Bringing Dynamic Queries to Mobile Devices: a Visual Preference-based Search Tool for Tourist Decision Support

Stefano Burigat, Luca Chittaro, and Luca De Marco

HCI Lab, Dept. of Math and Computer Science, University of Udine
Via delle Scienze 206, 33100, Udine, Italy
{burigat, chittaro, demarco}@dimi.uniud.it

Abstract. This paper discusses the design and development of a preference-based search tool (PBST) for tourists, operating on PDA devices. PBSTs are decision support systems that help users in finding the outcomes (e.g., multi-attribute products or services) that best satisfy their needs and preferences. Our tool is specifically aimed at filtering the amount of information about points of interest (POIs) in a geographic area, thus supporting users in the search of the most suitable solution to their needs (e.g., a hotel, a restaurant, a combination of POIs satisfying a set of constraints specified by the user). We focus on the design of an effective interface for the tool, by exploring the combination of dynamic queries to filter POIs on a map with a visualization of the degree of satisfaction of constraints set by the user. We also report the results of a usability test we carried out on the first prototype of the system.

1 Introduction

Mobile computing devices such as PDAs or high-end mobile phones are becoming more and more widespread and powerful. Due to their intrinsic portability, these devices are ideal for traveling users such as tourists or businessmen, who can benefit from a growing number of specific applications. In recent years, for example, there has been a growing interest towards the development of *mobile tourist guides* [1]. These guides provide users with easy access to various classes of information about places (e.g., history, entertainment, dining, transportation, ...), support users during navigation in an area and can allow one to take advantage of the most useful services for a given location (e.g., tour planning, online bookings, weather forecasts and so on). However, current mobile guides provide only limited help as *preference-based search tools* (PBSTs), i.e. applications that assist users in finding multi-attribute products or services that best satisfy their needs, preferences and constraints (e.g., the best hotel for staying overnight, the best place for dining, ...).

Existing PBSTs for tourist decision support on PDAs (e.g., [2]) are still limited in their flexibility and capabilities when compared to similar applications for desktop computers. In this paper, we present our work on the design and development of a PBST for tourists, operating on PDA devices. Our tool is specifically aimed at filtering the amount of information about points of interest (POIs) in a geographic area, thus supporting users in the search of the most suitable solution to their needs (e.g., a hotel, a restaurant, a combination of POIs satisfying a set of constraints specified by the user). Since device limitations pose constraints on what (and how) information can be visualized, the design of an effective interface for the tool is challenging. Our project focuses on combining dynamic queries to filter POIs on a map with a visualization of the degree of satisfaction of constraints set by the user.

The paper is organized as follows. Section 2 surveys related work on PBSTs. Section 3 presents our approach to the design of a PBST for PDAs, describing requirements and challenges and how we dealt with them. In Section 4, we report the results of a usability test we carried out on the first prototype of the system. Section 5 presents conclusions and future work.

2 Related Work

Searching for a product matching a set of requirements (user's preferences or constraints) is today a frequent task for users, e.g. in e-commerce sites. However, most search tools impose a fixed decision-making sequence on the user and typically visualize the results as ranked lists: products matching the user's request are ordered with respect to some attribute (e.g., alphabetically, by price, etc.). This approach becomes less and less usable as the number of product features and the complexity of user's criteria increase. Thus, researchers are studying how to improve the level of user support. Some of them have focused on modeling user's preferences, studying decision making processes and extending traditional decision theories (see [3] for a survey). Others have studied methods to incrementally elicit user's preferences [4]. Several advanced decision support systems for the search of multi-attribute products have been proposed in different domains (e.g., FindMe [5], ATA [6], Apt Decision [7], SmartClient [8][9]). Most of these systems are based on the *example critiquing* model of interactive problem solving: the system presents candidate solutions to the user based on an initial preference specification and the user either accepts a result or takes a near solution and critiques it by revising the current preference values. For example, using the SmartClient system for finding apartments, users can compose a critique to find a less expensive apartment than those proposed, by clicking on a pulldown menu next to the price attribute and selecting the "less expensive" option. Users can also set the weight of a preference, thus considering tradeoffs while searching for products. While in the real estate domain SmartClient offers only a textual list to visualize the results of a search, in the travel planning domain [8] it employs different visualization techniques such as maps, parallel coordinate plots and starfield displays. ScoreCat [10] does not only rank products as SmartClient, but also displays how each attribute scores in relation to user's preferences. These visualization techniques allow the user to better analyse solutions and augment her confidence level in the choices made.

A different approach to preference-based search is represented by *dynamic queries* [11][12]. Dynamic queries are typically used to explore large datasets, providing users with a fast and easy-to-use method to specify queries and visually present their results. The basic idea of dynamic queries is to combine input widgets (called "query devices" [13]), such as rangesliders, alphasliders [14], check buttons and radio buttons, with graphical representations of the results, such as maps, scatterplots [15] or other visual displays. By directly manipulating query devices, users can specify the desired values for the attributes of elements in the dataset and can thus easily explore different subsets of the data. Results are usually rapidly updated, enabling users to quickly learn interesting properties of the dataset. As shown with user studies [11][16], dynamic queries are more usable and powerful than lists, form filling, or natural language systems, to perform queries. They have been successfully employed in application domains such as real estate [16] and tourism [17].

The previously cited approaches have been developed for desktop systems, and PBSTs are still rare in the mobile computing domain area. The Michelin Guide for PDAs [2] is a commercial application allowing users to search for hotels or restaurants in a specific city by entering their preferences through drop-down lists and checkboxes. Results are then visualized as a ranked list ordered by quality and further information on an element can be retrieved by selecting it in the list. Some steps towards a more complex decision support tool for mobile devices have been recently proposed by Dunlop et al. [18][17]. In [17], they describe CityGuide, an application based on a geographic map that highlights tourist attractions in the city of Glasgow. The aim of the system is to support tourists' unstructured search. The current implementation contains an extensive restaurant guide that can be browsed through a set of dynamic filters. The implemented filters (restaurant type and price) can be activated by selecting them in the application toolbar: the first is controlled through a pop-up menu that provides a list of possible price ranges to choose from; the second through a pop-up window containing a set of check-boxes. The result of a query is immediately displayed on the map as a set of icons displaying the position of restaurants that pass through the filters. Users can then click on icons to obtain further details.

3 The Proposed Solution

The design of our PBST for searching POIs in a geographic area has been guided by different needs. On one side, behavior decision theories and user studies provide requirements that decision search tools and their interfaces should satisfy; in particular, as reported by [9], it should be possible for users to construct their preferences incrementally (that is, without being forced to specify all preferences initially and then examining the results), users should be able to specify their preferences in any order, the decision tool should display partially satisfied results and help users in decision tradeoff analysis, domain knowledge should be revealed whenever possible. Moreover, the development of PDA applications must face both technical and usability challenges with respect to desktop PC applications since mobile devices are characterized by scarce screen size (and resolution), limited computing performance and memory storage, and different input peripherals.

To satisfy all these requirements, we took an approach based on dynamic queries rather than employing the example critiquing model. Instead of letting the application compute the best solutions and propose them to the user (who can then refine her preferences by defining critiques to obtain better solutions), we designed an interface that allows users to specify their preferences incrementally by interactively imposing constraints on POIs attributes (through query devices) and immediately see the effects of their actions. By using this approach, users are in full control of the system, gain flexibility in exploring and analysing the solution space and possibly feel a greater confidence on the obtained results. However, implementing a PBST based on dynamic queries on a PDA is challenging because:

1. The standard behavior for dynamic query systems is to filter out those solutions that do not satisfy all specified preferences. A specific visualization technique must be instead employed to properly display partially satisfied results.
2. Both results and query devices (enabling users to perform searches) must be visualized at the same time. Since screen size is limited and users must be able to easily set various preferences on different attributes, a specific solution is needed.
3. Users must be able to quickly detect relevant attribute values associated with the elements under examination while performing queries. Again, the limited screen size forces to come up with proper solutions.

The following sections discuss how we dealt with these challenges. Section 3.1 will deal with the first issue, presenting our solution for the visualization of partially satisfied results, while section 3.2 will deal with the second and third issues by describing in detail the interface of our system. Finally, section 3.3 presents a typical example of system use.

3.1 Visualizing Partially Satisfied Results

The most common approach in visualizing the results of dynamic queries is to display all and only the elements that satisfy the query. However, as pointed out by Spence [19], this has a major drawback: only those objects whose attribute values satisfy all users' constraints are displayed. It is thus impossible for users to have a global view that shows also partially satisfied results, and to see how changing a query affects the hidden elements. In particular, elements whose attribute values fail to satisfy only a few constraints (e.g., only one) would be especially worthy of more detailed consideration. Moreover, when an empty result is obtained, the user has to backtrack without seeing how to find elements which are closest to the originally derived ones.

We propose a simple visualization technique to help users maintain contextual information on the whole dataset they are exploring. In our system, elements in the dataset (i.e., POIs) are represented by icons superimposed on a map of the geographic area, augmented by a vertical bar representing how much they satisfy users' queries (see Fig. 1).

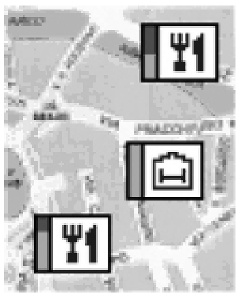

Fig. 1. Each element in the dataset is displayed as an icon representing its category, augmented by a vertical bar showing how much it satisfies users' queries.

This technique is an evolution of an idea presented by Fishkin and Stone [20] who introduced the concept of "real-valued queries" by assigning a real-valued score in the [0-1] range to each element in the dataset, based on the value of a specific attribute and on the particular scoring function that is being used. The score is visually presented by showing each element as a partially filled-in bar: the higher the score, the more the bar is filled. Instead of visually displaying a score dependent on the value of an attribute, we display a score dependent on the number of constraints satisfied by an element. We then fill the bar associated with each element with a green[1] area whose size is proportional to the number of satisfied constraints while the remaining area gets filled in red. This way, users can visually compare how much different elements satisfy the specified set of constraints obtaining a deeper understanding of the visualized dataset. Combining this visualization technique with dynamic queries, users can visually perceive the result of their queries by observing changes in the color-filled areas of the bars.

3.2 Interacting with the System

As reported in the previous section, we chose to display POIs as icons superimposed on a map of the considered area (see Fig. 2). This solution is much more natural and powerful than providing a simple ranked list of the results, which is the usual approach in PBSTs, because it provides spatial information by highlighting POIs positions.

[1] In the greyscale printed version of this paper, wherever colours are mentioned light grey in the figures corresponds to green, dark grey to red, black to blue.

We devote most of the screen to show the map. The bottom of the screen contains the menu bar and a toolbar which is initially empty except for the zoom icon. The map can be easily panned by dragging the pen on the screen in the desired direction.

Unlike current systems that usually support only one category of POIs at a time (usually hotels or restaurants), our tool allows users to deal with multiple categories at the same time. This provides users with much more information about the domain and is useful to compose more complex queries. Users can choose the categories by tapping on the item “POI” in the menu bar and then checking the proper boxes in a form. Each category is identified by an icon that will be used in the map to display elements. Once the form is closed, the icons of the selected categories appear in the toolbar in the lower part of the screen and the map gets populated with all the elements in those categories. Each element bar is initially fully green because there are no constraints specified.

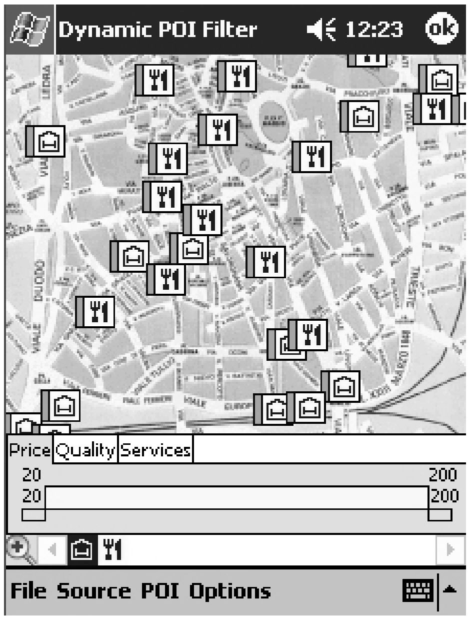

Fig. 2. The map displays all elements of the categories whose icons are shown by the toolbar in the lower part of the screen. A tabbed panel contains the query devices related to the currently selected category in the toolbar.

By tapping on a category icon in the toolbar, the user can specify preferences using the set of query devices associated to that category. These query devices, which are automatically generated by the system according to the type and the range of values of attributes, are organized in a tabbed interface placed above the toolbar, where each tab allows users to specify values for a single attribute of the considered elements. Figure 2 shows an example where the user has selected the “Hotel” category (see the highlighted icon in the toolbar) and can specify preferences for the “Price”, “Quality” and “Services” attributes by accessing the corresponding tabs. This layout allows users to specify their preferences in any order while visualizing results at the same time.

We implemented three types of query devices. The first is the classic rangeslider, usually associated with continuous attributes (e.g., price): the user acts on two independent handles to change the range of values of the related attribute. Figure 3 shows an example of this query device where the user has specified a price range between 40 and 80 Euro. The selected range is highlighted by using color and by showing the numeric value of the bounds. The design of the slider is slightly different from what can be usually seen on desktop interfaces. In particular, the handles have been placed under the body of the slider and they are aligned with the borders of the specified range area so as not to overlap when a small range is specified.

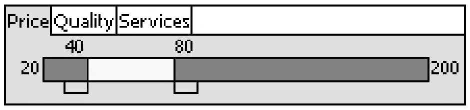

Fig. 3. The rangeslider.

The second query device (Fig. 4) is a modification of the rangeslider that can be used to deal with ordinal values (e.g., the number of stars of a hotel). Users operate this device as the classic rangeslider but its behavior is slightly different: when the user stops dragging on one of the two handles, the handle is automatically positioned at the nearest lower limit for the lower handle and at the nearest upper limit for the upper handle.

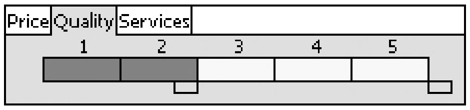

Fig. 4. The “discrete” rangeslider.

The third query device is based on the classic checkbox widget and can be used for multiple-choice attributes (e.g., types of services offered by a hotel). An example can be seen in Fig. 5: a POI satisfies the “Services” constraint if it has all the services specified through the checkboxes.

The system provides also a details-on-demand functionality: at any time, the user can tap on any POI on the map to obtain further information about it. As shown in Fig. 6, the tapped icon on the map becomes highlighted and details

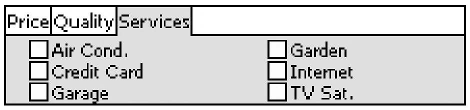

Fig. 5. A group of checkbox widgets.

are shown in the tabbed interface. In particular, a color coded line in the upper area of each tab tells if the POI satisfies (green line) or not (red line) the related constraint. In this way, a user can learn at-a-glance which constraints are satisfied (and which ones are not) by the POI, without having to examine the details. Moreover, attribute values for the selected POI are visualized in the query devices by using a blue horizontal bar (the same color as the highlighted POI) inside sliders and marking checkboxes with blue boxes. For example, in Fig. 6, the horizontal bar inside the "Price" query device shows the range of prices for the selected hotel, while, in Fig. 7, blue boxes highlight available services (i.e., "Air Conditioning", "Credit Card", "Garage" and "Garden") for the selected POI. The details-on-demand functionality aims at making the system easier to use by providing rapid access to information which is usually more difficult to obtain in traditional systems and that can be used as a guide for modifying a query. The additional tab named "Info" contains contact information on the selected POI such as name, address, phone number, etc.

Our query devices are not tight coupled [15]. In a dynamic query system using tight coupling, results of users' operations on a query device automatically trigger modifications to all other query devices so that only values associated with current solutions can be specified. Since in these systems only fully satisfied solutions are displayed, this behavior does not change the solution set and prevents users from specifying empty queries. On the other hand, tight coupling might influence the percentage of satisfied constraints for partially satisfied solutions, thus it cannot be used in our system. If it were applied, users might not be able to understand why some changes are taking place or they might think that a change is a consequence of their direct manipulation of a query device, while it is a consequence of query devices interrelations.

3.3 Using the System: a Real Scenario

In this section we will describe a typical session with the system, describing the steps needed to obtain the result and pointing out some features that help the user. We will refer to Fig. 8 for illustrative purposes.

The user of the system is a professor visiting a city for a two-day conference. She needs to find a hotel to stay overnight. After selecting hotels as the category of POIs to be displayed, she sets the price range she prefers (40-80 Euro) using the continuous range slider (Fig. 8a). She then sets the Quality constraint (asking for at least a three star hotel) using a discrete range slider (Fig. 8b) and the Services constraint (she wants air conditioning and prefers to pay with her credit card) using checkboxes (Fig. 8c). Then, she looks at the visualization, singles out

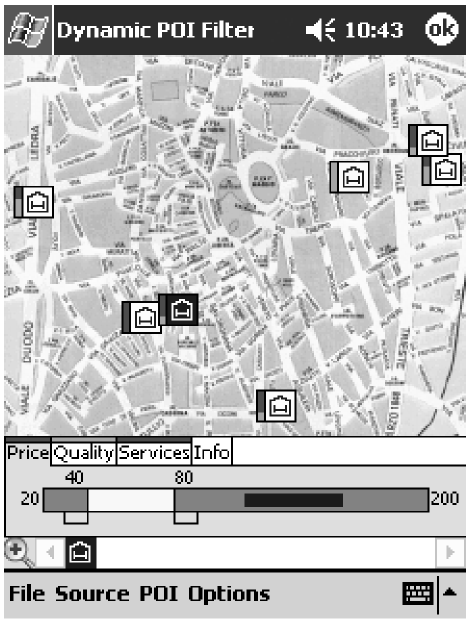

Fig. 6. The interface after the selection of a POI on the map. As shown by the color coded lines on each tab, the "Price" and "Services" constraints are not satisfied. A blue horizontal bar inside the "Price" query device shows the range of prices for the selected POI.

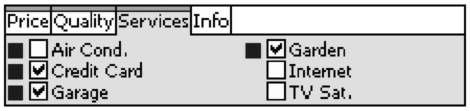

Fig. 7. Blue boxes highlight available services for the selected POI in the map.

a hotel satisfying all the specified constraints and taps it on the map to check its attributes (see Fig. 8d) and obtain contact information through the Info tab. She also checks the hotel that is nearest to the conference venue to know why it does not satisfy all her constraints and she immediately notices (by looking at the colored lines on the tabs) that it does not satisfy the Price and the Services constraints (Fig. 8e). In particular, the price range is too high for her. Then she wants to look for a restaurant near the chosen hotel. She thus selects the restaurant category and defines the constraints (she wants a restaurant that is open on Tuesday and does not cost too much, see Fig. 8f - 8h). Looking at the only two restaurants satisfying all the constraints she sees that they are far from her hotel (Fig. 8i). She then examines partially satisfied elements near her hotel and finds one that satisfies all constraints but "Type" (Fig. 8j). Anyway, this seems a good tradeoff for her needs and she proceeds getting contact information through the "Info" tab.

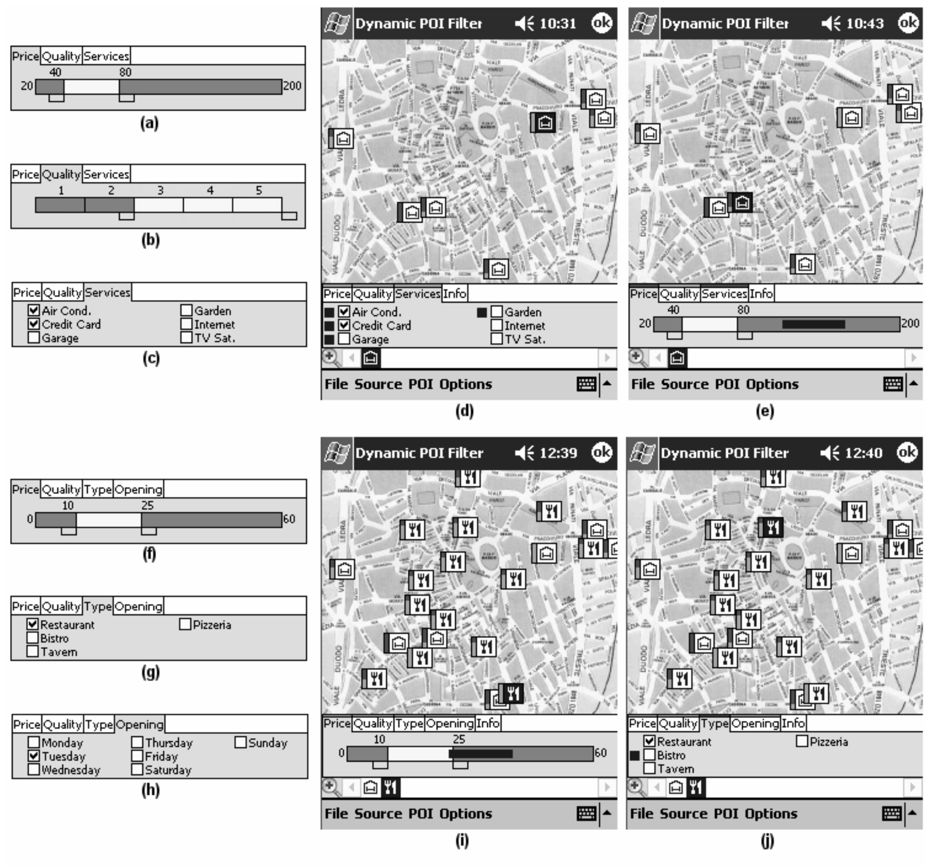

Fig. 8. Example: (a-b-c-d-e) finding a hotel, (f-g-h-i-j) finding the most suitable restaurant near the chosen hotel.

4 Usability Evaluation

We carried out a usability evaluation of the system to point out problems with the interface and obtain information to plan possible improvements.

Eight users, six male and two female, were recruited among the staff of our department to participate in the evaluation. The age of subjects ranged from 24 to 30, averaging at 26. All subjects were regular computer users but only two of them had previously used PDAs.

The evaluation procedure was organized in four phases and lasted a total of 30 to 40 minutes for each user. An iPAQ h3970, featuring an Intel XScale 400MHz processor and a 320x240 screen resolution, was used as testing device. During the evaluation, the experimenter observed users' interaction with the system and took notes about their behavior. In the first phase, the experimenter showed to the participant how to use the system, explaining all its features. In the second phase, the user was asked to try the system for a limited period of time (5 minutes) to become familiar with the available functions. In the third phase, users were asked to carry out a series of predefined tasks which took into consideration all system features. In particular, they had to specify some queries, ranging from simple (requiring to specify a single constraint) to complex ones (requiring to specify multiple constraints on different categories), and then point out in the map the elements satisfying all the constraints or, in the case of an empty result (that is, no fully green bar for any POI), those elements which best satisfied the constraints. The fourth phase of the experiment was based on a questionnaire consisting of 28 questions (inspired by user interaction satisfaction questionnaires such as QUIS [21]). Users were asked to rate system features on a 7-value Likert scale, where higher values corresponded to better ratings. More specifically, questions concerned widgets (ease of use, expected behavior, affordance), tabbed interface (ease of use, usefulness of colored tabs), visualization of results (usefulness, understandability), graphical interface (colors, aesthetics, organization), overall system usefulness and ease of use. Users could also add free comments about the system and its features.

4.1 Results

Most of the features received high ratings in the adopted Likert scale and positive comments. The mean value for the 28 questions was 5.6, with mean values for single questions ranging from a minimum of 4.0 to a maximum of 6.9. The worst results concerned the interaction with the rangeslider and discrete rangeslider query devices. In particular, users had difficulties in specifying the range of values in the rangeslider since they were asked to set values precisely and no fine tuning mechanism was available. In the case of the discrete range slider, users considered it too complex and too time-consuming. A more simple implementation based on checkboxes would have been preferred. Users expressed an high degree of satisfaction for the possibility to maintain a global view on all POIs, for the availability of the colored lines on the tabs that informed about the satisfaction of a constraint, for the graphical interface (colors, aesthetics, organization). Globally, the system was judged useful (6.8) and its features were considered easy to use (6.3) and easy to understand (6.3).

From the observation of user interaction with the system and from users' written comments we derived the following major considerations:

1. Users should be able to hide those POIs that do not satisfy constraints they consider to be most relevant (i.e., high-priority constraints).
2. If POIs fall outside the currently displayed part of the map there should be an indication of how many of them satisfy all the specified constraints and where to find them.
3. POIs satisfying all constraints should be made visually more evident.
4. POIs belonging to the currently explored category should be highlighted.
5. The discrete rangeslider would be better replaced by checkboxes.
6. The handles of the rangesliders should have better affordance.

7. Fine tuning of the rangesliders should be available.

All but the first two items require only minor changes to the current implementation and will not change system's behavior. Introducing priorities will allow users to filter out those POIs that do not satisfy constraints that cannot be relaxed. The behavior with high-priority constraints is similar to traditional dynamic query systems but allows the user to control when and on what attributes to apply the filtering. Note that this is different from using soft constraints, that is expressing users' criteria as a scale of preferences using weights [22]. Providing information on out-of-view POIs is a more difficult issue. A possible solution to this problem could be to adapt a technique such as Halo [23], which helps in visualizing off-screen locations on small-screen devices.

5 Conclusions and Future Work

Powerful and flexible PBSTs for PDAs, supporting users while traveling (and complementing existing applications such as mobile tourist guides), are still lacking. The work described in this paper is a first investigation to build such systems, and has been specifically aimed at allowing users to explore and filter data about POIs. We have proposed an approach based on dynamic queries and a constraints visualization technique that allows to better support users in making decisions. PDA limitations, in particular the reduced display area, have been faced in our system by adopting solutions such as tabbed organization, tailored versions of standard query devices, details-on-demand.

We are currently planning an experimental evaluation that will compare our system with a traditional PBST based on the use of form filling to specify preferences and the use of ranked lists to display the results. We will also compare our visualization technique with a slightly modified version of it based on the standard dynamic query paradigm (that is, partially satisfied solutions are not visualized). We will also improve the system by adding other features such as the capability to automatically set preferences based on past user's behavior.

Besides the current application to tourism, we will investigate if the approach we proposed (in particular, the constraint visualization technique) can be applied to other application areas as well in the mobile context.

6 Acknowledgments

Our research has been partially supported by the Italian Ministry of Education, University and Research (MIUR) under the project "Web-based management and representation of spatial and geographical data", subproject "User Interfaces for the Visualization of Geographical Data on Mobile Devices".

References

1. Baus, J., Cheverst, K., Kray, C.: A Survey of Map-based Mobile Guides. In: Map-based mobile services - Theories, Methods, and Implementations, Springer-Verlag (2005) 197–216
2. Michelin Guide for PDAs, http://www.viamichelin.com/viamichelin/gbr/tpl/psg /produits/htm/pda_guide_michelin.htm
3. Doyle, J., Thomason, R.: Background to Qualitative Decision Theory. AI Magazine **20** (1999) 55–68
4. Pu, P., Faltings, B., Torrens, M.: User-Involved Preference Elicitation. In: International Joint Conference on Artificial Intelligence (IJCAI 03) Workshop on Configuration. (2003)
5. Burke, K., Hammond, K., Young, B.: The FindMe Approach to Assistive Browsing. IEEE Expert **12** (1997) 32–40
6. Linden, G., Hanks, S., Lesh, N.: Interactive Assessment of User Preference Models: the Automated Travel Assistant. In: Proc. Conference on User Modeling (UM 97), Springer-Verlag (1997) 67–78
7. Shearin, S., Lieberman, H.: Intelligent Profiling by Example. In: Proc. Conference on Intelligent User Interfaces (IUI 2001), ACM Press (2001) 145–151
8. Pu, P., Faltings, B.: Enriching Buyers' Experiences: the SmartClient Approach. In: Proc. Conference on Human Factors in Computing Systems (CHI 2000), ACM Press (2000) 289–296
9. Pu, P., Kumar, P.: Evaluating Example-based Search Tools. In: Proc. Conference on Electronic Commerce (EC 04), ACM Press (2004) 208–217
10. Stolze, M.: Comparative Study of Analytical Product Selection Support Mechanisms. In: Proc. INTERACT 99, IFIP/IOS Press (1999) 45–53
11. Ahlberg, C., Williamson, C., Shneiderman, B.: Dynamic Queries for Information Exploration: an Implementation and Evaluation. In: Proc. Conference on Human Factors in Computing Systems (CHI 92), ACM Press (1992) 619–626
12. Shneiderman, B.: Dynamic Queries for Visual Information Seeking. IEEE Software **11** (1994) 70–77
13. Ahlberg, C., Truvé, S.: Exploring Terra Incognita in the Design Space of Query Devices. In: Proc. Working Conference on Engineering for Human Computer Interaction (EHCI 95), Chapman & Hall (1995) 305–321
14. Ahlberg, C., Shneiderman, B.: The Alphaslider: a Compact and Rapid Selector. In: Proc. Conference on Human Factors in Computing Systems (CHI 94), ACM Press (1994) 365–371
15. Ahlberg, C., Shneiderman, B.: Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In: Proc. Conference on Human Factors in Computing Systems (CHI 94), ACM Press (1994) 313–317
16. Williamson, C., Shneiderman, B.: The Dynamic HomeFinder: Evaluating Dynamic Queries in a Real-Estate Information Exploration System. In: Proc. Conference on Research and Development in Information Retrieval (SIGIR 92), ACM Press (1992) 338–346
17. Dunlop, M., Morrison, A., McCallum, S., Ptaskinski, P., Risbey, C., Stewart, F.: Focussed Palmtop Information Access Combining Starfield Displays and Profile-based Recommendations. In: Proc. Mobile HCI 2003 Workshop on Mobile and Ubiquitous Information Access, Springer-Verlag (2004) 79–89
18. Dunlop, M., Davidson, N.: Visual Information Seeking on Palmtop Devices. In: Proc. Conference of the British HCI Group (HCI 2000), Springer-Verlag (2000) 19–20

19. Spence, R.: Information Visualization. Addison-Wesley & ACM Press (2001)
20. Fishkin, K., Stone, M.: Enhanced Dynamic Queries via Movable Filters. In: Proc. Conference on Human Factors in Computing Systems (CHI 95), ACM Press (1995) 415–420
21. Chin, J.P., Diehl, V.A., Norman, K.: Development of an Instrument Measuring User Satisfaction of the Human-Computer Interface. In: Proc. Conference on Human Factors in Computing Systems (CHI 88), ACM Press (1988) 213–218
22. Keeney, R., Raiffa, H.: Decisions with Multiple Objectives: Preferences and Value Tradeoffs. John Wiley & Sons (1976)
23. Baudisch, P., Rosenholtz, R.: Halo: a Technique for Visualizing Off-Screen Locations. In: Proc. Conference on Human Factors in Computing Systems (CHI 2003), ACM Press (2003) 481–488