# Adaptive Recommender Systems for Travel Planning

Tariq Mahmood[a], Francesco Ricci[b], Adriano Venturini[c], and Wolfram Höpken[d]

[a] Department of Information and Communication Technology
University of Trento, Italy
tariq@itc.it

[b] Faculty of Computer Science
Free University of Bozen-Bolzano, Italy
fricci@unibz.it

[c] ECTRL Solutions
Trento, Italy
venturini@ectrlsolutions.com

[d] Etourism Competence Center Austria
Innsbruck, Austria
wolfram.hoepken@etourism-austria.at

## Abstract

Conversational recommender systems have been introduced in Travel and Tourism applications in order to support interactive dialogues which assist users in acquiring their goals, e.g., travel planning in a dynamic packaging system. Notwithstanding this increased interactivity, these systems employ an interaction strategy that is specified *a priori* (at design time) and is followed quite rigidly during the interaction. In this paper we illustrate a new type of conversational recommender system which uses Reinforcement Learning techniques in order to autonomously learn an adaptive interaction strategy. After a successful validation in an off-line experiment (with simulated users), the approach is now applied within an online recommender system which is supported by the Austrian Tourism portal (Austria.info). In this paper, we present the methodology behind the adaptive conversational recommender system and a summarization of the most important issues which have been addressed in order to validate the approach in an online context with real users.

**Keywords:** Conversational Recommender Systems, Reinforcement Learning, Markov Decision Process, Travel Planning, Information Presentation and Delivery.

## 1 Introduction

Recommender systems are intelligent E-commerce applications that assist users in their information-seeking tasks by offering personalized product recommendations during an interaction session (Adomavicius and Tuzhilin, 2005). Travel recommender systems are aimed at supporting the critical travel planning decisions that the traveler will face before travel or while on-the-move (Fesenmaier et al., 2006). These systems acquire the user needs and wants, either explicitly (by asking) or implicitly (by mining the user online activity), and suggest destinations to visit, points of interest, events or activities. The main objective of a travel recommender system is to ease the information search process of the traveler and to convince (persuade) her of the appropriateness of the proposed services. In recent years, a number of travel recommender systems have been designed and some of them are now operational in major tourism portals (Venturini & Ricci, 2006).

Traditional recommenders support quite simple and non-interactive search processes: at each stage of the traveler's interaction session, they only execute one type of action, i.e., decide which product(s) to recommend to the user. In order to support more real (natural) and interactive processes, *conversational* recommender systems (Bridge et al., 2006) have been recently proposed. These systems support a dialogue where, at each stage, the system can select one from amongst a set of available system actions, e.g., recommend some product or ask the user for some information. The particular action selected by the system is specified by its *recommendation strategy*. For instance, imagine that a conversational system is queried for hotels that would suit the user preferences. It could employ the following two strategies (among many others): 1) *ask the user in detail about her preferences, and use this information to extract a small product subset*, or 2) *propose a set of products to the user, and exploit the user feedback in order to refine future recommendations*. In fact these two conversational strategies were initially included in the DieToRecs system (Fesenmaier et al., 2003) and are now part of the Trip@dvice technology (Ricci et al., 2006). Conversational systems typically employ a rigid strategy, i.e., one which is determined at design time and hard-coded into the system. For instance, Trip@dvice allows the user to select one of these two strategies, but then she must rigidly follow the steps prescribed by the chosen strategy. A major limitation of this approach is that there could be a large number of conversational but rigid strategies for a given recommendation task, such as travel planning with multiple service types. Furthermore, the process of selecting a strategy is based on intuition and on the previous experience of the system designers, which by no means guarantees that the chosen strategy would really suit the users. Hence, the designers have to evaluate several strategies in order to discover an (almost) *optimal* one. This is an infeasible process, considering the necessary expense in terms of budget, effort, time etc. These limitations necessitate that conversational systems should be capable of determining *by themselves* the best strategy for assisting the users.

We tackle these requirements proposing a new type of Case-Based conversational recommender system that, rather than following a rigid interaction design, offers a range of information and decision support functions and, during the interaction, is able to improve an initial strategy and adopt an optimal one (Mahmood and Ricci, 2007a). The optimal strategy basically maximizes a reward function that models how much benefit the user gets from each interaction step.

In (Mahmood and Ricci, 2007a), we presented the more technical details of such a technology and we described the results of an evaluation, based on human/computer simulation. In that evaluation, when a search for a travel service (e.g. a hotel) returns too many options, the system should decide whether to ask the user for additional service features in order to retrieve a smaller set of services or to let the user autonomously take this decision and manually change the query. Our results show that the recommender system can improve an initial strategy that, in a rigid way, asks the user for additional features only when the result set size is greater than a given threshold. In fact, we show that different optimal strategies are learnt, depending on the system estimate of the cost of each interaction step (reward), i.e., the dissatisfaction of the user in not having, yet, acquired her goal.

Although these results are encouraging, they were obtained under a limited setting, i.e., there is only one situation in which the system must learn to make a decision, and the system's action set comprises only two actions. Moreover, we have carried out this evaluation *off-line*, i.e., in simulated interactions. In order to address these limitations, we have applied our approach within the Travel Planner tool (TP) that we built for the Austrian tourism web portal (Austria.info), in the context of the etPackaging project funded by Austrian Network for E-Tourism (ANET). This paper describes the general model of our adaptive recommender system and its application to the TP tool. Moreover, it describes the changes made to a previously designed recommender (conversational but not adaptive) and in particular, the interaction points where the new system is designed to be adaptive. The paper also describes the evaluation plan that we designed in order to further validate our approach with real users. Hence,, the major contribution of this paper is a generally-applicable model for a truly adaptive travel recommender system and its case study application to the Austria.info portal.

The paper is organized as follow. Section 2 describes our adaptive model. Then, section 3 illustrates our proposal for incorporating our adaptive approach within TP. Further on, section 4 illustrates our proposed system architecture for learning the optimal policy, along with a brief description of the on-line evaluation methodology. Finally, section 5 discusses our future work.

## 2 Learning Personalized Interaction Strategies for Conversational Recommenders

Our proposed recommendation model, shown in Figure 1, basically comprises two entities, namely the *Information System* (*IS*) and the *Recommendation Agent* (*RA*). The *IS* is the non-adaptive entity whose function is entirely controlled by the user, and serves simple user requests like displaying a query result, showing most popular selections etc. The *RA* is the adaptive entity which actively assists the user by providing her with the relevant information at appropriate stages of the interaction session. This process of assistance is dictated by the system's optimal recommendation strategy.

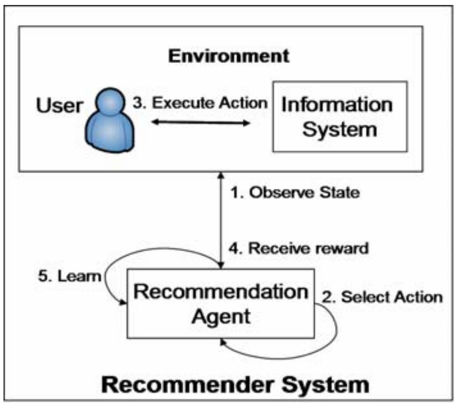

**Fig. 1.** General recommendation model

In our model, we learn the optimal strategy by using techniques from Reinforcement Learning (RL), which is based on the mathematical framework of Markov Decision Processes (MDP) (Sutton & Barto, 1998). In the context of RL, we shall use the term *optimal policy* rather than optimal strategy, where a policy simply specifies how the strategy is implemented in terms of the system's action selection rules. In order to learn the optimal policy through RL, we consider the *RA* as a decision-making Agent which must achieve a goal, i.e., learn the optimal policy. It does so by executing some trial-and-error interactions with the user, who is part of the Agent's Environment. Specifically, at each stage, and after the user has taken some action, the Agent observes the current situation, or *state*, of the system (step 1 in Figure 1). The observed system state could include information about the current state of the user (e.g., her goals and preferences), the on-going interaction (e.g., how many products the user has already viewed), or even the state of the Agent itself (e.g., if it has already applied a recommendation action). This is the information which the Agent needs in order to learn the optimal policy.

In one or more system states, multiple system actions could be available to the Agent for execution (steps of the conversation). For instance, as we mentioned in Section 1, there could be situations when the system is uncertain whether it is better to push a hotel recommendation or still propose some options. We label these states as the *System Decision Points* (SDPs). In the SDPs, the Agent tries out (selects) the different system actions during the interaction (Step 2 in Figure 1). For each action execution (Step 3) (and the corresponding user response), the Agent receives a numerical reward (reinforcement signal) from the Environment (Step 4), informing it whether its previous action was acceptable for the user or not. The magnitude and the sign of this reward is determined by the level of user acceptability, e.g., a large positive reward (+10) could be assigned for most favorable actions, e.g., a suggestion accepted by the user, and a large negative reward could be assigned for least favorable ones (-10), e.g., the user quits. As the interaction proceeds, the Agent exploits the received rewards in order to learn to avoid the unacceptable actions and to take acceptable ones (Step 5). This is performed in the *policy iteration* algorithm where basically the action that maximizes the immediate reward plus the expected discounted value of the next state defines the new optimal action in that state (Sutton and Barto, 1998). In order to compute the expected discounted value of the next states the state-to-state transition probabilities learned in the interactions are used. The RL policy iteration procedure guarantees that this process shall eventually lead the Agent to learn the optimal action at each SDP, i.e., learn the optimal policy. We note that in the *non-SDP* states, only a single action is available for execution, which we consider as the optimal action.

## 3 Adaptive Travel Planning Recommendations

In this section, we shall illustrate our concept for incorporating the adaptive recommendation approach within the *TP* tool. The Travel Planning tool is supported by the Trip@dvice technology. It enables a user to select travel components from catalogues (hotels, attractions, events) and bundle them in a new plan, or select an existing travel plan and customize it according to specific needs (delete or add travel components). As stated previously, our approach is based on learning the optimal action at one or more SDPs which the Agent could encounter as the user navigates through the system during her interaction session.

The basic model of the users' navigation flow is show in Figure 2. Here, a *view state* represents any particular web page which the user could view during her session (e.g. a query input form). In each view state, the user could take one or more *user actions*, which lead the navigation into a system state (e.g., submit a query or go back to a previous interaction state).

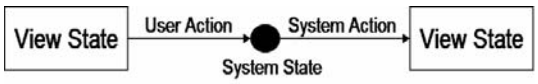

**Fig. 2.** User navigation model

The system state could be either a SDP or a non-SDP state, and is hence used for selecting the next *system action*, which (when executed) results in a new web page being shown to the user. For instance, suppose that in the query input view state, the user submits a query. Then, the ensuing system state is a SDP because here, the system must decide how to continue the conversation and what information to present in the next view state, e.g., ask for more preferences or execute the query.

Currently, we are employing the Trip@dvice CBR methodology (Ricci et al., 2006) in order to compute the relevant product recommendations for the users (i.e., rank higher products selected in similar sessions). It offers two product search functions: 1) Seeking for Inspirations (*SeekInsp*), where the user is prompted with a list of product proposals (in a loop) until she selects some product, and 2) Query Search (*QrySrch*), where the user constrains a logical query to a product catalogue in order to retrieve some product. The rigid conversational policy of Trip@dvice dictates the current non-adaptive navigation flow of the user through the *TP*. The flow comprises a single SDP which depicts the situation where the user has submitted a query to a product catalogue, and the system has retrieved a large number of products. Trip@dvice's policy specifies to always suggest additional features to the user when the result set size is greater than a certain threshold. In order to learn a better (optimal) policy we have proposed three additional SDPs to be incorporated within the non-adaptive flow, along with a more detailed system action set. Due to space limitations, we only show the navigation flow at the four SDPs (See Figure 3 and Figure 4) instead of the complete adaptive navigation flow (Mahmood et al., 2007c).

We will now describe the flow in Figure 3. The user enters the system through a *Welcome Page*. In this page, if she selects the *QrySrch* function (User Action *SlctQrySrch*), the system enters in *SDP1*, where it should decide whether to show the different search categories (destinations, events etc.) to the user in the view state *ViewSrchOpts* (system action *ShowOpts*) or to ask her initially for some travel characteristics (system action AskChars) before showing her the various service categories. Specifically, the user enters her characteristics in the view state *SpecChars* (User Action *EnterChars*). The system then enters in a non-SDP system state (black circle labeled 1), from where the only possible system action is to execute *ShowOpts*. Furthermore, if the user selects the *SeekInsp* function (User Action *SlctSeekInsp*), the system enters in *SDP2* where it should decide whether to show her the list of product proposals in the view state *ViewProposals* (system action *ShowProps*) or to initially ask her for some travel characteristics (system action *AskChars*) before showing the complete travel plan proposals. Here, once the user has entered her characteristics in *SpecChars*, the system enters a non-SDP state (black circle labeled 2), in which the only possible system action is *ShowProps*.

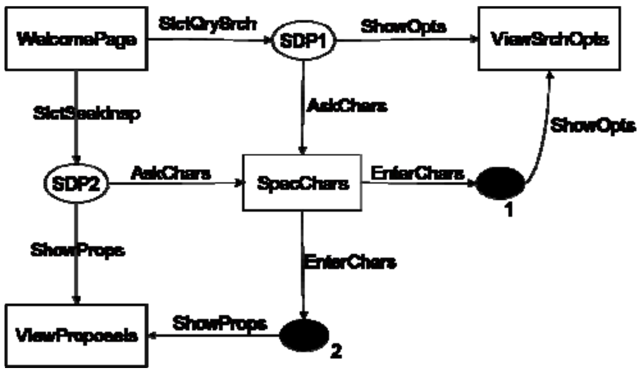

**Fig. 3.** Decision points after '*Query Search'* or '*Seeking for Inspiration*' user requests

We will now describe the flow in Figure 4. If the user constrains and submits a specific search query to a catalogue (e.g. hotels), the system encounters *SDP3* (Figure 4). If the query's result set size is greater than some threshold, the system should decide whether to suggest product features for reducing (or tightening) this result set in the view state *ViewTightFtrs* (system action *SuggTight*) or to execute the query and show all the retrievals (system action *Exec*). On the other hand, if the result set size is empty, i.e., no hotel satisfies the query conditions, the system should decide whether to suggest the user to relax some of the current query constraints in the View State *ViewRelaxFtrs* (system action *SuggRelax*) or to automatically relax one constraint (selected by the system and not involving the user) from the query (system action *autoRelax*) in order to produce a result set.

Finally, in the situation where a list of products is ready to be shown to the user, the system encounters *SDP4* (Figure 4), where it must decide to show one amongst four possible result pages:

- *ShowRP1*, i.e., show the page *ViewRP1*, in which the system simply shows the recommended products result list,
- *ShowRP2*, i.e., show the page *ViewRP2*, in which the system, besides showing the products result list, also requests the user to enter some (more) travel characteristics (in order to compute better recommendations),

- *ShowRP3*, i.e., show the page *ViewRP3*, in which the system, besides showing the products result list, also pushes the user to make some related searches on other types of products (cross selling),
- *ShowRP*, i.e., show the page *ViewRP4*, in which the system, besides showing the products result list, also pushes the user to add the top-ranked product to her shopping basket and to make some related searches on other product types.

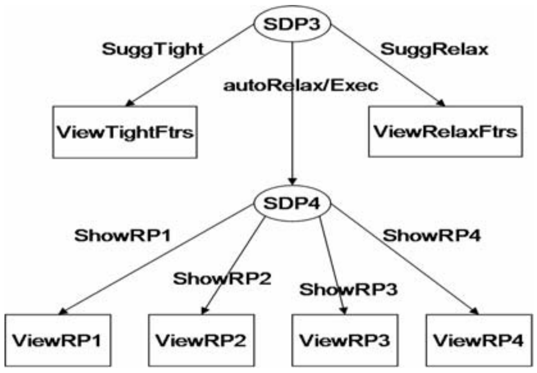

**Fig. 4.** Decision points after the user requests a query execution

In the complete adaptive navigation flow, we modeled a total of twenty view states and four system decision points. We have also implemented this flow as a Petri Net (Reisig & Rozenberg, 1998), which is a graphical formalism for systems specification. In the Petri Net terminology, we model the *view state*, *user action*, *system state*, and the *system action* as the *place*, *input place*, *transition*, and *output place* respectively. This representation is useful to track the navigation streams of the users (generated during the evaluation of our approach) and acquire diverse information about them (the general navigation model of the users, the unique navigation paths adopted by the users, most frequent paths) through graph mining tools such as the ProM framework (Dongen et al., 2005).

## 4 System Design and Evaluation

In this section, we shall illustrate the system architecture that we designed and built in order to incorporate our approach within the *TP* tool (Mahmood et al., 2007d) and the evaluation strategy required to test the proposed model. The architecture, according to the proposed recommendation model (Figure 1), consists of two main entities, namely the *Recommendation Agent*, which contains the functionality for learning the optimal policy, and the *Environment*, which represents the user and the Information System (see Figure 1), and contains the API which allows the Recommendation Agent to interact with objects that are *outside* it’s domain, in order to acquire the data necessary for learning the optimal policy, e.g., the user action taken at some interaction stage, the system state hence reached, etc. Let us now consider the Environment and the Recommendation Agent entities in detail.

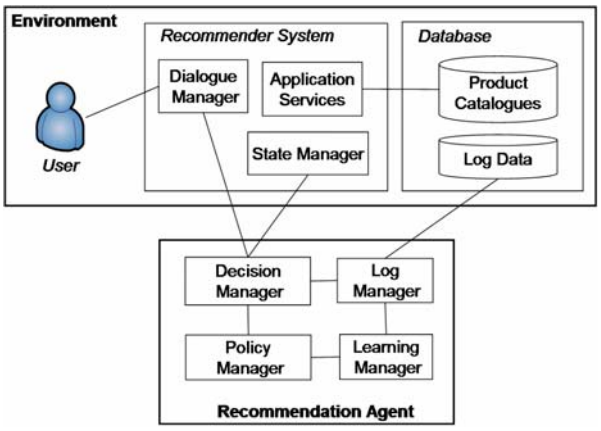

**Fig. 5.** System architecture

Within the Environment the *user* makes her requests to the *Recommender System* in order to satisfy one or more navigation/information goals. These requests are passed to the *Dialogue Manager* component of the system, which handles these requests and interacts with the *Application Services* component according to the system business logic (exploiting data from the *Product Catalogues*). The *State Manager* computes the system state after each user request, and asks the Recommendation Agent for the applicable system action in every SDP. Moreover it notifies the Recommendation Agent of state changes so that the Agent can acquire and store (in the *Log Data* component), at each stage of the user's session, the user action taken, the information related to the ensuing system state, the system action executed in this state, and the resulting view state (web page), which is produced by the Dialogue Manager. Within the Recommendation Agent, the *Decision Manager* is the interface to the Information System providing the requested system action to be executed in a SDP. It interacts with the *Policy Manager* component, which stores the current (optimal) policy of the Agent. Furthermore, the Decision Manager also logs all the relevant data at each stage (*Log Manager*). This data is exploited by the *Learning Manager* in order to learn the optimal policy.

This innovative approach to travel recommendation requires a specific and new evaluation methodology. We designed an approach based on the development of two alternative variants of the travel recommender system: a non-adaptive and an adaptive version. The first variant basically follows a set of default (rigid) interaction policies, offering to the user the designed views with a hard-coded logic. The second variant follows an interaction policy that is the result of the learning procedures. We want to test some hypotheses about the performance of these variants based on a set of dependent measures. Our hypotheses are based on the user's search and product choice behaviour, and their satisfaction with the behaviour (policy) of the system. In the evaluation, we must select a group of participants (meeting certain requirements) which will be randomly assigned to two experimental groups, and each group is asked to evaluate one of the variants of our recommender system. The evaluation proceeds in two phases. In a *training phase*, conducted initially with the first experimental group, the recommender variant employs a set of alternative hand-crafted policies. This variant forms our baseline system. In this phase, the Log Manager logs all the relevant sequential data (user action, system state information, system action, view state), which are then exploited by the Learning Manager component in order to learn the optimal policy for the adaptive flow of the *TP*. Then, in a *Testing phase*, conducted after the training phase, the second experimental group evaluates the recommender variant that employs the optimal policy.

During each phase, participants are asked to perform some tasks in the general context of '*planning a vacation in the region of Tirol, Austria*'. A series of performance measures must be recorded, some of them automatically, during the interaction (through the Log Data component), and some by asking the user to fill out a questionnaire after each interaction session. The aim of the evaluation is to test our hypotheses by determining the improvement in these measures from the training phase to the testing phase. Important measures of success are the total reward acquired by the recommendation agent (using a particular policy) and the rate of success of the system. Acting in this way, during the system evaluation we can discover the state variables that are more useful in finding an optimal policy. In fact, as the result of a preliminary simulation experiment has shown, the learned policy depends on the user and session model (Mahmood and Ricci, 2007b). In other words, the best interaction policy is influenced by the knowledge of the user, captured during the human-computer interaction and then used by the recommendation agent.

## 5 Conclusions and Future Work

In this paper, we have addressed some important issues related to the design of a radically new concept of recommender system for travel and tourism applications. To our knowledge, our approach is the first dedicated to learning interaction strategies for conversational systems, and this paper reports on the first concrete application of such an approach in a real on-line recommender system. The proposed model has been validated in a series of off-line experiments based on real-user interaction data acquired with the NutKing Recommender system (Ricci et al., 2006). For lack of space we refer the reader to (Mahmood and Ricci, 2007a; Mahmood and Ricci, 2007b) for details about the results of such experiments.

Currently (November 2007), we have completed the system implementation phase, the off-line evaluation (Mahmood and Ricci, 2007a; Mahmood and Ricci, 2007b), the usability evaluation, and we are now going to perform the on-line system validation as described above with a travel planner system that we developed for Austria.info. The proposed model still requires further analysis for a number of reasons. First of all, as we mentioned above, a critical aspect of our approach is the modelling of user-related information (demography, user goal, etc.) that should be included in the system state representation. This information is either acquired explicitly (asking the user) or implicitly (mining the interaction log). The exploitation of this implicit method is mandatory, as users usually refrain from replying to system questions. But, considering that online user behaviour is typically unpredictable and erroneous the quality of such implicit information could be questionable. For this reason we want to study reliable methods for determining which information is relevant for inclusion in the system state. This would extend the results presented in (Mahmood and Ricci, 2007b), based on simulations. Moreover, we need better methods to estimate the numerical reward values that should be assigned by the Recommendation Agent to a new state as effect of the user reply. These values should be as close as possible to the user satisfaction, and this again is typically inferred by observing the user activity. Notwithstanding these difficulties, the results of the off-line evaluation confirm that system policy can be improved using this model and this methodology has the potential to ease and speed the development of successful conversational travel recommender systems.

## References

Adomavicius, G. and Tuzhilin, A. (2005). 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):734–749.

Bridge, D., Goker, M., McGinty, L., and Smyth, B. (2006). Case-based recommender systems. The Knowledge Engineering Review, 20(3):315– 320.

Dongen, B. F. V., de Medeiros, A. K. A., Verbeek, H. M. W., Weijters, A. J. M. M., and van der Aalst, W. M. P. (2005). The prom framework: A new era in process mining tool support. In Ciardo, G., and Darondeau, P., editors, Application and Theory of Petri Nets 2005, pages 444-454, Berlin, Springer-Verlag.

Fesenmaier, D. R., Ricci, F., Schaumlechner, E., Woeber, K., and Zanella, C. (2003). DIETORECS: Travel advisory for multiple decision styles. In Frew, A. J., Hitz, M., and O'Connors, P., editors, Information and Communication Technologies in Tourism 2003, pages 232–241, New York, Springer.

Fesenmaier, D. R.,Werthner, H., and Woeber, K. (2006). Destination Recommendation Systems: Behavioural Foundations and Applications. Oxford, CABI Publishing.

Mahmood, T. and Ricci, F. (2007a). Learning and adaptivity in interactive recommender systems. In Dellarocas, C., and Dignum, F., editors, Proceedings of the ICEC'07 Conference, Minneapolis, USA, pages 75-84. New York, ACM Press.

Mahmood, T. and Ricci, F. (2007b). Towards learning user-adaptive state models in a conversational recommender system. In Brunkhorst, I., Krause, D., and Sitou, W., editors, Proceedings of the 15th Workshop on Adaptivity and User Modeling in Interactive Systems, ABIS'07, pages 373-378, Halle, Germany, Martin Luther University Press.

Mahmood, T., Cavada, D., Ricci, F., and Venturini, A. (2007c). Search and recommendation functionality. Technical Report D5.1, eTourism Competence Center Austria, Technikerstr. 21a, ICT-Technologiepark, 6020 Innsbruck.

Mahmood, T., Ricci, F., Venturini, A., and Cavada, D. (2007d). System architecture and design of the experiments. Technical Report D5.2, eTourism Competence Center Austria, Technikerstr. 21a, ICT-Technologiepark, 6020 Innsbruck.

Ricci, F., Cavada, D., Mirzadeh, N., and Venturini, A. (2006). Case-based travel recommendations. In Fesenmaier, D. R., Woeber, K. W., and Werthner, H., editors, Destination Recommendation Systems: Behavioural Foundations and Applications, pages 67-93, Oxford, CAB Publishing.

Reisig, W. and Rozenberg, G., editors (1998). Lectures on petri nets I: Basic models, advances in petri nets. New York, Springer.

Sutton, R. S. and Barto, A. G. (1998). Reinforcement Learning: An Introduction. London, UK, MIT Press.

Venturini, A. and Ricci, F. (2006). Applying trip@dvice recommendation technology to www.visiteurope.com. In Brewka, G., Coradeschi, S., Perini, A., and Traverso, P., editors, Proceedings of the 17th European Conference on Artificial Intelligence, pages 607-611, Amsterdam, IOS Press.

## Acknowledgements

We are grateful to Österreich Werbung, and in particular to M. Gratzer, N. Kellner and M. Shobert, for their collaboration and support during the project.