Collaborative Itinerary Recommender Systems

Liviu Adrian COTFAS
Academy of Economic Studies, Bucharest, Romania
liviu.cotfas@ase.ro

Although several itinerary recommender systems have been proposed in the literature, they usually don't allow users to further improve the available set of tourist attractions by submitting information in a collaborative manner. In this paper we present a collaborative itinerary recommender system that relies on crowd sourcing to both expand and maintain current the dataset of attractions. The system combines advanced recommendations and hybrid multi-objective genetic algorithms in order to build personalized itineraries. While the recommendation algorithm allows users to easily find interesting tourist attractions, the multi-objective genetic algorithm offers the possibility of choosing between several Pareto-optimal itinerary solutions. A crowd sourcing approach is used in order to better predict the transport and visiting times. The user interface relies on the latest web technologies in order to support a wide array of mobile devices and to provide a rich user experience.

Keywords: *Tourist Itineraries, Recommender Systems, Genetic Algorithms, Collaborative Filtering, Mobile Web Application*

1 Introduction

Designing flexible, efficient and user-friendly mobile tour guide applications is of a great interest both from a commercial and research point of view. Such systems are useful for tourists visiting a location in a limited period of time. Without any support from a system, a user manually building an itinerary, should spend a lot of time prior to the trip, searching for information regarding the tourist attractions, reading facts about them and designing possible itineraries by taking into account factors such as opening hours, visiting durations, distances and available means of public transport. Conversely, itinerary recommender systems not only assist users to schedule the initial itinerary, but can also easily revise it during the trip. Several reviews of existing solutions can be found in [1] and [2].

An important issue in existing itinerary recommender systems is the lack of proper collaboration methods that could allow users to both add information regarding new tourist attractions and updated outdated information regarding existing ones. Therefore, due to the high costs involved in maintaining a large dataset, most existing systems only target a city or a limited geographic area [1] [3].

Although the internet contains a wealth of geospatially referenced information, little effort has been made until recently to harness this data. As a result, most systems presented in the literature feature a closed design that doesn't take advantage of the huge amount of information available in the geospatial semantic web. [4] proposes DBPedia Mobile, a location-aware semantic web client capable of extracting information from W3C Linking Open Data. Limited user collaboration is also taken into consideration, by allowing users to post reviews and add photos for existing tourist attractions. However the system does not allow adding new tourist attractions and does not considers any validation of the user submitted information.

Compared with existing approaches, the proposed system uses crowd sourcing and integration with LinkedGeoData [5] in order to both allow users to add new content and also integrate the existing information from the geospatial semantic web. Advanced recommendation techniques and multi-objective genetic algorithms are used in order to build highly customized itineraries.

The rest of the paper is organized as follows: section 2 presents the collaborative filtering algorithm, section 3 illustrates the chosen collaborative approach. Section 4 introduces a novel multi-objective algorithm for building tourist itineraries. Section 5 and 6expand upon the distributed architecture and present the mobile client application developed using the latest web technologies. The last section concludes the article and presents some future research guidelines.

2 Collaborative filtering recommendations

A collaborative filtering recommender algorithm that takes into consideration the user's previous ratings, the ratings of other people with similar profiles as well as the specific context of the user request, has been implemented in order to allow the users to easily find interesting locations.

Recommender systems [6] try to model the degree of utility of an item POI_j, belonging to the set of tourist attractions, POI, for a user U_i, belonging to the set of application users U, as a function $Recom(U_i, POI_j)$. The key problem is how to predict the estimated utility, $Recom^*(U_i, POI_j)$, for different sets of users and items, by knowing the utility only for some pairs. The value of the utility function is a real number between 0 and 5. A value closer to 5 indicates a higher preference of the user for the predicted item, while a value closer to 0 indicates a lower preference.

Collaborative filtering is a recommendation technique that recommends to the current user items that other users with similar profiles have liked in the past. MoTripAssistent uses a state of the art collaborative filtering approach based on matrix factorization [7], which was shown to offer better results compared with approaches based on similarity measures like Cosine or Pearson similarity. Latent factor models, including matrix factorization, transform both users and items to the same latent factor space. The latent space is used to automatically explain ratings by characterizing both products and users on factors automatically inferred from user feedback [8].Matrix factorization uses the idea of baseline predictors or biases to isolate the influences of the rating which are due to factors that belong either only to the items or only to the users.

The estimated utility for a user, $Recom^*(U_i, POI_j)$, is rounded to the nearest integer and is displayed using 0 to 5 stars as shown in left side of Figure 1.Users are asked to rate each tourist attraction at the end of the visit. The new ratings are added to the set of known user ratings and contribute to better future predictions $Recom^*\left(U_i, POI_j\right)$.

Tourist attractions are divided in $Ncat = 4$ categories shown in Equation (1) forming the category set, Cat.

$$Cat = \{Castles, Museums, Churches, Nature\} \quad (1)$$

The profile for a user of the application, U_i, contains the following information used to provide recommendations:

- demographic data: $UAge_i$, $UGender_i$;
- category preferences: $UCatPref_i = \{(Cat_k, PrefCat_{ik}) | 1 \leq \text{k} \leq 4\}$ where $PrefCat_{ik}$ represents the preference of the user for a category of Points of Interest, Cat_k, expressed with a value between 0 and 5. The user can set his preferences from the Profile module shown in the right side of Figure 1.

The recommendation value that also takes into consideration the temporary category preferences can be calculated as shown in Equation (2) where Cat_k is the category to which POI_j belongs. $\beta$is a value between 0 and 1 used to weight the importance of the temporary preferences. During the performed user studies $\beta = 0.5$ was used.

$$RecomTPref^*\left(U_i, POI_j\right) = \beta * Recom^*\left(U_i, POI_j\right) + (1 - \beta) * PrefCat_{ik} \quad (2)$$

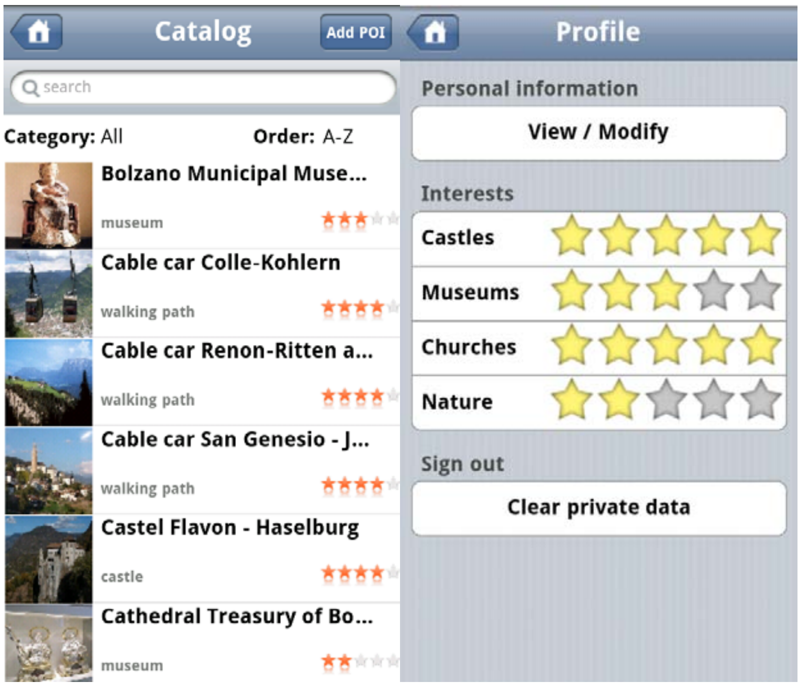

Fig. 1. a) Tourist attractions and predicted ratings; b) User interests

In order to provide meaningful recommendations, the context has to be taken into consideration. Context includes aspects that are not directly related to the user, such as time, weather information and traffic data. Thus, the system avoids recommending outdoor restaurants when it rains, or closed shops and museums. The contextual factors that can be selected by the users to be taken into consideration by the recommender algorithm are:

- $\boldsymbol{ContextF_1}$ - distance to the tourist attraction expressed in km;
- $\boldsymbol{ContextF_2}$ - temperature expressed as a value from the set $\{cold, warm, hot\}$;
- $\boldsymbol{ContextF_3}$ - weather as a value from the set $\{sunny,\ cloudy,\ rainy,\ snowing\}$
- $\boldsymbol{ContextF_4}$ - season as a value from the set $\{spring,\ summer,\ autumn,\ winter\}$;
- $\boldsymbol{ContextF_5}$ - weekday as a value from the set $\{workingday, weekend\}$;
- $\boldsymbol{ContextF_6}$ - time of the day expressed as a value from the set $\{morning,\ afternoon,\ evening,\ night\}$.

The value for the active context factors are determined automatically on the server using either the system time for $\boldsymbol{ContextF_4}$, $\boldsymbol{ContextF_5}$, $\boldsymbol{ContextF_6}$, or by connecting to an external weather web service for $\boldsymbol{ContextF_2}$, $\boldsymbol{ContextF_3}$. The value for $\boldsymbol{ContextF_1}$ is computed based on the GPS coordinates.

An Artificial Neural Network - ANN is used to predict the utility associated with a tourist attraction in a certain combination of contextual factors. The ANN has six input nodes associated to the contextual conditions and five output nodes corresponding to the five possible recommendation ratings.

$$Nhidden = \frac{Nattributes + Nclasses}{2} \qquad (3)$$

The number of neurons in the hidden layer was chosen as shown in Equation (3) where $Nattributes$ represents the number of input nodes and $Nclasses$ represents the number of output nodes. The back propagation implemented in Weka [9] was used in order to train the neuronal network on a set of manually classified instances.

3 Crowdsourcing

Due to high costs, the creation of geographic information was considered for a long time an area reserved to official agencies and large companies. Web 2.0 created a revolution in user generated content, which can be used to acquire and share information that could hardly be collected and organized in the past. Relying on people to generate content is also known as crowdsourcing. The term denotes a large group of people or a community handling tasks that have been traditionally associated with a specialist or a small group of experts [10]. Table 1 presents a comparison between the traditional data collecting approaches and the crowd sourcing one.

Table 1. Comparison between traditional GIS and crowdsourcing

	Traditional GIS	**Crowdsourcing**
Raw data quality	more accurate	less accurate
Associated cost	more expensive	less expensive
Users involved	skilled GIS users	savvy web-mapping users
Time required	depends on the resources involved	huge amounts of data can be collected in a short time

MoTripAssistent uses crowd sourcing in order to create a comprehensive database of tourist attractions. Contributors are able to:

- suggest new tourist attractions;
- suggest changes or contribute additional information for existing tourist attractions.

Compared with existing mapping approaches like [11] and [12], MoTripAssistent focuses on both automatically validating the user provided information and integrating with existing data sources like LinkedGeoData [5]. Administrators are allowed to define templates for each category and flexible validation rules.

The steps taken when a user submits a new tourist attraction, POI_i, are shown below. Great care has been taken in order to avoid submitting duplicate or incorrect information.

Step 1: The user chooses the position, Loc_i, of the new tourist attraction on the integrated map.

Step 2: The user selects the category, $CatPOI_i$, for the proposed tourist attraction. In order to avoid adding duplicated locations, the system displays all the POI matching the selected category around Loc_i.

Step 3: The user enters the name for the tourist attraction, $Name_i$ which is validated by the system using a vocabulary of not accepted words. The systems also checks for locations with similar names around Loc_i.

Step 4: Based on Loc_i,$CatPOI_i$ and $CatPOI_i$ the system searches for matching tourist attractions in LinkedGeoData. If a matching tourist attraction is found the corresponding fields are automatically filled. The user is required to enter any additional mandatory data defined in the template associated with the selected category, $CatPOI_i$.

Step 5: The new tourist attraction is added to the list of tourist attractions that must be validated before being added to the main set of tourist attractions.

4 Multi-objective evolutionary algorithm for building itineraries

We frame the problem of building tourist itineraries as an extension of the classic Orienteering Problem - OP, also known as the Time Dependent Orienteering Problem with Time Windows - TDOPTW. As the Orienteering Problem is known to be NP-hard, finding an exact solution for the TDOPTW in near real-time is considered to require a high computational power [13]. The proposed system uses a hybrid multi-objective genetic algorithm in order to generate itineraries in a short amount of time. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. In order to decrease the number of required generations, we use the following heuristic: each time we add a new POI to an itinerary we choose the one with the highest ratio between the associated score, S_i and the time required to visit the POI.

The itinerary building algorithm has as inputs:

- T_S - the start time;
- T_F - the finish time;
- L_S - the start location;
- L_F - the finish location;
- $POI_{Selected}$-the set of manually selected tourist attractions that will be included in all generated itineraries;
- $POI_{Favorites}$ - the set of attractions that were marked as favorites by the user;
- POI - the global set of tourist attractions;
- N^*_{Sol} - the number of requested solutions.

The output of the algorithm consists in a number of $N_{Sol} \leq N^*_{Sol}$ Pareto-optimal itineraries including both the tourist attractions and the traveling directions between them.

The considered optimization criteria are:

- **C1** - represents the useful visiting time calculated as shown in (4);
- **C2** - represents the average score of the visited locations calculated as shown in (5);
- **C3** - represents the diversity of the itinerary calculated as shown in (6).

Other criteria can also easily be taken into consideration. The criteria **C3** can also take into consideration the structure of the itinerary.

$$VisitT_j = \sum_{k=1}^{n_j} VisitD_k \tag{4}$$

$$MR_j = (\sum_{k=1}^{n_j} S_k)/n_j \tag{5}$$

$$DV_j = n_j \tag{6}$$

The start and finish locations, $L_S$and L_F, represent the GPS coordinates associated to the locations selected by the user either from the map or from the list of tourist attractions.

Each Point of Interest $i = 1 \ldots N_{POI}$ is characterized by:

- O_i - opening time;
- C_i - closing time;
- S_i - associated score;
- Fee_i - entrance fee;
- $VisitD_i$ - medium visit duration;
- Loc_i - GPS coordinates.

For each user u and tourist attraction i, $VisitD_i^u$ represents the real visiting duration and $VisitD_i^{*u}$ the predicted one. For a tourist attraction that has not yet been visited, the estimated visiting duration is computed as shown in (7), where $N_{POI}^{u,k}$ represents the number of tourist attractions in the category k to which the tourist attraction i belongs that have already been visited by the user u.

$$VisitD_i^{*u} = VisitD_i * \sqrt[N_{POI}^{u,k}]{\prod_{j=1}^{N_{POI}^{u,k}} \frac{VisitD_j^u}{VisitD_j}} \tag{7}$$

The algorithm is run until a maximum number of generations, G_{MG} is reached or until the best found solution doesn't change for G_{CG} consecutive generations. Depending on the purpose G_{MG} can be adjusted either for accuracy or speed. In order to increase the performance with compute in advance the minimum - $TravelD_{ij}^{min}$ and maximum time - $TravelD_{ij}^{max}$ required to travel between locations i and j. The values can be calculated for different hours or week days.

We have chosen an Elitist approach in which we use two populations, the normal one P_g and the population of non-dominated solutions in generation $g < G_{MG}$, E_g. The steps required for building an itinerary are shown in Figure 2.

Initialization: For $g = 0$, we create the initial population P_0, with a number of N_P itineraries called chromosomes. The population of non-dominated solutions is also initialized $E_0 = \emptyset$. In order to limit the number of required algorithm iterations, all itineraries in the initial population are generated valid.

First, the user selected tourist attractions, $POI_{Selected}$ are added to the new itinerary in a random manner.

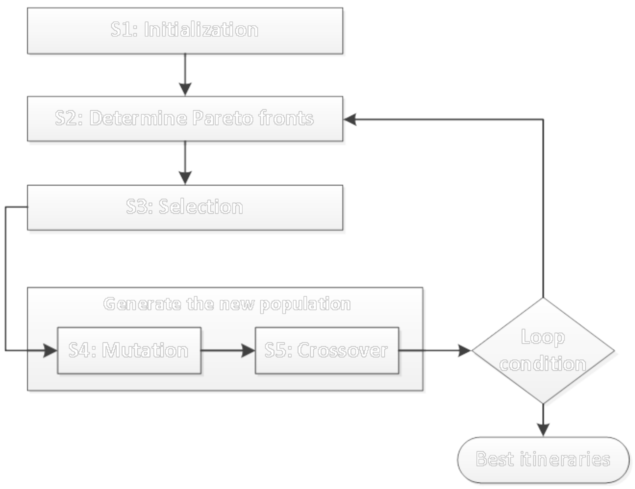

Fig. 2. Itinerary building algorithm

Afterwards, the remaining time is completed with tourist attractions selected from the set of favorites, $POI_{Favorites}$ and from the normal set, POI. A tourist attraction i can only be inserted if the arrival time, A_i, satisfies the conditions in (8) and (9).

$$A_i < C_i - VisitD_i \tag{8}$$

$$A_i + VisitD_i + TravelD_{iL_F} < T_F \tag{9}$$

The time needed to visit a tourist attraction can be computed as shown in (10) by summing the travel time from the previous location, the waiting time and the visit duration.

$$VisitTD_i = TravelD_{i-1,i} + WaitD_i + VisitD_i \tag{10}$$

The waiting duration, $WaitD_i$ can be computed as shown in (11), where A_i represents the arrival time.

$$WaitD_i = \max(0, O_i - A_i) \tag{11}$$

Evaluation: All N_P itineraries are evaluated based on the three selected criteria: **C1**, **C2**, **C3**. In order to determine the Pareto-optimal fronts we use an approach similar to the one in the NSGA-II algorithm [14].

Selection: As we have chosen an elitist approach in which we copy all non-dominated solutions from P_g in E_g. We remove all other solutions from P_g and we automatically include all solutions from E_g in P_{g+1}. Using mutation and crossover we generate $N_P - |E_g|$ solutions that are added to P_g in order to have $|P_{g+1}| = N_P$. If $|E_g| > N_P/2$ we use a clustering approach to select $N_P/2$ solutions as different as possible.

Mutation: Adds random variation to the evolving population. Two links are randomly chosen from the itinerary and all the locations between them that do not belong to the manually selected set of tourist attractions, $POI_{Selected}$, are removed. The itinerary is then completed by randomly inserting new POIs.

Crossover: Combines the features of two parent chromosomes to form new children by swapping corresponding itinerary segments of the parents. At first, one link offset is selected randomly. The resulting segments are swapped and a repair step is performed in order to preserve the manually selected tourist attractions from $POI_{Selected}$, in both itineraries. If necessary the POIs with the lowest S_i are removed in order to keep the itineraries valid.

The walking distances can be determined using either Dijkstra or A* for large graphs. The functionality is also provided by several libraries such as pgRouting and API's like

Google Maps Directions API.

5 System architecture

The proposed solution presented in Figure 3 relies on a multi-tier paradigm for both the server and the client implementations. Even though a client only architecture would offer several benefits, such as the possibility to work entirely offline, it is not a feasible option for our approach due to the big amounts of data used both by the semantic search and by the collaborative filtering algorithms. Therefore, we have chosen a mixed approach in which the resource intensive computations are performed on the server, while the simple ones are performed directly on the client. Client implementations rely on a local data persistence management solution in order to avoid unnecessary server requests. The server code is written in Java and uses Apache Tomcat as a web application server.

The **Interface Layer** has the role of facilitating the communication with the client implementations using HTTP requests, Web Service Calls and Socket Connections. Thanks to the multiple communication methods supported by this layer, client side applications can be implemented using a wide array of technologies.

The **Business Layer** includes the main functionalities of the system, implemented as semantically annotated web services [15]. The itinerary web service implements the multi-objective genetic algorithm described in the previous section, the recommender web service implements a context-aware collaborative filtering algorithm and the route finding web service implements the public transport route finding algorithm presented by the authors in [16].

The **Persistence Layer** stores information regarding the available tourist attractions, the public transport network, as well as user information. The POI Data database stores for all the attractions: the name, a short and a long description, photos, the location and the opening hours. The User Data database stores the user information as well as the tourist attractions ratings used by the recommender algorithm. A separate database is used to store the data required by the public transport route finding algorithm.

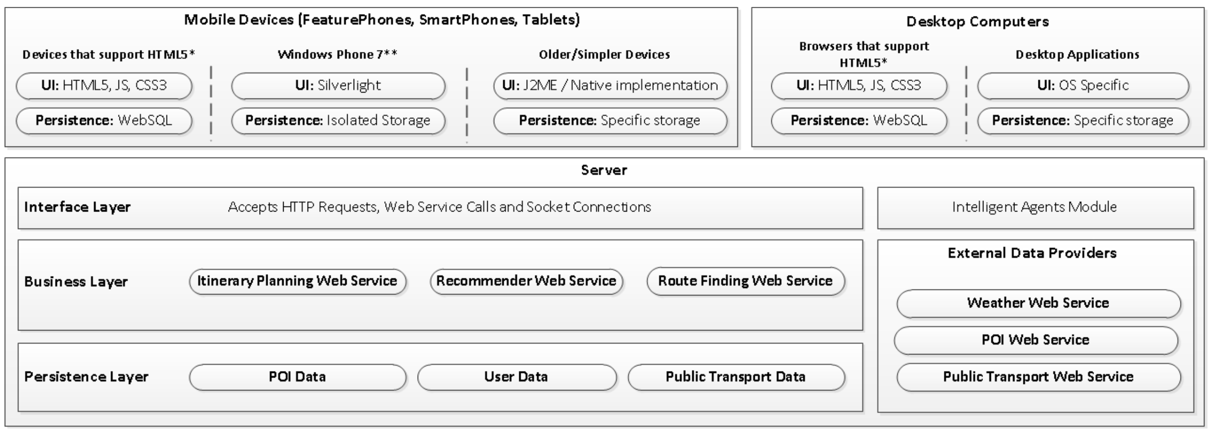
Mobile Devices (FeaturePhones, SmartPhones, Tablets)
Devices that support HTML5*
UI: HTML5, JS, CSS3
Persistence: WebSQL
Windows Phone 7**
UI: Silverlight
Persistence: Isolated Storage
Older/Simpler Devices
UI: J2ME / Native implementation
Persistence: Specific storage
Desktop Computers
Browsers that support HTML5*
UI: HTML5, JS, CSS3
Persistence: WebSQL
Desktop Applications
UI: OS Specific
Persistence: Specific storage
Server
Interface Layer
Accepts HTTP Requests, Web Service Calls and Socket Connections
Intelligent Agents Module
Business Layer
Itinerary Planning Web Service
Recommender Web Service
Route Finding Web Service
External Data Providers
Weather Web Service
POI Web Service
Public Transport Web Service
Persistence Layer
POI Data
User Data
Public Transport Data

Fig. 3. Distributed architecture

Using **External Data Providers**, such as external web services, the system collects weather forecasts, retrieves the changes in the list of available tourist attractions and updates the data for the public transport network.

An **Intelligent Agents Module** has the purpose of monitoring the users' behavior as well as of notifying them of any context changes, like weather changes or changes in the opening hours of the attractions included in the itinerary.

6 Mobile User Interface

The reference client implementation for mobile devices shown in Figure4 relies on the

latest standard web technologies such as WebSQLand WebStorage [17]to offer portability across different platforms as well as a rich user experience. W3C GeoLocation API [18] was used to determine and monitor the position of the user.
The application can either be used directly from the browser, or can be installed using a thin native wrapper that provides the required translation from JavaScript method calls to native method calls. The application works on all devices that comply with the HTML5 standard specifications including both smartphones and feature phones.
The steps that a user should perform in order to build an itinerary are presented below.
Step 1 - Create the list of tourist attractions
In order to build an itinerary, the user can both semantically search for POI as described in this paper or can select from a list of suggested tourist attractions. The list of suggestions is generated using a collaborative filtering algorithm that also takes into consideration the current context including weather, time, week day and season. The tourist attractions can either be added manually to the itinerary, corresponding to the $POI_{Selected}$ set, or can be added to the favorites list, corresponding to the $POI_{Favorites}$ set that are used as input parameters for the multi-objective itinerary building algorithm. As shown in Fig. 6, users can change the start time - T_S, the finish time - T_F, the start and finish locations - L_S, L_F.
Step 2 – Choosing the itinerary
The itinerary configuration screen shown in Figure 5 allows choosing if one or several itineraries will be generated. If the user chooses to generate multiple itineraries, a number of N_{Sol} possible itineraries are generated on the server using the multi-objective itinerary algorithm. The itineraries are displayed on the mobile phone including the score for the 3 optimization criteria **C1**, **C2**, **C3**. The system used previously recorded data to predict the visit duration for the current user.

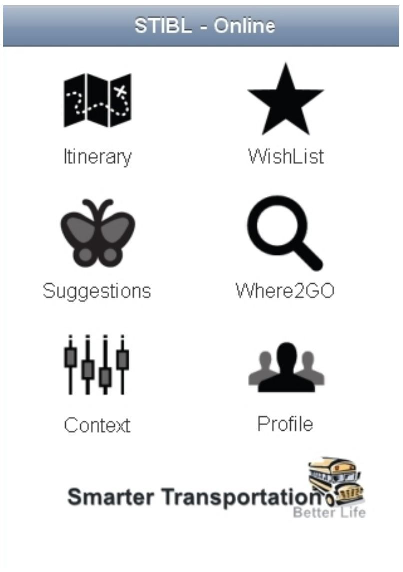

Fig. 4. Mobile HTML5 interface

Step 3 – Guidance during the tour
The selected itinerary can be viewed either as a list or using the map. In case the tourist exceeds the average duration for some of the points of interest, the itinerary is dynamically updated. Moreover, as shown in the fourth section, a crowd sourcing approach was implemented in order to determine the real visiting duration for the users of the application. For each user, the system constantly updates the difference for each category of POI between the current user's visiting duration and the average duration. This information is used to better predict the visiting durations in future itineraries.
Public transport route finding is also included in the application in order to allow the user to easily travel between the locations selected in the itinerary.

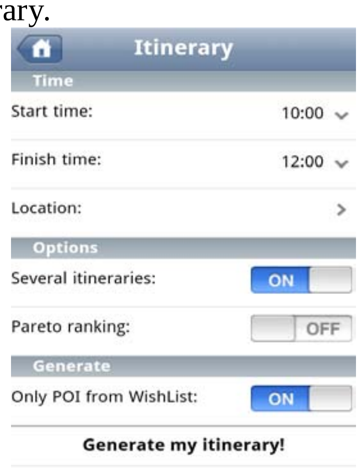
Itinerary
Time
Start time: 10:00
Finish time: 12:00
Location:
Options
Several itineraries: ON
Pareto ranking: OFF
Generate
Only POI from WishList: ON
Generate my itinerary!

Fig. 5. Itinerary settings screen

A comparison between native and mobile web applications built using the latest standard web technologies is provided in Table 2. As shown also in the table, developing native versions of an application for several mobile platforms requires more time and increases the costs due to the different development approaches characteristic for each platform

Table 2. Comparison between Native and Mobile Web Applications

	Native Applications	**Applications based on web technologies**
General		
Portable across many platforms	no	yes
Can run in the browser	no	yes
Can be installed	yes	yes
Development		
Technologies	different for each platform	the same for all platforms (HTML5, CSS3, JavaScript)
Local persistence	yes	yes
Possibility to work offline	yes	yes
Advanced hardware features	yes	yes
Resource intensive computations	better performance	worse performance
Others		
Accepted in Application Stores	yes	yes*
Easy to update	no	yes

Mobile web applications can be installed on mobile devices using a thin native wrapper. When running directly from the browser the local storage is limited to approximately 10Mb. This limitation can easily be avoided by using a native wrapper around the web application.

7 Conclusions

This paper presents an itinerary recommender system that uses crowdsourcing to build and maintain the list of tourist attractions. A multi-objective genetic algorithm using advanced heuristics has been implemented for building personalized itineraries. In order to provide recommendations the system uses an advanced collaborative filtering algorithm that takes context into consideration. Future work includes expanding the current approach for planning itineraries for several days. Moreover, using crowd-sourcing, we want to analyze the behavior of the tourists in order to better suggest future itineraries.

Acknowledgements

The author is grateful to Professor Ricci for the very useful suggestions and to the Free University of Bozen-Bolzano for kind hospitality during the time this work was carried out.

This article is a result of the project „Doctoral Program and PhD Students in the education research and innovation triangle”. This project is co funded by European Social Fund through The Sectorial Operational Program for Human Resources Development 2007-2013, coordinated by The Bucharest Academy of Economic Studies.

References

[1] A. Garcia, M.T. Linaza, O. Arbelaitz, and P. Vansteenwegen, “Intelligent routing system for a personalised electronic tourist guide,” *Information and Communication Technologies in Tourism 2009*, W. Höpken, U. Gretzel, and R. Law, eds., Vienna: Springer Vienna, 2009, p. 185–197.

[2] R. Kramer, M. Modsching, K. Hagen, and U. Gretzel, “Behavioural impacts of mobile tour guides,” *Information and Communication Technologies in Tourism 2007*, M. Sigala, L. Mich, and J. Murphy, eds., Vienna: Springer Vienna, 2007, p. 109–118.

[3] M. Kenteris, D. Gavalas, G. Pantziou, and C. Konstantopoulos, “Near-optimal personalized daily itineraries for a mobile tourist guide,” *IEEE Symposium on*

Computers and Communications, Riccione: IEEE, 2010, p. 862–864.
[4] C. Becker and C. Bizer, “Exploring the Geospatial Semantic Web with DBpedia Mobile,” *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 7, Dec. 2009, pp. 278-286.
[5] S. Auer, J. Lehmann, and S. Hellmann, “LinkedGeoData: Adding a spatial dimension to the Web of Data,” *The Semantic Web - ISWC 2009*, A. Bernstein, D. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, eds., Springer Berlin / Heidelberg, 2009, p. 731–746.
[6] F. Ricci, L. Rokach, B. Shapira, “Introduction to Recommender Systems Handbook,” *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor, eds., Boston: Springer US, 2011, pp. 145-186.
[7] L. Baltrunas, B. Ludwig, and F. Ricci, “Context relevance assessment for recommender systems,” *Proceedings of the 16th international conference on Intelligent user interfaces - IUI ’11*, New York, USA: ACM Press, 2011, pp. 287-290.
[8] Y. Koren and R. Bell, “Advances in Collaborative Filtering,” *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P.B. Kantor, eds., Boston: Springer US, 2011, pp. 145-186.
[9] Weka, “Weka,” 2011. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/. [Accessed: March. 7, 2011]
[10] S. Greengard, “Following the crowd,” *Communications of the ACM*, vol. 54, no. 2, Feb. 2011, p. 20-22.
[11] OpenStreetMap Foundation. OpenStreetMap.[Online]. Available: http://www.openstreetmap.org/.[Accessed: April 23, 2011].
[12] Google Inc. Map Maker. [Online]. Available: http://www.google.com/mapmaker.[Accessed: April 23, 2011].
[13] M. Kenteris, D. Gavalas, G. Pantziou, and C. Konstantopoulos, “Near-optimal personalized daily itineraries for a mobile tourist guide,” *IEEE Symposium on Computers and Communications*, IEEE Computer Society, 2010, p. 862–864.
[14] K. Deb, a Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, Apr. 2002, pp. 182-197.
[15] L.A. Cotfas, A. Dioşteanu, and I. Smeureanu, “Fractal web service composition framework,” *8th International Conference on Communications*, Bucharest: IEEE, 2010, pp. 405-408.
[16] L.A. Cotfas, M.C. Croicu, and D. Cotfas, “A Collaborative GIS Solution for Public Transport,” *Informatica Economică*, vol. 13, 2009, pp. 50-58.
[17] W3C, "Web Storage," 2011. [Online]. Available at: http://www.w3.org/TR/webstorage/. [Accessed: January. 15, 2010].
[18] W3C, "Geolocation API," 2011. [Online]. Available at: http://www.w3.org/TR/geolocation-API/. [Accessed: January. 15, 2010].

Liviu Adrian COTFAS is a Ph.D. student and a graduate of the Faculty of Cybernetics, Statistics and Economic Informatics. He is currently conducting research in Economic Informatics at Bucharest Academy of Economic Studies and he is also a Pre-Assistant Lecturer within the Department of Economic Informatics. Amongst his fields of interest are location based services, recommender systems, geographic information systems, evolutionary algorithms and web technologies.