International Journal on Artificial Intelligence Tools
Vol. 18, No. 5 (2009) 717–738

World Scientific
www.worldscientific.com

e-TOURISM: A TOURIST RECOMMENDATION AND PLANNING APPLICATION

LAURA SEBASTIA, INMA GARCIA, EVA ONAINDIA, CESAR GUZMAN

Universidad Politecnica de Valencia,
Dpt. Computer Science, Valencia, Spain
{*lstarin,ingarcia,onaindia,cguzman*}*@dsic.upv.es*

e-Tourism is a tourist recommendation and planning application to assist users on the organization of a leisure and tourist agenda. First, a recommender system offers the user a list of the city places that are likely of interest to the user. This list takes into account the user demographic classification, the user likes in former trips and the preferences for the current visit. Second, a planning module schedules the list of recommended places according to their temporal characteristics as well as the user restrictions; that is the planning system determines how and when to realize the recommended activities. Having the list of recommended activities organized as an agenda (i.e. an executable plan), is a relevant characteristic that most recommender systems lack.

Keywords: Recommendation; AI planning; tourism.

1. Introduction

Nowadays, most people who plan a trip or a day-out will first initiate a search through the internet. More and more people are aware of the advantages of the new technologies for planning leisure activities[25] as an increasing number of companies and institutions offer tourist information which is easily accessible through web services. However, travelers usually have a limited knowledge of the city to visit and they are unaware of the local artistic, social or entertainment places. A user may find a large amount of information about the city, but he may invest a long time selecting the activities he prefers and organizing them to profitably spend a day-out.

e-Tourism is a web application that generates recommendations about personalized tourist tours in the city of Valencia (Spain). It is intended to be a service for foreigners and locals to become deeply familiar with the city and plan leisure activities. *e-Tourism* makes recommendations based on the user's tastes, his demographic classification, the places visited by the user in former trips and, finally, his current visit preferences. One of the main components of *e-Tourism* is the planning module which is aimed at scheduling the recommended activities according to their duration, the opening hours of the places to visit and thegeographical distances

between places (time to move from one place to another). Thus, the *e-Tourism* output is a real agenda of activities which not only reflects the user's tastes but also provides details on how and when to perform the recommended activities.

The main component of *e-Tourism* is the *Generalist Recommender System Kernel* (*GRSK*) module, a domain-independent, taxonomy-driven recommender system that uses a mixed hybrid recommendation technique, fed by the recommendations obtained from different algorithms.

This paper describes the main characteristics of *e-Tourism*, devoting a special attention to the planning aspect. Section 2 summarizes the state-of-the-art of similar recommenders. Section 3 gives an overview of *e-Tourism* and introduces an scenario that will be used as an example of the *e-Tourism* working model. Section 4 describes the knowledge representation. Sections 5 and 6 detail the GRSK and the planning subsystems, respectively. We finish with some conclusions and future work.

2. Background

A **recommender system** (*RS*)[19] is a personalization tool that attempts to provide people with a list of information items that best fit their individual tastes. A RS infers the user's preferences by analyzing the available user data, information about other users and information about the environment. In summary, a RS offers the possibility of personalizing the information filtering so that only information tailored to the user's needs and preferences is shown. The adequacy of recommendations depends on the amount of available information.[19] However, the task of introducing information should not be too tedious for the user, so the RS must be able to infer new data items and enrich the user profile as the person interacts with the system.[17]

Four **basic recommendation techniques** are distinguished in RS[4]: demographic, collaborative, content-based and knowledge-based techniques:

- Demographic recommendation:[16] the user is classified into a demographic category by taking into account his preferences model and the recommendation is given according to the category he belongs to. The main advantage of using this technique is that it always provides a recommendation since it does not require neither user rating nor information about other users. It is the simplest technique but also the least accurate.
- Collaborative recommendation:[22] it is the most common technique. It recommends items selected by other users with a similar preferences model than the current user. This technique requires information from a large number of users to obtain an accurate recommendation, however it allows incorporating new elements in the recommendation — characteristic which is usually welcome by the users.
- Content-based recommendation:[17] each item is defined by its features and the recommended items are those with similar features that the user has rated positively in his historical interaction with the RS. A drawback of this technique is

that the recommendation only includes similar items to those already selected by the user.

- Knowledge-based recommendation:[24] each item is assigned information about how it satisfies a user's need. This technique establishes a relationship between a user's need and a recommendation thus suggesting products based on inferences about the user's needs and his/her preferences.

Therefore, each recommendation technique exhibits some advantages and disadvantages.[1] A common solution adopted by many RS is to combine these techniques into an hybrid RS[16] thus improving recommendations by alleviating the limitations of one technique with the advantages of others. For example, a system resulting from the combination of the collaborative and the content-based recommendation techniques will be able to recommend items similar to those selected by the user, and therefore close to the user's preferences, but also new items that have been selected by similar users. Some hybrid recommendation techniques are[4]: weighted, mixed, switching or cascade. The difference between them lies in the way the different RS are combined.

Tourism is an activity strongly connected to the personal preferences and interests of people. For this reason, travel, leisure and tourism web sites tend to incorporate RS for simulating the interaction with a human travel agent.[6] Some examples of tourist web services that use a RS are: DieToRecs,[8] ITR[20] or Trip@dvice.[21]

However, the use of RS for traveling and leisure presents several limitations:

- The most common recommendation technique, the collaborative recommendation, presents some difficulties to be applied in this domain,[7] because it requires asking the users to rate a great variety of items. This represents a major shortcoming for this type of RS as visiting the same city is not a frequently done activity. Some systems use conversational techniques to mitigate this problem.[15]
- People usually travel on group trips (family, friends, etc.) and, therefore, the recommendation should meet the preferences of the majority of the group members.[2]
- Recommendations should not only depend on the user's preferences but also on the information about the environment[6]: distance between places, modes of transportation, season of the year, opening hours of places, etc.

The above points reveal that the use of RS in tourism and leisure together with the integration of planning techniques still presents some inconvenience that pose a long-term challenge in this particular field. Despite of these inconvenience, there is an increasing interest in the RS community for researching in this field. For example, the *eCTRL (eCommerce and Tourism Research Laboratory)* is developing several projects such as Trip@dvice, Harmonise, ETD Project or Harmo-TEN.

Additionally, it is interesting to have the recommended activities organized as an agenda in order to visit as many places as possible and to minimize the movements between places. The definition of a tourist plan is a time consuming task that

involves managing different kinds of information as opening hours, distance between each place to visit or the time spent on the visit. So, the task of the tourism web service is not only to help selecting the places to visit but also to help organizing a plan.

However, it is not so common to find services to generate a tourist route in a city. Some projects like Guide,[5] Crumpet[18] or DeepMap[9] are specially prepared to assist the user during the realization of the tour; for example, helping the user move one place to another or by providing context-aware information about the tourist attractions. However, the organization of a coherent agenda is totally left to the user. Star,[13] for instance, is a web-based system that assists the user in building a personalized agenda for a tour, but it it the user who must specify the places he desires to visit.

The above points reveal that the use of RS in tourism and leisure together with the integration of planning techniques raises a long-term challenge in this particular field.

3. *e-Tourism* Overview

We are developing a tool, called *e-Tourism*, whose goal is to compute a leisure and tourist plan for a user, taking into account his preferences and the information of the context where the visit will take place. Our system does not solve the problem of traveling to an specific place but it focuses on recommending a list of the activities that a tourist may enjoy in a city. It also considers activities timetables and distances between the activities to compute a leisure and tourist agenda. *e-Tourism* is composed of three subsystems (Figure 1): the *control* subsystem, the *Generalist Recommender System Kernel* (*GRSK*) subsystem and the *planning* subsystem.

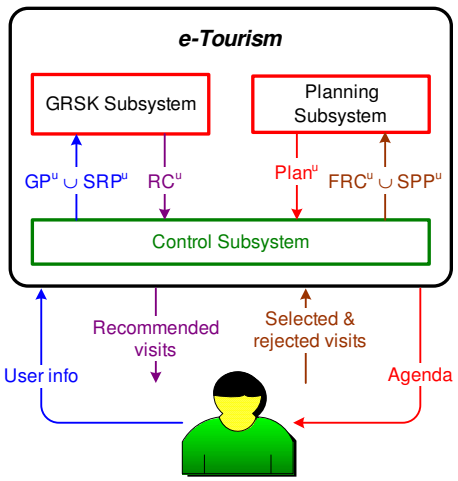

Fig. 1. *e-Tourism* system.

The GRSK is a general-purpose module whilst the remaining subsystems depend on the specific application.

To show the working model of *e-Tourism*, this section introduces a scenario that will be used as an example throughout the paper. John, 40 year-old, lives in a city near Valencia with his wife and two children (5 and 8 year-old). He usually goes out with his family. John was in Valencia some months ago and visited two churches: *San Miguel de los Reyes* and *San Nicolas*. He is planning a new visit to Valencia.

The *e-Tourism* first step is to **build the user profile**. The first time John uses *e-Tourism*, he must register and enter his personal details and general preferences. As general preferences, John likes visiting "*Science Museums*". With this information the system builds an initial user profile which will be updated accordingly with the relevance feedback obtained from the user; i.e. the activities which have been finally performed by John.

Besides this general information, each time John enters the system for a new visit he will be requested to introduce his specific preferences for the current visit (recommendation query): dates of the visit, his time schedule, whether he is on his own or with children, etc. John usually spends a day out with his wife and children, but his mother joins them in this new visit. John's mother likes "*Gothic Architecture*", so he introduces it as a particular preference for this visit. Moreover, their available time slot is from 12pm to 6pm.

The module in charge of building the user profile is the **control subsystem**, the core of *e-Tourism*. It works as an user interface, initiates the execution of the other subsystems and centralizes the exchange of information.

The second step is to **generate a list of activities that are likely of interest to the user**. This list is computed by the **GRSK subsystem**, whose input are the user's general and specific preferences. The GRSK computes the recommendations according to the current user's profile and other users' profiles (depending on the recommendation technique). The calculated recommendations constitute the list of proposed items to visit. Section 5 explains the functional behaviour of the GRSK.

Figure 5 shows the list of recommended activities for John's visit. Each item is associated a priority and the GRSK selects the ones that better suit John, i.e. the highest priority items. The final list of recommendations is composed of the items which appear in shadow. From this list of recommendations, John picks up the activities he is really interested in (marked with ✓ in Figure 2), and discards those ones he does not want to be included in the final agenda (X). The remaining items are marked as indifferent (∼).

The third step is to **compute the tourist agenda** with the selected activities of the previous step. At this stage the system schedules the activities according to the time restrictions of the user and the environment. The module in charge of computing the plan is the **planning subsystem**. The input of this module is the set of activities positively selected by the user (✓), the activities left as indifferent (∼), and other preferences necessary for planning like the user available time. The result is a tourist agenda containing the planned activities together with the time when

the activities should start and the estimated duration of each activity. Section 6 explains how this plan is computed.

The final step is to **process the user feedback**. When the user logs again in the system, he is asked to rate the activities in the last recommended plan. The information obtained from these ratings is used to improve the user profile and gain more suitable recommendations. The management of the user feedback is an ambitious task that is not still finished. We intend the system to dynamically adjust itself to offer higher quality recommendations by obtaining more information from the current user as well as learning from the decisions of all users.

4. *e-Tourism* Knowledge Representation

This section illustrates the different knowledge *e-Tourism* needs to provide an accurate tourist agenda.

4.1. *Taxonomy*

e-Tourism relies on the use of a taxonomy to represent the user's preferences and the items to recommend. The entities in a taxonomy are arranged in a hierarchical structure connected through a *is-a* relationship where the classification levels become more specific towards the bottom. In the tourism taxonomy, entities represent **concepts** that are commonly managed in a tourism domain like architectonic styles or types of buildings. The leaf nodes of the taxonomy are the **items** *e-Tourism* will recommend to the user and they are categorized by the lowest-level concept in the hierarchy, i.e. the most specific concept. Edges that connect an item to a concept are associated a value to indicate the **degree of interest** of the item (an activity in a tourism taxonomy) under the concept, i.e. as a member of the category denoted by the concept. More formally:

Definition 4.1. The *taxonomy* T is a directed graph (C, E), where C is the set of nodes of the graph which represent the taxonomy concepts and E is the set of edges that connect a concept with their successors. We distinguish two types of edges: $e_{c-c} = (c_j, c_k)$, which links a concept c_j with a successor concept c_k; and $e_{c-i} = (c_j, i, r_j^i)$, which links a concept c_j with an item i with a degree of interest $r_j^i \in \Re$.

e-Tourism utilizes a hand-crafted taxonomy that encompasses a set of concepts to describe tourist and leisure activities. These concepts are also used to classify the user's preferences and interests. The taxonomy is based on the "*Art & Architecture Thesaurus*[a]" which provides terminology about objects, concepts and places important to art, architecture and culture disciplines.

Figure 2 shows part of the tourism taxonomy. The leaf nodes of the graph are the activities that can be performed in the city of Valencia, e.g. "*Visit the Botanical*

[a]www.getty.edu/research/conducting-research/vocabularies/aat.

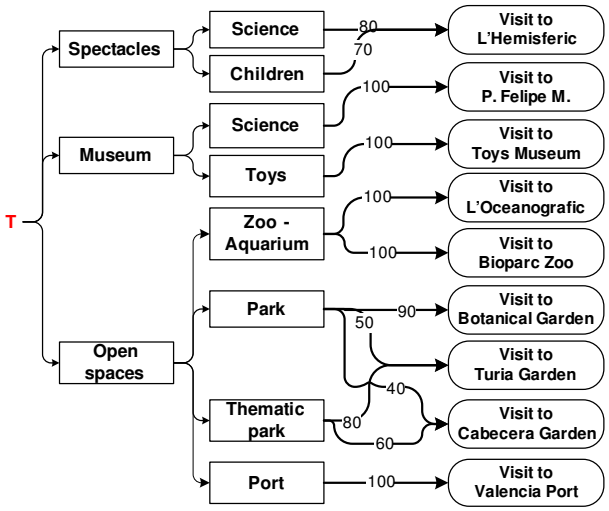

Fig. 2. Part of the *e-Tourism* taxonomy.

Fig. 3. Example of an item in the taxonomy.

Garden". As indicated above, the values of the edges connecting an activity with its most-specific concept show the interest of visiting that place. For example, both the *Botanical Garden* and the *Turia Garden* are parks, but the *Botanical Garden* is more worth visiting as a park than the *Turia Garden*.

Each leaf node of the graph represents an item to recommend (Figure 3), which in the particular case of the tourism taxonomy is described by a name, a short description and its location (address). It is also necessary to record the opening hours of places and buildings — which may be different depending on the season or the day in the week — and a standard duration of each activity — which can be personalized for an specific user. This information will be mainly used by the planning subsystem.

Additionally, an activity or item $i \in I$ is defined by a list of *features* F^i, which represent the incoming edges of each leaf node.

Definition 4.2. A *feature* is a pair on the form (c_n^i, r_n^i), where $c_n^i \in C$ is a concept defined in the taxonomy; $r_n^i \in [\kappa_1, \kappa_2]$ is the degree of interest of the item i under the concept c_n and $\kappa_1, \kappa_2 \in \Re$.[b]

For example, according to the taxonomy in Figure 2, if $i =$ "*Visit the Turia Garden*" then F^i could be set to $\{(Park, 50), (Thematic\ Park, 70)\}$. On the other hand, if $i' =$ "*Visit the Botanical Garden*" and this garden is considered to be a more interesting park than the *Turia Garden* then $F^{i'}$ will contain that feature with a higher degree of interest, i.e. $\{(Park, 70)\}$. The degree of interest can be dynamically updated through the user feedback.

Additionally, items are associated a numeric value AC^i (acceptance counter) to represent how popular the item i is among users; this value indicates how many times this item has been accepted when recommended.

4.2. *Information about the city*

We store the city map which comprises all the streets in the city with the following attributes: name, district, sections, length of each section and geographical coordinates. Moreover, we also represent the intersections of the different sections of each street. This information will be used by the planning subsystem to compute the geographic distances between the activities.

4.3. *User information*

In this section we detail the user information in *e-Tourism*. In a tourist domain, some pieces of the user information belong to he user's profile and are used by the GRSK (e.g. taste and preferences, historial interaction, etc.), and others are used by the planning process, like the visit date, the user available time or the user current location.

4.3.1. *User profile*

The profile[17] of a given user u defined in *e-Tourism* records:

- Personal and demographic details about the user like the age, the gender, the family or the country. In our scenario, the demographic details are: John, a 40 year-old man, lives in a city near Valencia with his wife and two children (5 and 8 year-old).
- The user general preferences model, denoted by GP^u, that contains the description of the types of items the user u is interested in. More formally: $GP^u = \{(c_n, r_n) : 1 \leq n \leq |C|\}$. John has defined as general preference: ("*Science Museum*", 70).

[b]In the examples along this paper we assume that $\kappa_1 = 0$ and $\kappa_2 = 100$.

- Information about the historical interaction of the user with the RS, that is, the set of items the user has been recommended and his degree of satisfaction with the recommendation. In our scenario the historical information contains two visits, {("*San Miguel de los Reyes*", 50), ("*San Nicolas*", 70)}, both classified in the taxonomy as "*Churches*".

4.3.2. *Recommendation query*

Each time the user enters the system for a new visit he will be requested to introduce his specific preferences for the current visit (arranged into a recommendation query), which may differ from his general preferences. A recommendation query contains the maximum number N of recommendations the user desires and the set of specific preferences, formally denoted as SP^u. SP^u is divided into specific recommendation preferences $SRP^u = \{(c_n, r_n) : 1 \leq n \leq |C|\}$ and specific planning preferences $SPP^u = \langle date, (Ts, Te), dur_{lunch}, dur_{dinner}, \textit{ user location}\rangle$, where *date* denotes the visit date, (Ts, Te) represent the user available time slot, dur_{lunch} and dur_{dinner} represent the time reserved for lunch and dinner, respectively, if the user wants the plan to include the time for meals and *user location* is the current geographical location of the user. In our scenario: $SRP^{John} = \{(\text{"}Gothic\ Architecture\text{"}, 100)\}$ and $SPP^{John} = \langle 10/8/2008, (12, 18), 1h30', 0, Astoria\ Hotel\rangle$.

5. The Generalist RS Kernel (GRSK)

The task of the *Generalist Recommender System Kernel (GRSK)*[10] is to generate the list of activities to recommend to the user. The GRSK uses the taxonomy hierarchy displayed in Figure 2 to classify the user's profile information and to generate the list of recommended activities. It has been designed to be *generalist*, that is independent of the current catalog of items to recommend. The GRSK can be applied in any application domain as long as the data of the new domain is defined as an organizational taxonomy representation. Figure 4 shows the GRSK architecture.

The **Engine** module is the core of the GRSK. It is the interface between the recommender system and the Control Subsystem of *e*-Tourism (see Figure 1). This module translates the user recommendation query into a set of data understandable by the RS, and, inversely, it converts the list of recommendations provided by the RS into the final recommendation data that are sent to the Control subsystem. The Engine module is also in charge of generating and updating the user profiles.

The **Control RS** module manages the recommendation process. Given a user profile and a specific recommendation query, the Control RS asks each different RS module for a list of recommendations. The Hybrid RS is then passed these lists of recommendations and it combines the information contained in these lists to generate the final lists of recommended items.

Specifically, the GRSK uses a **mixed hybrid recommendation technique** that combines the following basic RS techniques: demographic and content-based

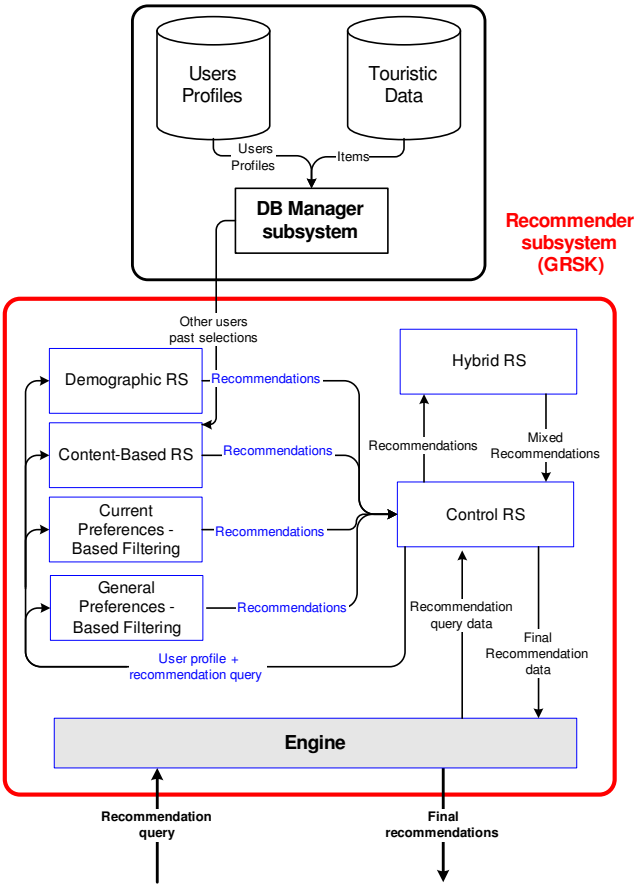

Fig. 4. GRSK architecture.

recommendations, general preferences-based filtering and current preferences-based filtering. We have defined an independent module for each basic technique, as Figure 4 shows. The design of the GRSK allows the developer to easily incorporate new basic or hybrid recommendation techniques. The recommendation generated by each basic module is independent from the others.

The **demographic RS technique** classifies the user into a demographic category according to the details of his profile and his general preferences (GP^u). This technique associates a list of the taxonomy concepts to a user type. In our scenario, John is classified as a "*Person with Children*", because this is the main characteristic of his profile. Therefore, the system considers, among other things, the following features to recommend activities: $\{(Zoo\text{-}Aquarium, 100), (Thematic\ Park, 90), \ldots\}$. We opted for a demographic RS because it is able to generate recommendation for the problem case of having a new user. In addition, it can recommend items which contain different characteristics from other previously recommended items.

	Recommended Item	Taxonomy		RS ratios				Acceptance Counter	Pr
		Concept	r	DRS	CBRS	GPBF	CPBF	%AC	
✓	**L'Oceanografic**	Zoo – Aquarium	90	70				70	**230**
x	**P. Felipe Museum**	Science museum	90			70		70	**230**
✓	**Valencia Port**	Open Spaces	90	55				50	**195**
✓	**L'Hemisferic**	Science spectac.	80	50				65	**195**
x	**Miguelete Tower**	Gothic	60				100	33	**193**
~	**Lonja**	Gothic	65				100	24	**189**
~	**BioParc**	Zoo – Aquarium	70	100				10	**180**
x	**Turia Garden**	Thematic Park	70	90				20	**180**
x	**Valencia Cathedral**	Church	70		30			68	**168**
	Reales Atarazanas	Gothic	50				100	9	**159**
	Quart Gate	Gothic	60				100	11	**171**
	Cabecera Garden	Thematic Park	60	100				10	**170**
	Serranos Gate	Gothic	55				100	12	**167**
	Jardín botánico	Park	70	50				20	**140**
	The Virgin Basílica	Church	80		30			40	**150**
	Santa Catalina Tower	Church	78		30			26	**134**
	Toys Museum	Toys museum	90	30				3	**123**
	San Juan de la Cruz	Church	50		30			4	**84**
	San Juan del Hospital	Church	44		30			5	**79**
	Santos Juanes	Church	35		30			3	**68**

Fig. 5. Items priority.

The **content-based RS technique** recommends a set of items by taking into account the features of the items previously accepted by the user. Our aim on using this recommendation technique is to increase the user satisfaction by recommending similar items to those already accepted. In our example, John has previously visited churches, so this RS will recommend other churches; more precisely, it recommends the *Valencia Cathedral*, as Figure 5 shows.

The **general preferences-based filtering** is an information filtering technique[14] that works with the general specific user preferences. This technique takes into account the general preferences (GP^u) specified by the user in his profile. For example, John specified *"Science Museums"* as a general preference, so this technique will recommend him visiting the *"P.Felipe Museum"*.

The **current preferences-based filtering** is an information filtering technique that works with the specific user preferences for the current interaction. Basically, it analyzes and stores the specific preferences (SRP^u) that differ from the general preferences (GP^u) without modifying the user profile. In our example, John has defined for the current visit "*Gothic Architecture*" as a specific preference and, for this reason, this RS recommends visiting the *Miguelete Tower* and the *Lonja* (see Figure 5).

Each recommendation technique calculates an independent list of items. These lists of items are then processed by the **mixed hybrid RS technique**. First, it computes a priority for each item in those lists:

$$Pr^a = \frac{AC^a}{\sum\limits_{\forall i \in I} AC^i} * (\kappa_2 - \kappa_1) + r^a_{RS} + r^a_{taxonomy}$$

where AC^a is the acceptance counter of the activity a, r^a_{RS} is the degree of interest of the activity a obtained from the RS technique and $r^a_{taxonomy}$ is the degree of interest of the activity a under the concept of the taxonomy.

For example, *L'Hemisferic* is classified into the category of Science spectacles (Figure 5) with the value $r^{L'Hemisferic}_{taxonomy} = 80$. Its acceptance counter (which measures how popular this activity is) is 65%. This item is recommended by the demographic recommender system (DRS) with an adequacy recommendation ratio of $r^{L'Hemisferic}_{DRS} = 50$. The priority of L'Hemisferic is calculated as:

$$Pr^{L'Hemisferic} = 65 + 50 + 80 = 195\,.$$

The hybrid RS combines the items in each RS list to obtain a single list of recommended items, which is ordered according to the computed priority. In case an item appears in more than one list (that is, it has been selected by several RS techniques), we only consider the appearance with the highest priority.

The engine module selects the N best recommendations, which are the set of recommended items for the user u (RC^u). Each recommended activity in RC^u is denoted by a pair of the form $\langle a, Pr^a \rangle$.

Figure 5 shows the initial set of items to recommend to John and the priority values. The columns *DRS* (demographic RS), *CBRS* (content-based RS) and *CPBF* (current preferences-based filter) show the ratio each technique assigns to the recommended item. The shadowed items are those that make up RC^{John}, that is the items recommended to John.

6. Planning Subsystem

Following with our scenario, once the GRSK has computed the set of recommended activities RC^{John}, John is shown these activities and asked to mark as selected (✓) those activities he is interested in and as rejected (X) those activities he does not want to perform in this occasion. The remaining activities are considered as indifferent (∼). As Figure 5 shows, John selects visiting *L'Oceanografic*, the *Valencia Port* and *L'Hemisferic*, and rejects visiting the *Miguelete Tower*, the *Turia Garden* and the *Valencia Cathedral*. The final list of activities filtered by John (FRC^{John}) will contain the selected activities plus those marked as indifferent. The list FRC^{John} is sent to the planning subsystem joint with his specific planning preferences. The planning subsystem analyzes this information and builds the user-adapted planning problem whose solution will be the tourist agenda.

6.1. *Building the planning problem*

The planning subsystem manages three groups of different data:

(1) The user's specific planning preferences SPP^u (see Section 4.3).
(2) The filtered recommended activities FRC^u which is a list of tuples of the form $\langle a, Pr^a, si^a \rangle$, where Pr^a is the priority computed by the GRSK for activity a

$(\langle a, Pr^a \rangle \in RC^u)$ and si^a is a value in the set $\{selected, indifferent\}$ which indicates whether the user has selected the activity a or has no preference over it (the rejected activities are not considered at this stage).

(3) The information about each activity a in FRC^u, which is a tuple of the form $\langle a, open_a, close_a, location_a \rangle$, where $open_a$ and $close_a$ indicate the opening hours of activity a (taking into account the date of the visit) and $location_a$ is the address of the place where the activity takes place. These values are extracted from the item information (see Figure 3).

Figure 6 summarizes the information managed by the planning subsystem of each selected/indifferent activity from our scenario. All these data are properly analyzed and combined to build the user-adapted planning problem. In this case, the final solution plan will contain two types of actions: the performance of the selected/indifferent activities (set A), and the movement actions to go from one activity to the next one (set M). Each activity $A_j \in A$ is described by a pair (dur^{A_j}, u^{A_j}). The duration dur^{A_j} of the activity A_j depends on a user classification that distinguishes several types of user. For example, Figure 3 shows the duration of the activity *"Visit to L'Ocenographic"* for four different types of user (user classification). On the other hand, the utility u^{A_j} of an activity A_j is a value computed in the following form:

$$u^{A_j} = \begin{cases} Pr^{A_j} & \text{if } si^{A_j} = selected \\ Pr^{A_j} * \alpha & \text{if } si^{A_j} = indifferent \end{cases}$$

where $\alpha \in [0, 1]$ is a parameter to weigh the relative importance of the indifferent activities (see Figure 6).

Additionally, we add two more activities to the set A, namely, $A_{|A|+1} = (dur^{lunch}, u^{lunch})$ and $A_{|A|+2} = (dur^{dinner}, u^{dinner})$, to represent the actions "*having lunch*" and "*having dinner*". If the user wants the plan to include the meals, u^{lunch} and u^{dinner} are set to ∞ and the duration of both actions are specified in SPP^u. Otherwise, the utility of these actions is set to 0. Moreover, we build a tuple of the form $\langle lunch, open_{lunch}, close_{lunch}, location_{lunch} \rangle$ (resp. for *dinner*), where $open_{lunch}, close_{lunch}$ (resp. *dinner*) are set to the typical start/end hours of meals in the city to visit and, for the sake of simplicity, we consider that the

	Activity	Priority	Opening hours	Duration	Utility (α = 0.5)
1	L'Oceanografic	230	10 - 24	4h	230
2	Valencia Port	195	8 - 24	1h 30'	195
3	L'Hemisferic	195	16 - 21	1h 15'	195
4	Lonja	189	10 - 15	1h	95
5	BioParc	180	10 - 21	2h 15'	90

Fig. 6. Activities in our planning problem.

locations of these activities coincide with the location of the last performed activity before lunch/dinner.

M is the set of movement actions, where each $M_{j,k} \in M$ represents the movement from the place of activity A_j to the place of activity A_k. Each movement action $M_{j,k}$ is described by its duration $dur^{M_{j,k}}$, which is computed by taking into account the distance between the places of activities A_j and A_k. $dur^{M_{j,k}}$ is computed by using the information described in section 4.2.

It is important to note that not all the activities in FRC^u will be likely included in the plan since the plan schedule will depend on the user available time, his temporal constraints and the time restrictions of the environment (i.e. opening hours of places). Therefore, the planning subsystem must select which activities to include in the plan as well as consider the initial user location and the distance between two consecutive actions to estimate the start time of the second activity.

In this paper, we introduce two different ways to tackle this problem. First, we formulate it as a *Constraint Satisfaction Problem* (CSP),[12] where each action $a \in A \cup M$ is associated to a variable in the problem and the constraints that establish the relationships between these variables are also defined. Second, we formulate this problem as a *Partial Satisfaction Planning*[23] (PSP) problem. In this case, we specify the problem by means of the Planning Domain Definition Language (PDDL) version 3.0[11] and use an existing planner to solve it.

6.2. *Formulation as a constraint satisfaction problem*

Given the user specific planning preferences, the set A of all the activities that can be performed and the set M of movement actions, we formulate the resulting planning problem as a *Constraint Satisfaction Problem* (CSP)[12] as follows. We define a set of variables $a_{i,j} \in \{0,1\}, \forall\, i,j \in [1,|A|]$ to denote whether activity A_i is performed in jth place in the plan or not. We post two constraints over these variables: $\sum_{\forall\, j} a_{i,j} \leq 1$, to prevent activity A_i from being performed twice in the plan and $\sum_{\forall\, i} a_{i,j} \leq 1$ to avoid performing two activities in the same plan position. In addition, we create a new variable $a_{0,0}$ to denote the action *user at initial location* is the first action to be executed. This variable is set equal to 1.

From the set of variables $a_{i,j}$ we can infer the necessary movements in the plan. In order to do so, a set of variables $m_{i,j,k}, \forall\, i,j,k \in [1,|A|]$ are introduced in the CSP. $m_{i,j,k}$ is set to 1 if activity A_i is performed in the kth position and A_j is performed in the $(k+1)$th position, that is $m_{i,j,k} = a_{i,k} * a_{j,k+1}$.

We also define $ts_i, te_i \in [\max(Ts, open_i), \min(Te, close_i)], \forall\, i \in [1,|A|]$ to denote the start and end time of activity A_i. The domain of these variables is determined by the opening hours of the places and the user's available time.

Following we specify some additional CSP constraints.

The values of the start and end time of each activity must be consistent with the activity duration:

$$te_i = ts_i + dur_i \;\; \forall\, i \in [1,|A|]\,.$$

For example, the constraint $te_1 = ts_1 + 4$ will set the values of the variables ts_1 and te_1, which represent the start and end time of activity 1, respectively.

The activities and movements in the plan must not overlap.

$$te_i + dur_{i,j} \leq ts_j + \left(1 - \sum_{\forall\, k \in [1,|A|]} m_{i,j,k}\right) * \Lambda \;\; \forall\, i,j \in [1,|A|]\,.$$

That is, if activity A_i is performed immediately before A_j (that is, $\sum_{\forall\, k} m_{i,j,k} = 1$), then the start time of A_j must be greater or equal than the end time of A_i plus the time to go from A_i to A_j ($dur_{i,j}$). Otherwise, we use a constant Λ which takes on a high enough value so as to satisfy the constraint. For example, the following constraint indicates that activities 1 and 2 cannot overlap, provided that it takes 15 minutes to move from *L'Oceanografic* to *Valencia Port*:

$$te_1 + 15 \leq ts_2 + \left(1 - \sum_{\forall\, k \in [1,6]} m_{1,2,k}\right) * \Lambda\,.$$

The total duration of activities and movements cannot exceed the available time of the user:

$$Te - Ts \geq \sum_{\forall\, i,j \in [1,|A|]} a_{i,j} * dur_i + \sum_{\forall\, i,j,k \in [1,|A|]} m_{i,j,k} * dur_{i,j}\,.$$

The system offers the user the choice of selecting the most preferable plan. Thus, we consider two optimization functions. The first one maximizes the utility of the whole plan:

$$Maximize\;\; U = \sum_{\forall\, i,j \in [1,|A|]} a_{i,j} * u_i - \sum_{\forall\, i,j,k \in [1,|A|]} m_{i,j,k} * dur_{i,j} * \beta\,.$$

In this case, the *utility* of the movement actions is treated as a penalty, because spending a lot of time in movements is not desirable; for this reason, in the second term of the above expression, the utility of the movement actions is computed as $-dur_{i,j} * \beta$, where $\beta \in \Re$ is a value to adjust that duration to the utility of the selected activities.

The second optimization function is aimed at maximizing the time spent in the preferred activities. This optimization function is defined as:

$$Maximize\;\; T = \sum_{\forall\, i,j \in [1,|A|]} a_{i,j} * dur_i * u_i - \sum_{\forall\, i,j,k \in [1,|A|]} m_{i,j,k} * dur_{i,j}\,.$$

Again, the duration of the movement actions is treated as a penalty.

In the tourist domain, the utilization of the U optimization function by the CSP usually returns plans that contain more activities than the plans computed with the T optimization function. This seems obvious as, in general, the more activities the more utility. In the current example, the plan consists of $a_0 \rightarrow a_4 \rightarrow a_6 \rightarrow a_2 \rightarrow a_3$, where a_0 representsthe user being in the initial location and a_6 represents the action

Fig. 7. Tourist agenda for John when using maximum utility.

of *having lunch*. The corresponding agenda is shown in Figure 7. On the other hand, the plan obtained when using the T optimization function contains fewer actions because in this case longer actions are preferable. For the current example, the obtained plan is $a_0 \rightarrow a_1$.

6.3. *Formulation as a partial satisfaction planning problem*

Given the user specific planning preferences, the set A of all the activities that can be performed and the set M of movement actions, we formulate this problem as a *Partial Satisfaction Planning* problem. First, we need some definitions.

Definition 6.1. Let $\mathcal{F}$ be a finite set of fluents and $\mathcal{A}$ be a finite set of actions, where each action a consists of a list of preconditions and a list of add and delete effects (denoted by $pre(a)$, $add(a)$ and $del(a)$, respectively). $\mathcal{I} \subseteq \mathcal{F}$ is the set of fluents describing the initial state and $\mathcal{G} \subseteq \mathcal{F}$ is the set of goals. Hence a **planning problem** is defined as a tuple $P = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$.

Definition 6.2. Given a set of fluents $\mathcal{S}$, an action $a \in \mathcal{A}$ is **applicable** in the state $\mathcal{S}$ if $pre(a) \subseteq \mathcal{S}$. The **result** of applying an action a to a state $\mathcal{S}$ is $Result(\mathcal{S}, a) = \mathcal{S} \cup add(a) - del(a)$. A sequence of actions $\Pi = \langle a_1, \ldots, a_n \rangle$ is a **solution plan** if $\mathcal{G} \subseteq Result(Result(\ldots Result(\mathcal{I}, a_1), \ldots, a_{n-1}), a_n)$.

It is important to remark that, in the previous definitions, $\mathcal{G}$ is considered as a conjunctive goal, that is, all the fluents in $\mathcal{G}$ must be satisfied in the state reached

after applying Π. However, in the particular context of a tourist agenda for a given user, as we explained above, not all the activities will be likely included in the plan since the plan schedule will depend on the user available time, his temporal constraints and the time restrictions of the environment (i.e. opening hours of places). Therefore, the planning subsystem must select which activities among all the activities in the set A to include in the plan. This type of problems are known as *Partial Satisfaction Planning*[23] (PSP) problems (also known as *over-subscripted planning problems*). Unlike classical planning problems, in PSP problems the solution plan is not required to achieve all the goals but instead achieve the *best* subset of goals given the resource limitations. More formally[3]:

Definition 6.3. Given a planning problem $P = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$ and, for each action $a \in \mathcal{A}$, a "cost" $c_a \geq 0$ and, for each goal specification $g \in \mathcal{G}$, a "utility" $u_g \geq 0$, the **Partial Satisfaction Planning problem** is defined as finding a finite sequence of actions $\Pi = \langle a_1, \ldots, a_n \rangle$ starting from $\mathcal{I}$ that leads to a state $\mathcal{S}$ maximizing the net benefit value $\sum_{g \in \mathcal{G}_\mathcal{S}} u_g - \sum_{a \in \Pi} c_a$, where $\mathcal{G}_\mathcal{S} \subseteq \mathcal{G}$ is the set of goals satisfied in $\mathcal{S}$.

As we said before, we identify two sets of actions in our problem: the set A, where each $A_j \in A$ is described by (dur^{A_j}, u^{A_j}); and the set M, where each $M_{j,k} \in M$ is described by $dur^{M_{j,k}}$. Unlike a CSP, in a PSP formulation the problem is modelled by means of *action schemata* (we use PDDL3[11] to describe this problem). An action schemata is a generic action that represents the type of action that can be performed in the domain; for example, 'move' from one place to another or 'perform' an activity. The action schematas are then particularized for the values of the problem, thus giving rise to a set of instantiated actions. For example, if we have the activities *"Visit L'Hemisferic"* and *"Visit La Lonja"* in our problem, we will have two different actions: `perform hemisferic` and `perform lonja`. The specific values of a problem are described in the initial state by means of predicates and functions. The predicates and functions for an activity are:

- The duration of an action a (and also its cost c_a) is defined by means of the function `(activity_duration` a`)`.
- An activity a is performed in a place p; this relation is established by means of the predicate (`takes_place` a p`)`.
- A place has an opening hour and a closing hour that are specified by two functions: `(opening_hour` p`)` and `(closing_hour` p`)`.

For example, the corresponding functions and predicates for the activity *L'Hemisferic*, that are represented in the initial state, are:

```
(= (activity_duration hemisferic) 1.25)
(takes_place hemisferic pl_hemis)
(= (opening_hour pl_hemis) 16)
(= (closing_hour pl_hemis) 21)
```

On the other hand, the duration of moving from one location p_j to another location p_k is defined by means of the function `(move_duration` p_j p_k`)`. For example, the duration of the action to move from *L'Hemisferic* to *La Lonja* is established in the initial state as `(= (move_duration pl_hemis pl_lonja) 0.5)`. We also need the following predicates and functions:

- a predicate to represent the initial user location, `(person_at` p`)`: this predicate will be modified when a movement action is performed
- a function to represent the user available time, `(available_time)`: the initial value of this function is the user available time and this value will be decreased when an activity is performed
- a function to represent the current time, `(current_time)`: the initial value of this function is the start hour of the user time slot and this value will be increased when an activity is performed
- a function to represent the end time, `(end_time)`: the initial value of this function is the finish hour of the user time slot, and this value never changes
- a function to represent the lunch (resp. dinner) duration, `(lunch_duration)`; the start and the end time of the lunch (resp. dinner), `(lunch_start)` and `(lunch_end)`.

In our scenario, the user is initially at the Astoria Hotel and his available time slot is from 12pm to 6pm The lunch will take 1h30' between 1pm and 3pm. This information is specified in the initial state as follows:

```
(person_at pl_hotel)
(= (current_time) 12)
(= (end_time) 18)
(= (available_time) 6)
(= (start_lunch) 13)
(= (end_lunch) 15)
(= (lunch_duration) 1.5)
```

The action to perform an activity is defined as follows:

```
(:durative-action perform
  :parameters (?a - activity ?w - place)
  :duration (= ?duration (activity_duration ?a))
  :cost (activity_duration ?a)
  :condition (and (over all (takes_place ?a ?w))
     (over all (person_at ?w))
     (at start (not (performed ?a)))
     (at start (>= (current_time) (opening_hour ?w)))
     (at start (>= (closing_hour ?w) (+ (current_time) (activity_duration ?a))))
     (at start (> (available_time) (activity_duration ?a))))
  :effect (and (at end (performed ?a))
     (at start (increase (current_time) (activity_duration ?a)))
     (at start (decrease (available_time) (activity_duration ?a))))
)
```

The action `perform` takes as parameters the activity to perform `?a` and the corresponding place `?w`. Both the duration and the cost of the action are established by the activity duration. The preconditions for this action to be applicable are: (1) the activity happens in the place indicated by the parameter `?w`; (2) the user is at this location; (3) the activity has not been performed yet; (4) the current time is greater than the opening hour of the place; (5) the activity will be finished before the closing hour of the place and (6) the available user time is greater than the activity duration. The effects of the action assert that the activity is done, and that the current time and the user available time are modified according to the activity duration.

The action to perform the activities of "*having lunch*" or "*having dinner*" are similarly defined to the action `perform`. Finally, the action to move from one location to another is defined as follows:

```
(:durative-action move
  :parameters (?w1 - place ?w2 - place)
  :duration (= ?duration (move_duration ?w1 ?w2))
  :cost (move_duration ?w1 ?w2)
  :condition (and (at start (person_at ?w1))
     (at start (> (available_time) (move_duration ?w1 ?w2))))
  :effect (and (at end (person_at ?w2))
     (at start (not (person_at ?w1)))
     (at start (increase (current_time) (move_duration ?w1 ?w2))
     (at start (decrease (available_time) (move_duration ?w1 ?w2))))
)
```

The action `move` takes as parameters the initial place `?w1` and the destination `?w2`. Again, both the duration and the cost of the action are defined in terms of the movement duration. The preconditions for this action to be applicable are: (1) the user is at location `?w1` and (2) the available user time is greater than the movement duration. The effects of the action assert that the user is at the new location at the end of the action, and that the current time and the user available time are modified according to the movement duration.

Finally, each goal in $\mathcal{G}$ denotes the completion of an activity a and this is represented by means of the predicate `(performed` a`)`. We have two different types of goals in $\mathcal{G}$:

- *Hard goals* represent the realization of an activity that the user has specified as mandatory, for example, "having lunch". In this case, no utility is defined: `(performed havinglunch)`.
- *Soft goals* represent the realization of a tourist visit, for example, visiting *L'Hemisferic*. The utility of a soft goal g is defined as $u_g = u^{A_j}$; for example: `((performed hemisferic) soft 195)`, where 195 is the goal utility.

From our experience with both formulations, CSP and PSP, we can conclude PSP is better suited for this type of problems. CSP is a general framework for solving any type of constraint-based problem, by finding the variables values that

satisfy the conditions imposed by the constraints. A more natural and human-oriented approach of solving a tourist agenda is to use a planning framework for the problem definition and the problem solving, like a PSP formulation, which provides a great flexibility and expressivity to tackle this type of problems.

We used the SAPA planner[3] in our tourist agenda performance tests. For the particular problem instance we have presented in this section, both the CSP and PSP obtained the same solution plan but the PSP performance was much more efficient. The reason behind this efficiency is that, although SAPA is a domain-independent planner, it makes use of planning heuristics to guide search through the causal relationships between actions. In contrast, in a CSP formulation we cannot define such a causal structure with variables, and so only use generic heuristics for assigning values to variables can be used.

6.4. *Tourist agenda*

When the system solves the problem either using a CSP or a PSP formulation, we obtain a plan which contains a subset of the activities in FRC^u joint with the time when such activities should start and finish. This plan is shown as an agenda of activities. A **plan** is defined as a tuple of the form: $\langle date, (a, ts_a, te_a)^* \rangle$, where $date$ is the date of the visit, a refers to the scheduled activity, and ts_a and te_a are the start and end time of that activity, respectively.

7. Conclusions and Further Work

Nowadays there exists an increasing interest on tourism recommender systems as more and more people use travel web services to obtain information for their trips. However, most of the existing services are simply aimed to provide specific travel items to the user; the generation of personalized tourism tours require, among other things, the incorporation of planning capabilities to properly combine and relate the travel items.

e-Tourism is a web service that generates recommendations about personalized tourist tours in the city of Valencia (Spain). It is intended to be a service for foreigners and locals to become deeply familiar with the city and plan leisure activities. *e-Tourism* makes recommendations based on the user's tastes, his demographic classification, the places visited by the user in former trips and, finally, his current visit preferences. The tool shows the user an agenda of recommended activities which reflect the user's tastes and takes into account the geographical distance between places or the opening hours of such places.

We also plan to incorporate new hybrid techniques and good metrics in the GRSK to measure the effectiveness of recommendations. Moreover, we are interested in group recommendation as people usually travel on group trips. This introduces a new problematic as now recommendations must adapt the preferences of the majority of users or be in accordance with the common likes of all users.

On the other hand, we are also working on some improvements in the planning process. Our objective is to incorporate the preferences of the user about how he would like his agenda to be organized in the planning process. For example, some people prefer visiting museums in the morning and other people prefer to have a relaxed visit with no many activities scheduled in the same day. The user preferences are extracted from his feedback by analyzing the activities he ended up doing in his former visits, their duration, the schedule of such activities, the number of visits in a day or the elapsed time between one visit and the next one.

Acknowledgments

This work has been partially funded by Consolider Ingenio 2010 CSD2007-00022 project, by the Spanish Government MICINN TIN2008-6701-C03-01 project and by the Valencian Government GVPRE/2008/384 project. We thank J. Benton for having provided us with the system Sapa to execute our experiments.

References

1. Adomavicius G., Tuzhilin A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 2005.
2. Ardissono L., Goy A., Petrone G., Segnan M., Torasso P. INTRIGUE: personalized recommendation of tourist attractions for desktop and handset devices. *Applied AI, Special Issue on Artificial Intelligence for Cultural Heritage and Digital Libraries*, 17(8-9), 2003.
3. Benton, J., Do, Minh B., Kambhampati, S. Over-subscription planning with numeric goals. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.
4. Burke R. *The Adaptive Web*, chapter Hybrid web recommender systems. Springer Berlin/Heidelberg, 2007.
5. Cheverst K., Mitchell K., Davies N. The role of adaptive hypermedia within a context-aware tourist guide. *Communications of the ACM*, 2002.
6. Delgado J., Davidson R. Knowledge bases and user profiling in travel and hospitality recommender systems. In *Proc. of the ENTER*, 2002.
7. Felfernig A., Gordea S., Jannach D., Teppan E., Zanker M. A short survey of recommendation technologies in travel and tourism. *OEGAI Journal*, 25(7), 2007.
8. Fesenmaier D.R., Ricci F., Schaumlechner E., Wober K., Zanella C. Dietorecs: Travel advisory for multiple decision styles. In *Proc. of ENTER*, 2003.
9. Fink J., Kobsa A. User modeling for personalized city tours. *Artificial Intelligence Review*, 18(1), 2002.
10. Garcia, I., Sebastia, L., Onaindia, E., Guzman, C. Using a generalist recommender system for a tourist planning application. Technical Report TR-GRPS-AI-1, Universidad Politecnica Valencia, 2008.
11. Gerevini, A., Long, D. Plan constraints and preferences in pddl3: The language of the fifth international planning competition. Technical report, University of Brescia, 2005.
12. Ghallab M., Nau, D., Traverso P. *Automated Planning. Theory and Practice*. Morgan Kaufmann, 2004.

13. Goy A., Magro D. Dynamic configuration of a personalized tourist agenda. In *Proc of the ICWI*, 2004.
14. Hanani U., Shapira B., Shoval P. Information filtering: Overview of issues, research and systems. *User Modeling and User-Adapted Interaction*, 11, 2001.
15. Jannach D., Zanker M., Jessenitschnig M., Seidler O. *Information and Communication Technologies in Tourism 2007*, chapter Developing a conversational travel advisor with Advisor Suite. Springer, 2007.
16. Pazzani M.J. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13, 1999.
17. Pazzani M.J., Billsus D. *The Adaptive Web*, chapter Content-based recommendation systems. Springer Berlin/Heidelberg, 2007.
18. Poslad S., Laamanen H., Malaka R., Nick A., Buckle P., Zipf A. CRUMPET: Creation of userfriendly mobile services personalised for tourism. In *Second International Conference on 3G Mobile Communication Technologies*, 2001.
19. Resnick P., Varian H. Recommender systems. *Communications of the ACM*, 40(3), 1997.
20. Ricci F., Arslan B., Mirzadeh N., Venturini A. ITR: A case-based travel advisory system. In *Proc. of the ECCBR, LNCS*, volume 2416, 2002.
21. Ricci F., Cavada D., Nguyen Q.N. Integrating travel planning and on-tour support in a case-based recommender system. In *Proc. of the Workshop on Mobile Tourism Systems (with Mobile HCI)*, 2002.
22. Schafer J.B., Konstan J.A, Riedl J. Recommender systems in e-commerce. In *Proceedings of the First ACM Conference on Electronic Commerce (EC'99)*, 1999.
23. Smith. D. E. Choosing objectives in over-subscription planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 2004.
24. Towle B., Quinn C. Knowledge based recommender systems using explicit user models. Technical report, Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, AAAI Technical Report WS-00-04, 2000.
25. Werthner H. Intelligent systems in travel and tourism. In *Proc. of the IJCAI*, 2003.