 | 97	1.1802935892922017 | 97	1.3703413537788025 | 97	1.3703413537788025 | 97
0.40111868085972846 | 98		1.1437982757327843 | 98	1.1437982757327843 | 98	1.322166033124277 | 98	1.322166033124277 | 98
0.3801502801999515 | 99		1.1061177603219494 | 99	1.1061177603219494 | 99	1.273804652254407 | 99	1.273804652254407 | 99
0.3615648194295581 | 100		1.0697360035555674 | 100	1.0697360035555674 | 100	1.2256860775104066 | 100	1.2256860775104066 | 100
0.34615003193379384 | 101		1.0308179812359783 | 101	1.0308179812359783 | 101	1.1828883897170726 | 101	1.1828883897170726 | 101
0.33172359700336307 | 102		0.9991099763579302 | 102	0.9991099763579302 | 102	1.1369398240632784 | 102	1.1369398240632784 | 102
0.3170234350741134 | 103		0.9677356157929442 | 103		1.0927483461159793 | 103	1.0927483461159793 | 103
0.30215381296015265 | 104		0.938527500078761 | 104		1.0487732583041816 | 104	1.0487732583041816 | 104
0.29033660293106606 | 105		0.9096013922774631 | 105		0.9979382485598856 | 105	0.9979382485598856 | 105
0.2760305239058753 | 106		0.8779018707812506 | 106		0.9554241136308866 | 106	
0.2617719345791679 | 107		0.8492961261210239 | 107		0.9238270059530992 | 107	
0.25032789790993515 | 108		0.821944685217288 | 108		0.8879232171175584 | 108	
0.23866259772675585 | 109		0.7945262638291992 | 109		0.8532926188914648 | 109	
0.22835882179599154 | 110		0.7684532523401622 | 110		0.8191666678134352 | 110	
0.2178874557402429 | 111		0.7423231180013072 | 111		0.7905412330472236 | 111	
0.2081821538722333 | 112		0.7187841853239288 | 112		0.7525960253254363 | 112	
		0.6948901332902918 | 113		0.7212488767673451 | 113	
		0.6719605240623377 | 114		0.6969867560997253 | 114	
		0.6495093781328235 | 115		0.6736233647791866 | 115	
		0.6276317866241401 | 116		0.651048015409307 | 116	
		0.6072132440493933 | 117		0.6246434268643627 | 117	
		0.5855017402262112 | 118		0.6036432081566364 | 118	
		0.5630614269963689 | 119		0.5818545670972791 | 119	
		0.5448510860369746 | 120		0.5603036348353078 | 120	
		0.5230223339375468 | 121		0.5376249396950965 | 121	
		0.5041281816890405 | 122		0.5140195637789641 | 122	
		0.4865869606227989 | 123		0.49236811528904756 | 123	
		0.46927537802716685 | 124		0.47278707216507976 | 124	
		0.4531346230212376 | 125		0.4516549908623071 | 125	
		0.4366494223120063 | 126		0.4331457288571056 | 126	
		0.420723122997763 | 127		0.41595021116057546 | 127	
		0.4066616415740705 | 128		0.39925677103087975 | 128	
		0.3903647327535948 | 129		0.3849594346654607 | 129	
		0.37657023994931405 | 130		0.37404146906266084 | 130	
		0.3622709451904303 | 131		0.3590998217830802 | 131	
		0.347637998586572 | 132		0.3446386680834473 | 132	
		0.3354410035312246 4 | 133		0.33040404095279374 | 133	
		0.3240262945560171 | 134		0.3175876309511282 | 134	
		0.31258779985966106 | 135		0.3059102870973088 | 135	
		0.30247595170882163 | 136		0.2942861884184322 | 136	

Appendix B – APIs responses

Yahoo weather response for city of Ottawa:


```
<query xmlns:yahoo="http://www.yahooapis.com/v1/base.rng" yahoo:count="2" yahoo:created="
2016-09-14T20:19:02Z" yahoo:lang="en-US">
<results>
<place xmlns="http://where.yahooapis.com/v1/schema.rng" xmlns:yahoo="http://www.yahooapis
.com/v1/base.rng" xml:lang="en-
US"yahoo:uri="http://where.yahooapis.com/v1/place/91982014">
<woeid>91982014</woeid>
<placeTypeName code="7">Town</placeTypeName>
<name>Ottawa</name>
<country code="CA" type="Country" woeid="23424775">Canada</country>
<admin1 code="CA-ON" type="Province" woeid="2344922">Ontario</admin1>
<admin2 code="" type="County/District" woeid="29375164">Ottawa</admin2>
<admin3/>
<locality1 type="Town" woeid="91982014">Ottawa</locality1>
<locality2/>
<postal/>
<centroid>
<latitude>45.37138</latitude>
<longitude>-75.685341</longitude>
</centroid>
<boundingBox>
<southWest>
<latitude>45.262161</latitude>
<longitude>-75.857178</longitude>
</southWest>
<northEast>
<latitude>45.465359</latitude>
<longitude>-75.578568</longitude>
</northEast>
</boundingBox>
<areaRank>1</areaRank>
<popRank>1</popRank>
<timezone type="Time Zone" woeid="56043697">America/Toronto</timezone>
</place>
<place xmlns="http://where.yahooapis.com/v1/schema.rng" xmlns:yahoo="http://www.yahooapis
.com/v1/base.rng" xml:lang="en-
US"yahoo:uri="http://where.yahooapis.com/v1/place/29375164">
<woeid>29375164</woeid>
<placeTypeName code="9">County/District</placeTypeName>
<name>Ottawa</name>
<country code="CA" type="Country" woeid="23424775">Canada</country>
<admin1 code="CA-ON" type="Province" woeid="2344922">Ontario</admin1>
<admin2 code="" type="County/District" woeid="29375164">Ottawa</admin2>
<admin3/>
<locality1/>
<locality2/>
<postal/>
<centroid>
<latitude>45.249298</latitude>
<longitude>-75.800903</longitude>
</centroid>
<boundingBox>
<southWest>
<latitude>44.962002</latitude>
<longitude>-76.355766</longitude>
</southWest>
<northEast>
<latitude>45.536541</latitude>
<longitude>-75.246033</longitude>
</northEast>
</boundingBox>
<areaRank>1</areaRank>
<popRank>1</popRank>
<timezone type="Time Zone" woeid="56043697">America/Toronto</timezone>
```

```
</place>
</results>
</query>
<!--  total: 10  -->
<!--
 prod_bf1_1;paas.yql;queryyahooapiscomproductionbf1;d41a6d01-75e9-11e6-a2c9-d4ae52974741
```


Google places response for parliament hill:


```
{
   "html_attributions" : [],
   "result" : {
      "address_components" : [
         {
            "long_name" : "Wellington Street",
            "short_name" : "Wellington St",
            "types" : [ "route" ]
         },
         {
            "long_name" : "Byward Market - Parliament Hill",
            "short_name" : "Byward Market - Parliament Hill",
            "types" : [ "neighborhood", "political" ]
         },
         {
            "long_name" : "Ottawa",
            "short_name" : "Ottawa",
            "types" : [ "locality", "political" ]
         },
         {
            "long_name" : "Ottawa Division",
            "short_name" : "Ottawa Division",
            "types" : [ "administrative_area_level_2", "political" ]
         },
         {
            "long_name" : "Ontario",
            "short_name" : "ON",
            "types" : [ "administrative_area_level_1", "political" ]
         },
         {
            "long_name" : "Canada",
            "short_name" : "CA",
            "types" : [ "country", "political" ]
         },
         {
            "long_name" : "K1A",
            "short_name" : "K1A",
            "types" : [ "postal_code_prefix", "postal_code" ]
         }
      ],
      "adr_address"      :      "\u003cspan      class=\"street-address\"\u003eWellington
St\u003c/span\u003e,      \u003cspan      class=\"locality\"\u003eOttawa\u003c/span\u003e,
\u003cspan     class=\"region\"\u003eON\u003c/span\u003e     \u003cspan     class=\"postal-
code\"\u003eK1A\u003c/span\u003e,               \u003cspan               class=\"country-
name\"\u003eCanada\u003c/span\u003e",
      "formatted_address" : "Wellington St, Ottawa, ON K1A, Canada",
      "formatted_phone_number" : "(613) 992-4793",
      "geometry" : {
         "location" : {
            "lat" : 45.4235937,
            "lng" : -75.700929
         },
         "viewport" : {
            "northeast" : {
               "lat" : 45.42473465,
               "lng" : -75.69583995000001
            },
            "southwest" : {
               "lat" : 45.42075245,
```

```
            "lng" : -75.70730214999999
         }
      }
   },
   "icon"    :    "https://maps.gstatic.com/mapfiles/place_api/icons/generic_business-
71.png",
   "id" : "09a644f608228131a36108ff2f6358f52382ad56",
   "international_phone_number" : "+1 613-992-4793",
   "name" : "Parliament Hill",
   "opening_hours" : {
      "open_now" : true,
      "periods" : [
         {
            "close" : {
               "day" : 1,
               "time" : "1800"
            },
            "open" : {
               "day" : 1,
               "time" : "0830"
            }
         },
         {
            "close" : {
               "day" : 2,
               "time" : "1800"
            },
            "open" : {
               "day" : 2,
               "time" : "0830"
            }
         },
         {
            "close" : {
               "day" : 3,
               "time" : "1800"
            },
            "open" : {
               "day" : 3,
               "time" : "0830"
            }
         },
         {
            "close" : {
               "day" : 4,
               "time" : "1800"
            },
            "open" : {
               "day" : 4,
               "time" : "0830"
            }
         },
         {
            "close" : {
               "day" : 5,
               "time" : "1700"
            },
            "open" : {
               "day" : 5,
               "time" : "0830"
            }
         }
      ],
      "weekday_text" : [
         "Monday: 8:30 AM – 6:00 PM",
         "Tuesday: 8:30 AM – 6:00 PM",
         "Wednesday: 8:30 AM – 6:00 PM",
         "Thursday: 8:30 AM – 6:00 PM",
         "Friday: 8:30 AM – 5:00 PM",
         "Saturday: Closed",
         "Sunday: Closed"
```

```
         ]
      },
      "photos" : [
         {
            "height" : 1944,
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/113430137563292314858/photos\"\u003eYau
Sir\u003c/a\u003e"
            ],
            "photo_reference"                    :                    "CoQBdwAAANHEjoYCp-
yne7guX6JjVw4T1w0I3iwfb4s2pRrLmHzbzMUHGGFNxNxJR6RNhyaoEc_I-
f8fvCYdGVBgzisHwTSRVdderOttWJ9y2oPlxYqsqz10qVd5sz0f8SETo4AcRbAkJDo0GZ9zRAPYjCM28HQR3_3Jh7
vjBWoeXZQUFLvSEhDYbswzMOn6ViHCe4iPAVfUGhSlLwdYhmZXgKQUCey3tqrfEn7xmw",
            "width" : 2592
         },
         {
            "height" : 1530,
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/105608278213193244363/photos\"\u003ePierre
Roussin\u003c/a\u003e"
            ],
            "photo_reference"      :      "CoQBcwAAAJN1J8c3ersUPqCKDL3zKi8YvVCROj2r5_j2w-
Nxe7Top7TAPj6cMdIgrURxNSRkUATjusnoAk4Tl3gJKTRSeQ-
WC0jxmpTV9BaLld52h0yNUDGOdAVZMhOvkIYg7TiEz5hqyCxaY_X9X9lXJOFXkiZx4b08UjTjyJnpF1U4XtOqEhAo
gKup0zXHZMXx-r6_QdW9GhS8AbuqIbeB7NP4M-329JX1t_Ry7A",
            "width" : 2048
         },
         {
            "height" : 3137,
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/110680787361957156642/photos\"\u003eHai
Vu\u003c/a\u003e"
            ],
            "photo_reference"        :        "CoQBcwAAAH6u7ezov9frqeIGpWYYfp84c2E6KsZaNp-
GmYXCvdXNqnMjYK_lJGdx_DSuhOLwSFkw11qB40PcuPzymJ73TmyN8agyR-qwrULOaE74-Un8-
6Cj4EeV6s86K1Gh3gnWpSqdIxND1yG9nfDqfIrHH_lw5LG86Hun2EeG1Ra2M946EhCD56fCrHBxKOPkNV63qYuiGh
T5fmrWds4kqdw0i_cc7grM5v8O-w",
            "width" : 4706
         },
         {
            "height" : 1535,
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/110351858250777290588/photos\"\u003eSachin
Bakshi\u003c/a\u003e"
            ],
            "photo_reference"                  :                  "CoQBcwAAAJDAS8nR4c8_2u-
nXt51AK7asOvERKUz27dqA0z8uO2A7URHGhDafbyj-gZxYtEjvNKp0P_ZBq1SaiJALmUU10DujVfb0icFA4-
H5Rsds7-
3b0I8dXxNQTDWzO1aY3H_TjIItuJcfkxpshfOu9wh0TDwPkiXUxXq4jox5oMCT0bVEhDFkSxkeRFBF9BTvCWJ9hDQ
GhQJapU8Mz7SvNLq63Wy9Vpm4zS55Q",
            "width" : 2048
         },
         {
            "height" : 1600,
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/113882217801513770275/photos\"\u003eCarlos
Cortada\u003c/a\u003e"
            ],
            "photo_reference"                                                                :
"CoQBcwAAAFNmq0bQaYqUOOzAyuM6Lmf9m0T_0131CUXuLfZSdUUWVDtvrLT4xsSOz3hfL_GMsTJ2KOx5f8LNjFuc
gpQbC0TXXnfaJSNSqPNKaFDYaonDhz1gcyH4qDmeI44uDBS1m8j_RuwZtKL03M-xqI-
pHZeOSeEXAAaig7wQ69e2jh-gEhAYSVWc6w3eYqSv5_FHiXNmGhSwpmVH7BJHma9vHT8A_VVpY9dasQ",
            "width" : 1200
         },
         {
            "height" : 1530,
```

```
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/105608278213193244363/photos\"\u003ePierre
Roussin\u003c/a\u003e"
            ],
            "photo_reference"                        :                    "CoQBcwAAALzT-
AZSyYJeH3xkRqkqEDW9vvURQFj9fr6XHCcWvtrsrDtpx5VOLSqUnpqAmrdtEtrNWpVOfioOKrGER08z5QTPsZXvzy
LmGqPLZjxXaN2hk52IGxqb-
TV4BpIw44VBAFR1dmD3dkYIswadrcBbv68ZsbTgFGXQqfE1Vl3zewzeEhBvf3rP_K2XjdKNU3Gars4MGhSPeVjnvs
-VHZQPdDp3vpuvqrSlKg",
            "width" : 2048
         },
         {
            "height" : 1530,
            "html_attributions" : [
               "\u003ca
href=\"https://maps.google.com/maps/contrib/115536394175732749811/photos\"\u003eDominick
K\u003c/a\u003e"
            ],
            "photo_reference"                        :                    "CoQBcwAAAG0c_-
MVPIEynh5vlGlSHixTUX7IMYStg1AVtYlTTBCWJIpSchhZ0POHxb8igh8NmJPuth0lMMok1KV2OL-dr2Onf_bzF-
zEkbXOOL_5SuLKpU5zQQbpbcr57AZpg6QFZFthn5FD42AomzSPt5rzA5bKhUMkaJzmr4mEFx_7m-
4IEhArEyHZR5f7fiunJS2Cmo3LGhRsgvfn4RPYAsfHJVWg37GkUO3K8w",
            "width" : 2048
         }
      ],
      "place_id" : "ChIJ75TkT_8EzkwRbp_CYE-1awI",
      "rating" : 4.7,
      "reference"        :        "CnRiAAAAoWvKQ--xIExCgY9xztl8M-lxVYB9iYgayNTwPxAXatHCEX-
Gx9489n0WvHOGbBg9w0iPqWCbaH9Rx2AGKjzZlaI3H5iEJUBAlt1wvdx5GBEFFzDSkmrRr2UWVQWTUODtTgMxM3aZ
fUICDYKxD_yE3xIQfqvozpD7AgB-ZX84Kth5YxoUQ-LKXbE4rT0liByw-Mtoj-B5BKU",
      "reviews" : [
         {
            "aspects" : [
               {
                  "rating" : 3,
                  "type" : "overall"
               }
            ],
            "author_name" : "Jessica A",
            "author_url" : "https://plus.google.com/111165642780714221243",
            "language" : "en",
            "profile_photo_url"              :              "//lh5.googleusercontent.com/-
MPRm2zs05h4/AAAAAAAAAAI/AAAAAAAAAEE/_RxEsdEukuE/photo.jpg",
            "rating" : 5,
            "text" : "If you want to do the free tour, show up at 830 or 845am to line up
for one hour (weekday in August.) It was definitely worth it though. Very cool and important
part of Canada. ",
            "time" : 1471633402
         },
         {
            "aspects" : [
               {
                  "rating" : 3,
                  "type" : "overall"
               }
            ],
            "author_name" : "Mina Buysse",
            "author_url" : "https://plus.google.com/103100181342235219580",
            "language" : "en",
            "profile_photo_url"              :              "//lh3.googleusercontent.com/-
LeH3kz7KRbI/AAAAAAAAAAI/AAAAAAAABPM/PTYEi0T7lyU/photo.jpg",
            "rating" : 5,
            "text" : "The nation's capital! It's a great place to tour. The scenery is
beautiful (well, if it wasn't always under construction it'd be a bit nicer). The prices
are reasonable and the surrounding area is filled with tons of Ottawa's specialty foods.
Definitely a must visit if you're in Ottawa. ",
            "time" : 1472393687
         },
         {
            "aspects" : [
```

```
               {
                  "rating" : 3,
                  "type" : "overall"
               }
            ],
            "author_name" : "Don Bodnaruk",
            "author_url" : "https://plus.google.com/116079318502757079386",
            "language" : "en",
            "profile_photo_url"              :              "//lh6.googleusercontent.com/-
lULKLt9b1j0/AAAAAAAAAAI/AAAAAAAACDg/XwBvBbbdhak/photo.jpg",
            "rating" : 5,
            "text" : "We showed up at 8:20am to get tickets and were 100 deep in line.
People that were there at 8 were 27th in line.\nMy favorite stop in Ottawa so far. Tours of
East block and Center Block were great. If you remember your high school Canadian History,
nothing is really new information.  However,  seeing where things happen and the architecture
is something else. Light show is a great way to spend the evening. ",
            "time" : 1470311661
         },
         {
            "aspects" : [
               {
                  "rating" : 3,
                  "type" : "overall"
               }
            ],
            "author_name" : "Lang Wong",
            "author_url" : "https://plus.google.com/117619111043828559369",
            "language" : "en",
            "profile_photo_url"              :              "//lh3.googleusercontent.com/-
_TpGUdLGtik/AAAAAAAAAAI/AAAAAAAAACY/9MyBAVBKKSs/photo.jpg",
            "rating" : 5,
            "text" : "Ohhhhh Canada!\nFirst time visit to our nation's capital.\nWe could
not help but feel extra proud to be a Canadian.\nOttawa city in general is a beautiful city
but the piece de resistance is Parliament Hill.  One should not leave Ottawa without taking
a tour through the grounds and the buildings.  They offer a free tour of the Centre Block
that contains the House of Commons and Senate Chambers and of course the iconic Peace Tower
(tours can be busy).  They are all very majestic in their own way.  However, the most
impressive area for me was the Library of Parliament.  It is beautiful.  \nA must see for
all Canadians and those who visit Canada from abroad.",
            "time" : 1470162904
         },
         {
            "aspects" : [
               {
                  "rating" : 3,
                  "type" : "overall"
               }
            ],
            "author_name" : "Katerina Foret",
            "author_url" : "https://plus.google.com/117902335238895790363",
            "language" : "en",
            "rating" : 5,
            "text" : "The place is breathtaking both inside and outside. We especially
appreciated the fact that the entrance to the Parliament and its Peace Tower was completely
free. They have a good system in making reservations for guided tours - you need to pick up
the tickets in the opposite building to the Parliament. They only have a limited number of
tickets for each tour that departa as often as every 10 minutes. Even though you come in
the morning they still have tickets for afternoon tours. There are 2 different tickets -
for the Peace tower and for the Parliament. The one around Parliament tskes about 40 minutes.
",
            "time" : 1473118771
         }
      ],
      "scope" : "GOOGLE",
      "types" : [ "point_of_interest", "establishment" ],
      "url" : "https://maps.google.com/?cid=174432363114307438",
      "utc_offset" : -240,
      "vicinity" : "Wellington Street, Ottawa",
      "website" : "http://www.parl.gc.ca/"
   },
   "status" : "OK"
```

Google Radar Search response:


```
{
   "html_attributions" : [],
   "results" : [
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4235937,
               "lng" : -75.700929
            }
         },
         "id" : "09a644f608228131a36108ff2f6358f52382ad56",
         "place_id" : "ChIJ75TkT_8EzkwRbp_CYE-1awI",
         "reference"                                                                    :
"CnRiAAAAQoV8wphG_BOW5vR1xVB04ZuFYFjwDYPX6vdb9RGNPuDxmNT14ryvTI3XY5MlRWDuoeCM1rsXJeRgKEU4
_G4b1TqGLfX10oM2_asAmEAyeFAWf3C-
htued6iwrX7P8Df8gAjOZmkJ8EwJkzZ6Dwoi5RIQVOAS8Avqato7Gy0FG3MGjBoU5s5wSuSnyPMfywalmEgKHA5fj
a8"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.417736,
               "lng" : -75.70667349999999
            }
         },
         "id" : "b7df66b10ecbc50dfebe1d64ab5d76f5851501dd",
         "place_id" : "ChIJEa2zPVEEzkwRRRgvdwLBxb0",
         "reference"       :       "CoQBeQAAAIEa5cOgVfTPy9Luk0bA4QQp4Sc72KB2eIF5Zk0gSMdQ-
3MUlLe_icA9ZYCAqNm5oHagEfZbjUlTZAGYVTgP728NgylDxuZ_OQiTUGphIMsGqwqvmvOgAurElDmaYrofMpirFX
4n7NtyfmqsOQRskPYOcHsHkLdH-
oeJxc0uU_4VEhAj5M2mGUXhpPToLkVgjP_4GhSnsvrxO859tGlMUlFIeNo3U98eyQ"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.42144039999999,
               "lng" : -75.6942906
            }
         },
         "id" : "8845b6b91659609262a00539ed1f30a5c6475c42",
         "place_id" : "ChIJg4YwW1QEzkwRF440yQfrV9w",
         "reference"                                                                    :
"CnRoAAAAm3CzeUsTLujNIEDCohi2ALI2ZVqzGfCsexxJEFBLTkdZgBtxEyf_G9KvXXSdrv8IHgsoxX3URF5v92f4
-l7e6TvKSJSVdHYToaBdu0Gs7HCYQa96L-
v3LsTO_4BXVAckcTxYkoNbDPJBWRQHQMGMvRIQlgajCrpdCzb_YCo8Cv2ebRoUwjEXU9rzGxeMbLi5--sEn-
tNWt8"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.42281560000001,
               "lng" : -75.6978446
            }
         },
         "id" : "c9681f9f66598affc504e1fd0a88a500155632e5",
         "place_id" : "ChIJzZDOi_8EzkwR9cbwhaWQpQc",
         "reference"       :       "CnRhAAAAhbqiYKy7O8FbjsUEcvKuN7jfZjXHSciIRkGeiO5vgURk-
nAk7vTR7pN7DBAYlOLv-mmmbaqwhY6E5evBsGOKE1n5SN477E_kl6TtQZ08Yi-
fTYsKOK3x7MGJFMd8pkf48SbVb7IcIzMj-
2JTfVt9_BIQ6LvBqFAVBR6exQq3mhcQmxoUYvrzYHanj3FehHn7Py_wd2xC2Jk"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.40898199999999,
               "lng" : -75.71789199999999
            }
```

```
         },
         "id" : "764fff3fe2bbaabfe0d22421788c1ea13eeebfcb",
         "place_id" : "ChIJL2UZT0YEzkwR1Jquw7awMps",
         "reference"        :         "CoQBcwAAAAlB2j7AgNVfCsa1SP8snA3ueX-A_mRqzMso1iRYnB-
VQUl9VGSdjhzpP88BWuWdpvfQl33xcWnGb2IxsKt4Hp74b11I7-nGp5yvaRbwr9-
8EUzGEsRgPl19OqbnTm3xnBzCLboFx4OQKHHNOepWsGuYonOKDpbE9zPP0gSH33YvEhB8x2UPIjDK2_PrLJVVCyxE
GhSRT4olUvXY7DSGZwDnXhQmJt4jKQ"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4252638,
               "lng" : -75.7000494
            }
         },
         "id" : "a3f6159f3e87f1723359e1de8896e09335394ca4",
         "place_id" : "ChIJXeXB1v4EzkwROlLo7z-Ym9Y",
         "reference"      :      "CnRpAAAAfANHxS6MIScvz02WeRPmheqQCDbO2W8F89D8ZgAxlil4tGG-
ZuWDJdj1EX9cd71iPf_AqkwTNwxH_91vgcdN_wLx6hEM92_qZAhOb61xaIueVuuwjo_91dnQ137dM8lVHdsSHicXC
Q_ebKX5CNrLZxIQ6nWCwicCIn6qDwcdg8GDNxoUS5FWZY2Ajwr_ND1PEua10NBgp3Q"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4231479,
               "lng" : -75.6974386
            }
         },
         "id" : "7e73ddaf254326700d5237850523a82a0fe33cb8",
         "place_id" : "ChIJe4zljf8EzkwRCO7iUrXguCU",
         "reference"                                                                    :
"CnRwAAAA5hqES_AHxbIQRjqzEZpnhyHK6ulQXop8nJrelUu6AniNyN1v7RHIkkSBpsVI8Livc5O7elnG4Rftz7mL
jTu5Njxe1hkb0RAuA8y-
yQdybxWhDdZUrSxEYQ0cghzYlTgyefjCNcidaLhDMBcvRpUsRBIQ215aP3F1AXcb4gOzMCl0MRoUvUwREvI9h5bkM
gCAQXlOatHTZCo"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.423658,
               "lng" : -75.6959387
            }
         },
         "id" : "52bcf5ea69d70941afdfa0f4970ed3b1f36cff54",
         "place_id" : "ChIJWcia3f8EzkwRMjfNaBHWLsI",
         "reference"           :           "CnRmAAAA6-ENcFwc7KKoTrDDQOgFF9ZygahvfpXD2ZV-
2sP1aLT1Nm5pyMbPW_CfXbbpAhUPxl7Ij7SewKxn-COibBNBn3tk62app8FbqK76bH-Y2wUIjQvBUKwjPVE-
XsnThtUKgHPr0Nw6hTCvFumzlbrKchIQi_6Tu9F0FCev4pBl_xYP0xoUwYzN0UxUMrpR-fuR-Jq3XfBkB3U"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.42553530000001,
               "lng" : -75.7002806
            }
         },
         "id" : "6a662401b04b534c3d58b82d5bc4fc0b8d8fcacb",
         "place_id" : "ChIJJ2KVfFUEzkwRWM4O03KYaPQ",
         "reference"           :           "CnRpAAAAS7AeEoqBs9EpOGkDjnDx16b1PwwGsi-I3WQ-
YtITn514mvYXjdx5Ro5Nmh3gC6HxvHa1JtTtdDKLZ4Ps7yrd3AjEpIlTW1wf5X7jhf50M9USNA7DejbukhxHhF9if
nx7Hs9yB7l0SvLLVb3lINgBIRIQu42KbtPHMWl1sPRwE3R6YBoU3it3NGRGTkGM7xFj-oHpLgqAl4o"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.42468059999999,
               "lng" : -75.6974238
            }
         },
         "id" : "134f4ca59634155534f8eb5207074f76b767555f",
         "place_id" : "ChIJQXEsrP8EzkwRe-RSeUWxbOY",
```

```
         "reference" : "CmReAAAAu7QajYfUvYutTvRgpF9j7GErFgNCS5AyLege6z5Xn1zhZM914F_GJAz15-
qMnY4cFAVyMaeJy5Uo8iArkCxcYpX_gayf-q7Uf7bTES8g9YzonzSH0N7_HP-
aWHSyPfduEhBl9vB1xWLjq3eBgX6UlfkSGhQy_U8S9fPtmH18pMnUd1CnZwziqA"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4251258,
               "lng" : -75.6999307
            }
         },
         "id" : "1c8f52993203530 79d1c6b340c5fcf48ed15ea0e",
         "place_id" : "ChIJfzUWJ_8EzkwRpCZVyUWx1ho",
         "reference"                                                                    :
"CnRnAAAATtO0arFwGukXL7k4BEuyMJ5nT1gm6NL_SifrI9GoDCgxsnbU70VuWyw5jcY9yvqFE-
TqWIPfpBCNHHXJDkGP4iFMRDmecsesmofL3GXyTm4bl2MHdLIOmor_ZvOjD9PLqA7L130tm1hmRRQHVuG5CBIQcd_
CRBJNBm7ofxrKLfLYcxoUBu-sgbmRyxVcgN8Um1rF0LKFo04"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4224963,
               "lng" : -75.6982816
            }
         },
         "id" : "a33b3e1608f44b5b1942bb8eb42752c0d6a480fe",
         "place_id" : "ChIJ097UfFUEzkwRBohNPZsGWC8",
         "reference"              :              "CoQBfwAAAGtfUsC9cYgWDfPxAbak6kdcGGn4ZLd-
jfcZmyoyUPBA2FOW0M72Uy2t8mt47bmqqr6WyyOEtdUENRRMQBWhn8uPBCfTXG_RgzBXvRzLfF2ViMr8sxtTtkmn4
Z1c2aL_1CsGJV52fRCHVl5Jlx-Q7W3yvp52-rYB0tsk0FLrvQxtEhDJOilH-
27Rk78cbW8pIpE6GhQMNz35q3XkL5KHT2Xh_7wDJ0-MYg"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.41928,
               "lng" : -75.691987
            }
         },
         "id" : "a771f3a068c26762baa597037e9fabba90bbc257",
         "place_id" : "ChIJW91VqGUFzkwRS3UJWpYY840",
         "reference"                    :                    "CnRrAAAAcdcefaOZqgHI6Vy2M-
CI0I0tc5Y00e02CyqZ4mcDexe4iZzy3EJvjPefcQua0XTraz_cCjHWuW9CWr7XKc39oGAz-
C5yqJdGMFPYy7ayBO11xKPBiNJgu6UpWvT9G3w1b_H_OKT_iRXNPtsljB72eBIQUUwq-
1D5PJy9go518gj39RoUYkLvoMxbbqhpHKdCXAUUkrBFpuY"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.42083069999999,
               "lng" : -75.6989356
            }
         },
         "id" : "b0b98851c18fb82cfbaf3aa7afd484eed8427f67",
         "place_id" : "ChIJkzteF1UEzkwR20s94SmLCYk",
         "reference"             :             "CnRwAAAA26ZNI1Rj0ci-f0fbdh8Gji4MHRtEBuz-
5VQXTDA7Pl2KwEavHNq9ZhNrTSE4Oyg9raHEhemA48sH5mixCyJzWn0AQwstImoYIK_8aeiBGtygR33R5bf217Sau
Ppz8ioJlmNhAOk4KxIAtaVK_hZ3YRIQodDo-drCGOcrVDGXNpOBlhoU8KI23UT1n838GLM7tz3cxTbGpmI"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4236549,
               "lng" : -75.6971985
            }
         },
         "id" : "146c33b834ff8911dc0b106fefabdc55c878c61b",
         "place_id" : "ChIJWa_rlf8EzkwRErD31-1WZFY",
         "reference"         :         "CnRhAAAAKPuBRN-X1OsluU0oFK0mTOCox6AS0R4XUuQ2x8IW-
QBF1QGbNx_3Oa7G1n7Eh0fMrljTKIQSp58K6OjZ8GeC3xFl42L-olkLFUZsPzch21Cj-Pf_s-
```

```
Xqry5n2zwhEiQyEQXMel1bwbu0-IbcDI5SaRIQrgszmFrLFVe22qqkeliIqhoUGKn-
2Y4GspOSv92pcos3k0vcuyI"
      },
      {
         "geometry" : {
            "location" : {
               "lat" : 45.4247773,
               "lng" : -75.6995905
            }
         },
         "id" : "122b792d39beb4d1d31846dcbcc52841da3ff969",
         "place_id" : "ChIJh18uOVMEzkwRNLz4FxCgBUU",
         "reference"                                                                                    :
"CnRnAAAAa80UICuBpUdM38uZT8kaOIFSycRqkwQbzvyV2UiftoWJbajVYI3SxjmRbViBbwiDCZm0v0efdy4qeGay
mXwmn7f4ZuSoR8JCVJY_cOnQakeeYsKtlsI0HfaCNhgroUprWG9yQRZdcT_XuPPWOqMpJRIQ7dTF717f7Ao9rq7ul
cX5RhoUdgnE-hry9eI0B2HkR3CTdk0Hegk"
      }
   ],
   "status" : "OK"
}
```