Recommending Improved Configurations for Complex Objects with an Application in Travel Planning

Amihai Savir
Department of Computer Science
Ben-Gurion University of the Negev
asavir@cs.bgu.ac.il

Ronen I. Brafman
Department of Computer Science
Ben-Gurion University of the Negev
brafman@cs.bgu.ac.il

Guy Shani
Information Systems Engineering
Ben-Gurion University of the Negev
shanigu@bgu.ac.il

Abstract

In many applications a user attempts to configure a complex object with many possible internal choices. Recommendation engines that automatically configure such objects given user preferences and constraints, may provide much value in such cases. These applications offer the user various methods to provide the input and generate appropriate recommendations. It is likely, though, that the user will not be able to fully express her preferences and constraints, requiring a phase of manual tuning of the recommended configuration. We suggest that following this manual revision, additional constraints and preferences can be *automatically* collected, and the recommended configuration can be *automatically* improved. Specifically, we suggest a recommender component that takes as input an initial manual configuration of a complex object, deduces certain user preferences and constraints from this configuration, and constructs an alternative configuration. We show an appealing application for our method in complex trip planning, and demonstrate its usability in a user study.

Introduction

The configuration of complex objects is known to be a difficult task (Magro and Torasso 2000; Soininen et al. 2000; Aldanondo, Moynard, and Hamou 2000). Consider the task of purchasing a computer. Such a computer is constructed from a CPU, RAM, hard disk, and many more internal items. The user must select appropriate components to meet her needs. An even more complex problem is that of constructing a trip to a country, where the goal of such a trip is to visit a set of attractions located in different parts of the country. A configuration of a such trip consists of a set of attractions together with a schedule for visiting them, as well as, possibly, additional locations for over-night stays and meals.

To come up with a good travel plan, a traveler may research the destination country thoroughly, reading travel books or talking to past travelers. Although some people enjoy this planning phase, it is clear that constructing a satisfactory trip requires investing significant time and effort. Given one's limited resources, it unlikely that she be able to exhaustively search the huge configuration space involved and design an optimal trip.

Alternatively, a soon-to-be traveler may use a sophisticated recommender system (e.g., (Venturini and Ricci 2006)) for constructing trips. In these systems the user often answers a set of required questions about her preferences, and receives a recommended plan. This plan is unlikely to be taken as is by the user, and users typically revise the recommended plan manually, by, e.g., switching attractions, changing the schedule, and so forth. This manual revision phase is needed because it is well known that expressing preferences and constraints is difficult and exhausting for users (Pu et al. 2011). Thus, one can consider the manual revision phase as implicitly expressing additional preferences and constraints that were not initially inserted.

In this paper we propose a method that can be used to automatically improve such manually constructed or revised plans. We envision our method as a component within a sophisticated travel recommender system, used after the manual revision phase. Our suggested algorithm takes as input a manual configuration from the user, with additional personal preference data. In the trip example, the user provides an initial planned trip, possibly generated through a manual revision of a previously suggested trip. The user also specifies numeric ratings for the attraction in the input trip, reflecting how interesting each chosen attraction is to her.

The system then learns from the trip and the attraction ratings the user's preferences over both attraction properties and more global trip properties. For example, from a user who rates many museums positively the system may learn that the user prefers museums to hiking. The system also learns constraints over the relationships between items in the configuration. For example, the system may learn that the user plans a trip with no more than an hour drive between attractions, and hence deduce a constraint of short distances between attractions. Thus, while the user supplies preferences on the components, the system deduces preferences and constraints about both component classes and the relationship between components, automatically. Our algorithm then translates the constraints and preferences into an integer programming (IP) problem, and runs an IP solver to obtain a solution. The solution is translated back into a revised trip, and returned to the user for inspection. The user may choose to farther revise the trip, and so forth.

To examine the utility of this approach, we ran a user study for trips to New Zealand, an attractive destination for tourists, and in particular hikers, many of which spend considerable time planning their trip. Participants in the user study, all people who toured New Zealand, provided an initial 5-6 day trip. After running our algorithm, we displayed 3 optional suggestions and asked participants to rate them. Study participants clearly preferred the revised trip, and indicated that if a system like that was available, they would have liked to use it.

Background

Recommender systems (Ricci, Rokach, and Shapira 2011) are now widely used in many applications, such as the online retailer Amazon[1] and the online video streaming service NetFlix[2]. Perhaps the most popular approach for providing recommendations is the collaborative filtering (CF) approach (Resnick et al. 1994; Breese, Heckerman, and Kadie 1998) where the system recommends to a user items that where preferred by users with similar behavior (e.g. similar past purchases). This approach requires that users provide historical data, typically through past interactions with the system. In CF, the system typically ignores the attributes of items and users, and computes similarity only based on the interactions of users with items.

A second popular alternative is the content-based (CB) approach (Lops, de Gemmis, and Semeraro 2011), where items are described using a set of attributes. For example, in our system, attractions may be described by their category (e.g. museum, or hiking), by their location, by their price, and so forth. We can also describe users in terms of preferred attributes, and then compute which items are preferred by the user. Pure CB systems require no knowledge of past user choices/preferences, or they may be augmented with CF data in the case of hybrid systems (Burke 2002).

In some cases, in addition to the information about item attribute values, the system must also maintain information about relationships between items. For example, in the case of attractions we may need to have distance information or public transportation opportunities between attractions. Two attractions in New Zealand, for instance, may be relatively nearby geographically, but due to a mountain range in between may require significant time to travel in between. Also, when visiting an attraction one must plan accommodations, dining, and so forth. The museum, the restaurant, and the hotel must be sufficiently close in order for the user to enjoy them all in one afternoon. Other information may describe how similar two attractions are. For example, two museums may be very similar, making a visit to both less desirable, or they could be distinct, and a visit to both can be interesting for a museum lover.

A system that maintains, in addition to the item attributes, relationships between attributes and items, is often referred to as a knowledge-based recommender system (Cunningham and Bonzano 1999). Such systems are naturally costly to build, as the knowledge must be identified by a domain expert, then collected and inserted into the system. Therefore, they are appropriate mainly to systems that sell relatively profitable items. Indeed, vacation packages are an obvious example of such complex and costly products that require much knowledge to construct properly, and it is no surprise that much work on knowledge-based recommenders considers this domain (Fesenmaier et al. 2003; Ricci et al. 2006).

A second, orthogonal, research direction considers the interactions of the users with the system (Cavada, Ricci, and Venturini 2003). When recommending costly and complex items, it is unlikely that the system will provide good recommendations without collecting information from the user concerning her preferences by, e.g., filling out a lengthy questionnaire. An alternative to a tiresome questionnaire is to present the user with recommended items after a very brief collection of user goals (e.g. the vacation destination). These recommendations are likely not to be satisfactory, and we must allow the user to criticize them. From the user's response, the system learns additional preference information concerning, and provides improved recommendations.

Such systems provide relatively lengthy interactions between the user and the system in order to identify good items. Therefore, such systems are only useful in the case where a user gains significant value from the items, and is hence willing to invest the required effort in a conversation with the system.

There are a few examples of previous systems designed to recommend travel destinations or plans to users (Fesenmaier et al. 2003; Cavada, Ricci, and Venturini 2003; Dema 2009; Cao et al. 2011; Gretzel et al. 2004; Bauernfeind 2003; Fesenmaier et al. 2010), in addition to many online travel planning systems with no recommendation component, such as Travelocity[3] or Trip Advisor[4]. These systems typically ask the user to specify many input parameters, such as the destination, the travel party (e.g. couple or family), duration, budget, attraction types and so forth. Then, these systems may use information over the choices of other users to recommend attractions, accommodations and so forth.

Recommending Complex Objects

In this section we outline our general approach to recommending complex objects. We explain the various components of our system, but their actual implementation is often domain dependant. Hence, we return to these components in the next section, and explain how they manifest within a specific Travel Planning application.

Figure 1 presents our complex objects recommender system, composed of 3 major components — a recommender, a constraint generator, and a constraint solver. The system takes as input an example of a possible configuration of a complex object designed by the user. It is important that the user will provide a sufficiently rich example, because the system then deduces from this example various features. Two separate types of features are learned, the user preference over components, and a set of constraints over the combination of components into a complex object. Then,

[1]www.amazon.com

[2]www.netflix.com

[3]www.travelocity.com

[4]www.TripAdvisor.com

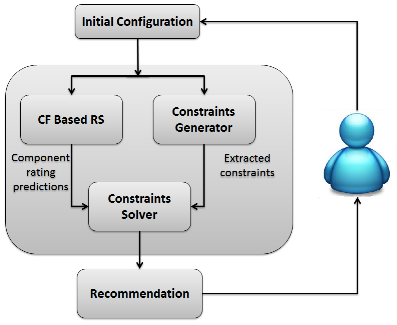

Figure 1: Complex Objects Recommender System.

the system uses the deduced features to compute a recommended complex object. This object is then returned to the user, and she may revise it manually, and provide it as input for a second recommendation.

Recommender: The recommender component takes as input a list of preferred or positively rated components. Then, it uses some rating prediction approach, such as CF, to predict ratings for other components. These ratings do not take into consideration any constraints or correlations between components.

Constraint generation: The constraint generator takes as input the constraints expressed in the given example complex object. Some constraints, such as total price, are global, i.e., apply to the complete complex object. Other constraints apply to subsets of components. For example, the maximal time allotted in the trip to travel between attractions. Finally, there are some constraints that apply to specific components. For example, the user may decide that some attractions must be kept. In our system, all attractions that got the highest possible rating, are considered irreplaceable.

Constraint solver: We now feed the results of the two previous stages to an optimizer, designed to provide a solution that satisfies all constraints while maximizing the sum of predicted ratings. The solution directly corresponds to a complex object that meets all constraints and is considered best in terms of user preferences, e.g., the sum of ratings of all attractions, or some other aggregate measure.

Travel Planning Recommender System

While the system we suggest above is designed generically to fit various types of complex objects, in this paper we focus on a travel planning application. We consider a trip to be a complex object constructed of a sequence of attractions (the components of the complex object). The system's goal is to provide a trip, i.e., a schedule of activities for a 5 day visit to New Zealand. We now explain how the generic phases in the former section manifest themselves in our application.

Database: As we explain above, our system requires a database consisting of the various possible components (attractions), their attributes, and their relationships. Our database contained 327 attractions, 21 attraction types, and 50 different areas of New Zealand. All the information in the database was taken from the popular Lonely Planet travel guide (Bain et al. 2006).

Input: The interaction of users with the system begins with inserting information about a possible trip. In our application we asked the user to specify which attractions she would like to visit during each of the 5 days. In addition, the user specified an attraction interest rating, specifying how important is the attraction, on a scale of 1 to 5. Finally, the user was asked to provide information regarding the area where she would like to spend each night.

Recommender: In order to extrapolate attraction ratings for users, we used a simple CF engine. We constructed a memory-based algorithm, using the popular Pearson correlation to compare user profiles (Breese, Heckerman, and Kadie 1998). The algorithm was run on all the users who provided initial ratings for attractions in their input.

Constraint generation: This is perhaps the most domain-specific part of our system. We defined the following set of constraints over a trip:

- Budget — we computed the overall price of the trip that the user has given us as input. We then require the budget of the revised trip not to exceed this budget by some fixed amount δ (in our experiment we used $\delta = 50NZD$).
- Daily load — we compute the minimal and maximal time required for the attractions that the user chose to visit in a single day. We use this to measure whether the user prefers a relaxed trip with much leisure time, or a packed trip, visiting as many attractions as possible.
- Travel distance — we compute maximal travel distance between attractions. This allows us to measure how much time the user is willing to spend traveling between places.
- Diversity level — a user may enter a trip with various types of attractions. We hence constrained the revised trip to also contain the same attraction types, so as not to reduce the diversity of the trip. For example, even if the recommender predicts higher scores for all hiking attractions than all museum attractions, yet the user entered a few museum attractions in her input trip, we will force the revised trip to also contain some museums. We considered here only attractions rated 3 or higher.
- Preserving important attractions — in the input the user may have specified a few attractions as "very important" (a rating of 5). In that case we force these attractions to appear in the revised trip as well.
- Local transportation constraints — while our system does not maintain transportation information between attractions, such as how to get from one museum to another within the same city, we add a fixed transportation time (30 minutes) for travel between attractions in the same city.

Constraint Optimization: We can formulate the problem of optimizing the total value of visited attractions subject to the above constraints using an integer programming (IP). To do so, we define two types of boolean variables — $x_{i,j}$ is true whenever attraction j is visited on day i, and $y_{i,l}$ is true whenever area l is visited in day i. We now explain how the constraint types above are implemented in the IP.

- **Budget:**

$$\sum_{i,j} C_j x_{i,j} \leq Budget + \delta \qquad (1)$$

specifies the budget constraints, where C_j is the cost per attraction, $Budget$ is the overall allowed budget, and δ is a fixed parameter to allow us some flexibility in exceeding the budget that the user has specified.

- **Diversity:** For each attraction type t we have a constraint

$$\sum_{i,j} Type_{t,j} x_{i,j} = \#Type_t \pm 1 \qquad (2)$$

where t is an attraction type, $Type_{t,j} = 1$ iff attraction j is of type t, and $\#Type_t$ is the number of attractions of type t with rating higher than 3 that were specified in the input example.

- **Time:** For each day i we have two constraints

$$\sum_{j} Time_j x_{i,j} \quad \geq \quad MinTime \qquad (3)$$

$$\sum_{j} Time_j x_{i,j} \quad \leq \quad MaxTime \qquad (4)$$

where $Time_j$ is the time required for visiting attraction j, and $MinTime$ and $MaxTime$ are the minimal and maximal time required to visit all attractions in a single day in the input example, respectively. We added 30 minutes to $Time_j$ to model the local transportation costs.

- **Mandatory attractions:** For each attraction j rated 5 (the highest rating) in the input example we have a constraint

$$\sum_{i,j} x_{i,j} \geq 1 \qquad (5)$$

to force it to appear in the revised trip. When we have multiple mandatory attractions, we define a traveling sales person problem on the attractions to provide the constraint solver with a given ordering of attractions, which will reduce the effort and time in solving the IP problem.

- **Attraction time:** Some attractions, such as certain hiking trips or cruises, requires multiple days. We hence add for each such attraction that requires $D_j > 1$ days a variable x_j signifying whether this attraction has been chosen. We add constraints:

$$\sum_{i} x_{i,j} = D_j \cdot x_j \qquad (6)$$

- **Distance:** The distance constraints were somewhat more complicated to define. We divide the destination region into a set of areas A. First, we add constraints that model distances between areas. For each day i and area a we define

$$|A| y_{i,a} + \sum_{l \neq a} d_{l,a} y_{i,l} + d_{l,a} y_{i+1,l} \leq |A| \qquad (7)$$

where d_{a_1,a_2} is 1 if the distance between areas a_1 and a_2 is greater than the maximum distance that the user is willing to visit in a single day, and 0 otherwise. That is, if the solution includes a visit to area a in day i then it cannot visit any other area which is farther than the maximum allowed distance on day i or the next day $(i + 1)$.

To specify which attraction belong to an area, we add for each day i, each area a, and each attraction j belonging to area a the constraint

$$x_{i,j} - y_{i,a} \leq 0 \qquad (8)$$

To specify that each attraction j is visited once at most we add a constraint

$$\sum_{i} x_{i,j} \leq length(j) \qquad (9)$$

where $length(j)$ is the number of days required for completing attraction j (for example, hiking trails may take more that a day to complete). Attractions requiring more than a single day required us to add additional variables to specify that they must be executed in adjacent days. We skip the exact definition of these variables and constraints due to space restrictions.

- **Optimization criterion:**

$$\sum_{i,j} R_j x_{i,j} \qquad (10)$$

where R_j is the predicted or specified rating for attraction j. That is, the program optimizes the sum of attraction ratings.

We now use an IP solver [5] to solve the specified integer program. Unfortunately, the computation of an optimal solution for the above IP takes too long to compute. It is possible, however, to stop the solver after a predefined time and request the best current solution. We would like to ensure, though, that every returned solution is at least as good as the original program. We hence add a constraint

$$\sum_{i,j} R_j x_{i,j} \geq R_{example} \qquad (11)$$

where $R_{example}$ is the sum of ratings of the attractions in the trip that was inserted as input by the user.

Any solution to the IP problem is mapped directly into a specification of which attractions to visit in each day, using the $x_{i,j}$ boolean parameters.

Empirical Evaluation

We now describe our evaluation of the trip revision recommender system presented in the previous section. Clearly, evaluating the system must be conducted in a user study with people who have knowledge of the destination region (New Zealand), so that they can provide an initial trip. While the system is intended for people who are planning a trip, we evaluate it using people who have already visited New Zealand, enabling educated judgments of the revised trips.

While we envision our method as a component within a larger recommender system that helps the user in constructing an initial configuration (trip), in this paper we evaluate our suggested method as a stand-alone system, leaving its evaluation within a complete system to future research. In our experiments, we did not use a complete system, rather, users were asked to specify a trip in a given format. The trip was fed into to the algorithm, and the results were translated into a map showing the trip. Figure 2 show an example of a suggested trip displayed to the users.

The users in our study were thus asked to specify a 5 day trip to New Zealand. They were asked to provide a sequence

[5]In our experiments, we used lp_solve (http://lpsolve.sourceforge.net/5.5/).

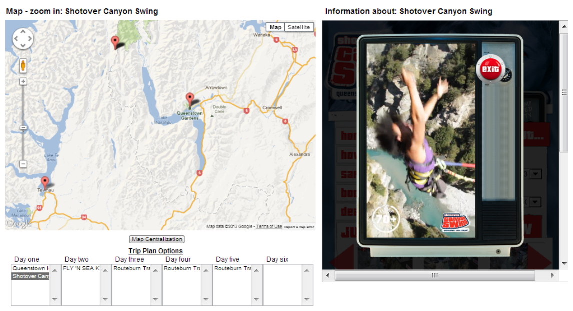

Figure 2: Example of a trip suggested to users. Clicking on a day on the bottom centralizes the map on the attractions designed for that day. Clicking on an attraction on the map opens an information page for that attraction.

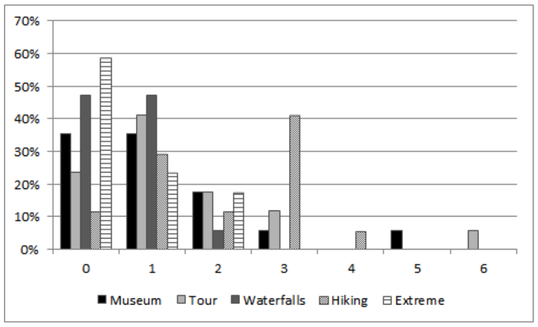

Figure 3: The distribution of the most popular attraction types in input trips. The x-axis is the number of attractions of that type per trip. The y-axis is the portion of the users who chose this number of attractions in their input trip.

of attractions that they would like to visit in their trip, and to specify in which city they would spend each night. Users were also requested to add an interest rating for each attraction in the trip that they constructed. In a complete system, this would amount to specifying how much the user thinks that each of the attractions suggested by the system for the trip is interesting after, e.g., reading information about the suggested attractions online.

These manually constructed trips were ran through our system and we computed 3 different suggestions:

1. Best revision: the output of the algorithm of the previous section, within the given time limit.
2. Conservative revision: a revision with an additional set of constraints forcing at most two attractions that were ranked lowest to be replaced.
3. Best attraction list: a list of the attractions with the highest predicted rating, with no trip planning, as is done by simpler recommendation systems.

The alternatives were presented in a random order to reduce biasing due to presentation order. The users could observe the revisions, watch the suggested trip on the map, and display information from Wikipedia concerning the attractions in a separate pane on the user interface.

The users were then asked to fill a short survey asking questions about the 3 suggestions. We used a 1-6 scale so as to avoid users making neutral judgments.

Collecting Ratings for New Zealand Attractions

As a part of this research, we have constructed a simple online application that allows travelers to New Zealand to provide ratings for attractions that they visited. This application was installed by the owners of a number of guest houses in New Zealand, which are popular among travelers. The application contained a list of attractions from the Lonely Planet travel guide (Bain et al. 2006), but users could also add additional attractions. We collected more than 600 responses of travelers through this online application, and we use the collected ratings to perform the collaborative filtering predictions over attractions.

Experimental Results

We had 19 participants in our user study, ages 20 through 45. All of them have visited New Zealand in the past 5 years. We had 12 male participants and 7 females. Although this is certainly not a large user study, we still got significant results, which further suggests that the differences between the methods are very noticeable.

Diversity of the input trips: Figure 3 shows the distribution of the 5 most popular attraction types. We can observe that the user population had museum lovers, and people who never visit museum, avid hikers alongside people who choose one or less hikes, and so forth. In addition, there were 7 attraction types, such as cruises, fishing, and horse riding, that were selected by only a single person.

Our system extracts several constraints from the input trips, such as the minimal and maximal time per day, and the overall total budget (for attractions only). Here we again spot significant diversity of user preferences. For the minimal time per day, 16% of the users allowed for a very relaxed day of less than 3 hours for visiting attractions, 67% allowed between 3 to 4 hours in their most relaxed day, and 17% had more than 4 hours in their most relaxed day. For the maximal time per day, 22% had 6 hours at most in their busiest day, 39% had 6 to 7 hours in their busiest day, and 28% had more than 7 hours in their most busy day. For budget constraints, 28% spent less than 150 NZD on attractions, 28% spent 150 to 250 NZD, another 28% spent 250 to 350 NZD, and 22% spent 350 to 625 NZD.

These extracted preferences, along with the attraction ratings that the users supplied, were inserted into the system and the 3 suggestions were computed and presented to the users. After presenting the users with the suggestions, they were asked to fill a short survey.

User acceptance of automated suggestions: We asked several general questions regarding the participants perception of recommender systems in general and our system in particular: 90% (17) of the participants said that they feel that automated recommendations can properly identify their preferences. 90% (17) of the participants said that trip planning is difficult and requires much effort, and 95% (18) of the participants thought that our recommender system for

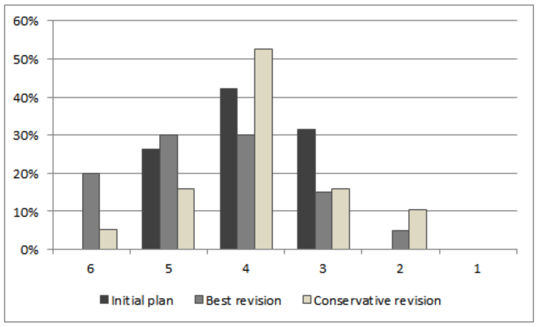

Figure 4: The distribution of grades on a 1-6 scale for the initial plan and the two revisions presented to the users.

trip planning will be successful, if made available widely. 95% of the users claimed that given the system suggestions they would have made some change to their initial plan: 53% would make minor changes and 37% would make significant changes. Users show willingness to accept the suggestions of automated systems. It seems that the users in our study believe that automated systems can help in the complex task of constructing trips.

Attractions load balance: A possible concern with trip planning is that the number of attractions per day is too high and the plan becomes overloaded. This is particularly true given our objective function which seeks to maximize the *sum* of ratings of visited attractions, albeit subject to load constraints derived from the original plan. 90% of the users claimed that the best revision was well balanced and was not overloaded with too many attractions per day. Only 10% said that the travel plan was overloaded.

Quality of the plan revisions: We now turn to an evaluation of the travel plans and their revisions. Figure 4 shows distribution of grades for the revisions. The average grade for the initial plan was 3.94, while the average for the best revision was 4.45, and the average for the conservative revision was 3.89. In 10 times out of 19, the best revision got a higher grade than the initial plan, and for 6 users both plans got the same grade. Only 3 users gave a higher grade to their initial plan than the best revision. A sign-test gives a p-value of 0.002 to the null hypothesis that the revised plan is no better than the initial plan.

Only 6 participants marked the conservative revision to be better than the original plan, and 4 gave the original plan higher grades than the conservative plan. We allowed the participants to comment on the plans, and we got responses criticizing the conservative plan as being not novel enough.

We also asked the participants which suggestion they preferred. 79% (15) of the users preferred the best revision suggestion over the other two suggestions. These results clearly suggest that, as expected, in general users found the best revision to be most appropriate then the other two plans.

Attraction list vs. complete plans: Classic tourism recommendation systems suggest a list of attractions for a user. In this paper we assume that suggestions of complete trips is more useful to users. Indeed, 90% preferred a suggestion for a complete plan over a list of interesting attractions. In general, for complex objects constructed of multiple components, these results demonstrate the value of supplying the actual target object, rather than useful components.

Return on investment: The construction of an initial plan represents a significant effort, and we asked whether the effort was worthwhile. Only one participant claimed the effort was too high for the given results. One other participant said that the effort is high, but the results are interesting. 26% (5) of the participants said that the effort is considerable, but that when planning a trip the effort is acceptable, and the rest 63% (12) of the participants said that the effort is minor given the results.

Related Work

Our work is somewhat related to the general problem of re-configuration, where a system modifies or fixes a given configuration. For example, Stumpner et al. (Stumptner and Wotawa 1998) uses methods originating in model-based diagnosis in order to identify the cause that a given configuration does not match the user goals. As is often the case in diagnosis, they formulate their problem, as we do, as a constraint optimization problem. In contrast to our work, they do not assume that the current configuration expresses user preferences and thus, they do not extract such preferences from the configuration. Their work is more general and thus they do not use CF techniques as we do. They provide an example of a telephone network reconfiguration problem.

Falkner et al. (Falkner, Felfernig, and Haag 2011) brings the areas of configuration and recommender systems together, discussing a diverse set of advantages that could be gained, including explanations, recommendations of features, and recommendations for feature values. For example, using a recommendation component can be used to eliminate several features from the problem, thus reducing the search space considerably for the configuration engine. They also suggest that a user will rank features or components by their importance, allowing us to compare his taste to the tastes of other using CF.

Coster et al. (Coster et al. 2002) suggest an interactive method, as we do, for constructing a computer configuration. In their method a user receives recommendations for various components, given the current components that she has selected. They begin by asking the user questions concerning her needs and suggest an initial configuration given the user preferences. For example, a user who expresses an interest in graphics, will be suggested a system with stronger graphical components. Thus, they do not have a mechanism for extracting the user preferences from an initial configuration. No validating user study is reported.

Zanker et al. (Zanker, Aschinger, and Jessenitsching 2010) suggest a system which is closer to our own, using constraints and CF techniques for constructing a single city trip. They attempt to find a resort and a set of attractions within the same area which will fit the user preferences. As opposed to us, they do not begin with an initial configuration expressing the user preferences, but start with the user expressing her preferences where each preference becomes a constraint. After seeing the result of the constraint solver, the user can further criticize the solution, which results in additional constraints being added. The CF component is used to provide a preference for certain attractions, which needs to be weighted against the existing constraints. They also do not report a user study.

Conclusion

In this paper we presented a recommendation component that takes as input a manually configured or tuned complex object, such as a trip to a country, and some preference data, such as attraction ratings, and produces a better configuration, e.g. an improved trip, taking into account a set of constraints that were extracted from the input example.

In a user study, we demonstrated that for the problem of planning a 5 day trip to New Zealand, our component was in most cases able to produce better trips than the ones that were manually constructed by the users. We also showed that users prefer a recommendation for a trip over the more traditional list of recommended attractions.

In the future we plan to test our component within a complete recommender system, as an integral part of the trip planning process. In addition, we intend to test our ideas in other domains where complex objects need to be configured, such as configuring an appropriate computer system.

References

Aldanondo, M.; Moynard, G.; and Hamou, K. 2000. General configurator requirements and modeling elements. In *ECAI Workshop on Configuration*, 1–6.

Bain, C.; Dunford, G.; Miller, K.; O'Brien, S.; and Rawlings-Way, C. 2006. *New Zealand — Lonely Planet*. Lonely Planet.

Bauernfeind, U. 2003. The evaluation of a recommendation system for tourist destination decision making.

Breese, J. S.; Heckerman, D.; and Kadie, C. M. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In Cooper, G. F., and Moral, S., eds., *UAI*, 43–52. Morgan Kaufmann.

Burke, R. D. 2002. Hybrid recommender systems: Survey and experiments. *User Model. User-Adapt. Interact.* 12(4):331–370.

Cao, T.; Nguyen, Q.; Nguyen, A.; and Le, T. 2011. Integrating open data and generating travel itinerary in semantic-aware tourist information system. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, 214–221. ACM.

Cavada, D.; Ricci, F.; and Venturini, A. 2003. Interactive trip planning with trip@dvise. In Rauterberg, M.; Menozzi, M.; and Wesson, J., eds., *INTERACT*. IOS Press.

Coster, R.; Gustavsson, A.; Olsson, T.; sa Rudstrm; and Rudstrm, A. 2002. Enhancing web-based configuration with recommendations and cluster-based help. In *In Proceedings of the AH'2002 Workshop on Recommendation and Personalization in eCommerce*.

Cunningham, P., and Bonzano, A. 1999. Knowledge engineering issues in developing a case-based reasoning application. *Knowl.-Based Syst.* 12(7):371–379.

Dema, T. 2009. *eTourPlan: A Knowledge-Based Tourist Route and Activity Planner.* University of New Brunswick (Canada).

Falkner, A. A.; Felfernig, A.; and Haag, A. 2011. Recommendation technologies for configurable products. *AI Magazine* 32(3):99–108.

Fesenmaier, D.; Ricci, F.; Schaumlechner, E.; Wöber, K.; and Zanella, C. 2003. Dietorecs: Travel advisory for multiple decision styles. *information and communication technologies in tourism* 2003:232–241.

Fesenmaier, D. R.; Xiang, Z.; Pan, B.; and Law, R. 2010. An analysis of search engine use for travel planning. In *ENTER*, 381–392.

Gretzel, U.; Mitsche, N.; Hwang, Y.-H.; and Fesenmaier, D. R. 2004. Tell me who you are and i will tell you where to go: Use of travel personalities in destination recommendation systems. *J. of IT & Tourism* 7(1):3–12.

Lops, P.; de Gemmis, M.; and Semeraro, G. 2011. Content-based recommender systems: State of the art and trends. In Ricci et al. (2011). 73–105.

Magro, D., and Torasso, P. 2000. Description and configuration of complex technical products in a virtual store. In *Proc. ECAI 2000 Configuration WS*, 50–55.

Pu, P.; Faltings, B.; Chen, L.; Zhang, J.; and Viappiani, P. 2011. Usability guidelines for product recommenders based on example critiquing research. In Ricci et al. (2011). 511–545.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In Smith, J. B.; Smith, F. D.; and Malone, T. W., eds., *CSCW*, 175–186. ACM.

Ricci, F.; Cavada, D.; Mirzadeh, N.; and Venturini, A. 2006. Case-based travel recommendations. *Destination Recommendation Systems: Behavioural Foundations and Applications* 67–93.

Ricci, F.; Rokach, L.; Shapira, B.; and Kantor, P. B., eds. 2011. *Recommender Systems Handbook*. Springer.

Ricci, F.; Rokach, L.; and Shapira, B. 2011. Introduction to recommender systems handbook. In Ricci et al. (2011). 1–35.

Soininen, T.; Niemelä, I.; Tiihonen, J.; and Sulonen, R. 2000. *Unified configuration knowledge representation using weight constraint rules*. Citeseer.

Stumptner, M., and Wotawa, F. 1998. Model-based reconfiguration. In *In Proceedings Artificial Intelligence in Design*, 45–64. Kluwer Academic Publishers.

Venturini, A., and Ricci, F. 2006. Aplying trip@dvice recommendation technology to www.visiteurope.com. In Brewka, G.; Coradeschi, S.; Perini, A.; and Traverso, P., eds., *ECAI*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, 607–611. IOS Press.

Zanker, M.; Aschinger, M.; and Jessenitsching, M. 2010. Contraint-based personlaized configuration of product and service bundles. *International Journal on Mass Customization* 3(4):407–425.