A Travel Recommender System for Combining Multiple Travel Regions to a Composite Trip

Daniel Herzog
TU München
Boltzmannstr. 3
85748 Garching
Germany
herzogd@in.tum.de

Wolfgang Wörndl
TU München
Boltzmannstr. 3
85748 Garching
Germany
woerndl@in.tum.de

ABSTRACT

Internet-based services are available to recommend destinations and activities for organized trips. Only few systems support travelers when creating composite trips consisting of multiple destinations or activities. The idea in this work is to select travel regions that maximize the value of the composite trip for the user while still respecting her limitations in time and money. The value of a travel region can be determined by the similarity between a specified user query and the cases in a travel region database. The recommendation algorithm needs to find a decent routing between the regions while still satisfying diversity of the whole trip. We developed an algorithm based on an approximation for the knapsack problem and extended it to recognize dependencies between the regions while calculating best combinations. It is able to determine the optimal duration of stay per region and its performance improves when benefiting from the hierarchical structure of our travel database. In an expert study, we verified the results of our approach. The study proves that our algorithm for composite trips delivers good recommendations which satisfied an expert user more than baseline algorithms. Regions in the composite trip fit together better and a decent routing between the regions can be ensured. Nevertheless, the algorithm leaves room for improvement by combining less similar regions in a composite trip, thus leading to a higher diversity of the recommendation.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Dynamic programming

General Terms

Algorithms

CBRecSys 2014, October 6, 2014, Silicon Valley, CA, USA.

Keywords

case-based recommender system, tourist trip design problem, travel recommendation, knapsack problem, similarity

1. INTRODUCTION

Recommender systems are an established technology in online shops to provide users with a list of recommended items during their shopping experience. In comparison to recommending single items, travel recommendation turns out to be a more complicated issue since tourists have to deal with both limited budget and time frame. If users call for an individual trip composed of multiple regions, the task becomes even more complex. An exemplary situation illustrates the difficulty: A person is looking for a trip for four weeks in September with a budget of 2000 Euros; she likes nature and hiking with some cultural highlights as a plus. A high number of possible routes could be packaged to a composite trip and the constraints in time and money have to be taken into account. This is a special case of the so called tourist trip design problem (TTDP) [14]. TTDP is an extension of the orienteering problem (OP): A score which determines the value for the user is assigned to every location in a sequence. The goal is to maximize the sum of the scores of the selected locations while still meeting the given limitations [16].

Recommender systems use different algorithms to find a suitable list of recommendations in a feasible time. A subarea of content-based recommender systems is case-based recommender systems. Case-based recommenders rely on items structured as cases using a well defined set of features and feature values. Those cases are compared to a user query or her profile and the most similar cases are delivered as a recommendation [13].

These recommenders are already applied for travel recommendation. In order to recommend a longer trip, the cases have to be combined to maximize the value while still respecting the users' limitations. The simplest approach to determine the value of a composite recommendation is to add the values of all single parts of the proposed trip. This approach does not consider dependencies between the parts, for instance travel activities which cannot be executed at the same time as other activities. Furthermore, determining the maximum score of all possible combinations is not computationally feasible. This is the reason why these algorithms need to work with heuristics.

This paper aims to investigate possible solutions for combining multiple travel regions to a composite trip for in-

dependent travelers. The more specific research problem is whether an algorithm which recognizes dependencies between items and is based on a hierarchically structured data model can lead to better recommendations of a sequence of items. The rest of this paper is organized as follows. Section 2 looks at related work in the field of travel recommendation. In Section 3 we present the underlying data model we developed for testing our algorithm. Section 4 explains the main idea, the single steps and the implementation of our algorithm. This algorithm is evaluated in Section 5. Section 6 concludes our work and presents future work.

2. RELATED WORK

Research already focuses on travel recommendation and bundling k items to a recommendation package. Related work is [7] who developed the TAST model which represents the interests of tourists as well as extracted features of regions like the location or travel seasons. Travel packages are recommended to the user based on this model combined with additional information like prices. [17] developed a composite recommender system with access to one or more underlying recommender systems and therefore devises two algorithms for generating top-k packages as recommendations, both with high quality and one being much faster than the other. Furthermore, [1] explains how to bundle recommendation packages by regarding relationships and associations between the entities. Hence, the best recommendations for a given set of keywords can be determined. Early pruning and terminations strategies are used to develop an efficient algorithm.

[14] introduces the TTDP, shows the connection to the OP and presents a fast algorithm that produces solutions of better quality by using a guided local search meta-heuristic. [5] tries to solve the TTDP by presenting a cost-aware travel tour recommender, while Trip-Mine is looking for optimal trips by satisfying the user's travel time constraints [8].

[9] shows that case-based recommenders can be used to bundle travel items. The developed recommendation methodology Trip@advice stores recommendation sessions as cases. Furthermore, the user can give direct feedback to invoke suggested query changes during the recommendation process. [12] confirms that case-based reasoning and making association rules are solutions for recommending tourism travel packages. [3] devises a hybrid algorithm that additionally includes collaborative filtering in the recommendation process. The bundling of trips to packages in [15] is based on an object orientation solution which faces the high variation in travel requirements.

[10] developed DieToRecs, a travel recommender system which offers personalized interaction during the recommendation process to create individual bundles of trips. [11] shows the application of automatically collected constraints and preferences which can be added to user queries in order to improve the results of complex trip recommender systems. Further approaches make use of geo-location and pictures of the travel entities [4].

Our developed algorithm combines multiple travel regions to a composite trip. It is based on an approximation for the knapsack problem and extends the existing approaches by taking dependencies between single regions into account. The algorithm considers the fact that the value of each region in a composite trip is dependent on the presence or absence of other regions in the same recommendation. Furthermore, it calculates the optimal duration of stay per region in the composite trip and benefits from the hierarchical structure of the underlying data model.

3. UNDERLYING DATA MODEL

In this paper, we aim to recommend trips composed of multiple regions for individual traveling. We have developed a travel database with realistic data in order to test the proposed algorithm. The data model is composed of several layers ordered in a hierarchical manner as explained in this section.

In our model, a region is always a subregion of another region while the world is the parent region of every region. Regions contain the necessary information for recommendation:

- How good a region matches traveler types such as *Free spirits* or *Cultural explorer*, on a 5-point Likert scale
- Minimum and a maximum recommended duration of stay
- Minimum and optimal budget a traveler has to spend per pay
- Recommended periods of traveling in the region as 5-point Likert scale per month
- Security for travelers (crime level), on a 5-point Likert scale

If a region lacks specified attributes, it inherits the values from the parent region. We also model the connections between regions by specifying the necessary effort (time and cost) in one number to travel from one region to another. The connection cost between neighboring regions is 0. Regions are part of a country or can be spread over several countries. For example, larger countries such as the USA are divided in several travel regions, while smaller countries such as the Netherlands, Belgium and Luxembourg are combined into one region. Furthermore, regions contain one or more routes, i.e. concrete itineraries to visit a region. At this time, our systems recommends regions only, but in future work, routes can be considered for recommendation as well. The model can also be extended with additional attributes such as spoken languages in regions. Regions can be assigned to the countries they belong to and store additional information like visa requirements, but we do not use countries for recommendation at this time.

Figure 1 illustrates the data model by showing an example with several layers of regions. USA Southwest, USA Pacific Northwest and Western Canada are subregions which can be recommended by our algorithm. When developing and testing our algorithm, our database was composed of a total of 152 regions with 124 leaves in the region tree which can be recommended in a trip. The data was complied based on various Internet sources.

A user who asks for a recommendation chooses one traveler type which expresses her expectations towards trips. The offered traveler types such as *Free spirits* or *Cultural explorer* are inspired by a market segmentation tool of the Canadian Tourism Commission[1]. Based on the described

[1]http://en-corporate.canada.travel/resources-industry/explorer-quotient

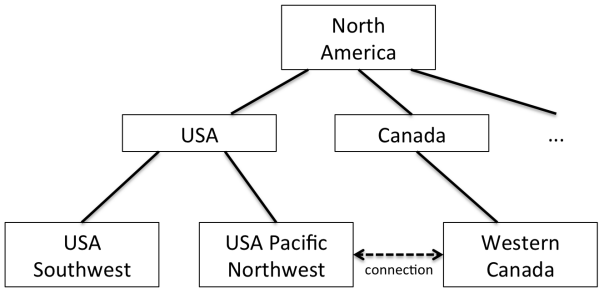

Figure 1: Data example from region tree

preferences and suggested activities for each traveler type, we rated each region in the database to express how it matches the traveler types on a 5-point Likert scale.

4. ALGORITHM FOR THE RECOMMENDATION OF COMPOSITE TRIPS

In this section, we present our algorithm to recommend trips composed of multiple travel regions. This algorithm can serve as a basis for a fully functional travel recommendation application.

4.1 Basic idea and goals

Composite trip recommendation can be seen as a special case of the knapsack problem to find best combinations of multiple travel items [17]. The underlying idea is to combine as many single travel items like regions, routes or activities as necessary to maximize the benefit for the user while still respecting existing limitations like time and money. The problem can be described formally as:

maximize

$$\sum_{i=1}^{n} f_i(x_i, X_i)$$

subject to

$$\sum_{i=1}^{n} d_i \cdot x_i \leq D$$

and

$$\sum_{i=1}^{n} b_i \cdot x_i \leq B$$

with

$$x_i \in \{0, 1\}$$

where $f_i(x_i, X_i)$ is the value function for item i and X_i is the set of items upon which the value of item i depends. x_i is either 0 or 1 which means that each travel item can be added to the travel sequence only once (or not). d_i is item i's recommended duration of stay and b_i is item i's recommended daily budget. D is the maximum possible duration of stay and B is the maximum budget the user can spend on her trip. In this paper, travel items are represented by regions from all over the world.

The first challenge that arises is to determine the value a single region offers to the user. This value is dependent on the user's requirements and preferences for her trip. Hence, a travel recommendation algorithm needs to collect information like the user's traveling type or preferred activities, her intended period of traveling and her monetary and time limitations. The algorithm needs a predefined way to calculate the value by using this information. To decide whether a region will be considered for a trip, we calculate a rating (see below).

In our case, the problem gets more complicated than the standard knapsack problem because the value of a region is not only determined by the user query. Rather, it depends on the presence or absence of other regions in the recommended composite trip. This extension of the knapsack problem is called the Oregon Trail Knapsack Problem [2]. Imagine a travel sequence that recommends visiting Germany, Austria and Switzerland. If the user could spend more time and money, the system can add further regions to this trip. Depending on the user's preferences, additional regions in or close to Central Europe should be recommended. Only exceptional circumstances legitimate an additional region far from Central Europe because the effort of visiting this region during this trip is disproportional. The value of the composite trip itself is lower than the sum of the region values because the distance between regions has a negative impact on the composite trip.

In addition, a region's value in a composite trip influences the diversity of the sequence. For instance, if the recommender accepts different activities as user input, the algorithm should focus on a decent coverage of all demanded activities. Downgrading the values of similar regions, when combining them in a recommendation, allows avoiding situations where only a few activities are served. If a user wants to go skiing and swimming, a recommender should not only recommend regions for one of these activities, but both.

In this paper, regions instead of routes are recommended. Regions in our data model are not limited to specific activities, diversity is mainly expressed by the the duration of stay in each recommended region. Every additional week a user stays in a region leads to a lower value for the user because the probability that she explored this region enough is increasing. Hence, another goal of the algorithm is determining the optimal duration of stay per region in the travel sequence.

To conclude, a region reaches its maximum value for a given query when regarding it exclusively, and not within a composite trip. Other regions in the composite trip can decrease this value because of increasing distances and lack of diversity. This negative impact can be calculated by a penalty function. Hence, the value function for the composite trip recommendation problem extends the value functions of the Oregon Trail Knapsack Problem by a penalty function [2]. The resulting value function can be described formally as:

$$f_i(x_i, X_i) = x_i \cdot v_i - \sum_{e \in X_i} (t(i, e) \cdot x_i \cdot v_i \cdot [x_e > 0])$$

where v_i is the value of region i for the specified user query and $t(i, e)$ is the penalty function for two regions i and e. $[x_e > 0]$ represents a Boolean value whether item i is in the knapsack. The algorithm needs to provide an implementation of $t(i, e)$.

After determining the best regions and the durations of

stay for the final composite trip, an optimal routing should be ensured. The problem of combining regions to a composite trip is part of the orienteering problem. Basically, the orienteering problem extends the knapsack problem by charging time and money for the routing between the travel items [14]. Hence, finding the shortest and cheapest routing between regions reduces loss of time and money when executing the algorithm.

We have developed a travel sequence recommender based on the explained data model (cf. Section 3). The algorithm exploits the hierarchical structure. For instance, if the user wants to travel through Europe but already explored Scandinavia, there is no need to execute any calculations for other continents or Scandinavian regions. As a consequence strictly respecting the hierarchical structure promises improvements in the algorithm's runtime.

The basic idea is implemented in an algorithm that aims to achieve these targets. It is composed of three phases which are gradually executed on the available database:

1. Reduce number of regions
2. Rate regions
3. Calculate the best combination of regions

First, the approach reduces the number of considered regions for the recommendation process. Users can explicitly exclude regions in their query in our approach. If one or more regions are excluded, all subregions are removed from consideration as well, utilizing the hierarchical structure of our travel database. Reducing the number of regions means fewer items for the next two steps and therefore, the algorithm's runtime is improved.

In the following, we describe the other two steps in more detail.

4.2 Rate regions

The remaining travel items of the pruned region tree are rated in the following step. In our case, only the leaves are considered for recommendation, reducing the number of items in the rating phase. This behavior can be adapted according to the recommender's use case.

Travel regions in our scenario can be rated in several ways. For case-based recommender systems, the similarity between the user query and the case - the region - can be calculated in order to determine the value of a region for a specified user query. [13] explains that the assessment of similarity in case-based recommender systems involves combining the individual feature level similarities for relevant features. Relevant features in our travel recommender are the traveling type served by a region, recommended traveling periods etc. (cf. Section 3). Budget and duration of stay are not taken into account at this step because they are used as the constraints for the knapsack algorithm (cf. Section 4.3). The similarity between single features of the query and the case can be calculated with the following formula:

$$sim_{feature}(f_q, f_c) = 1 - \frac{|f_q - f_c|}{max(f_q, f_c)}$$

[13], where f_q is the feature expectation defined in the user query and f_c is the actual feature value in the case. The feature level similarities determine the similarity between the query and the case by weighting all features:

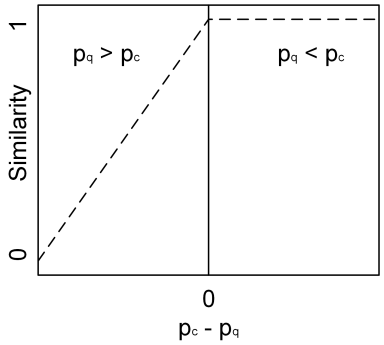

Figure 2: Asymmetric similarity metric (adapted from [13])

$$Similarity(q, c) = \frac{\sum_{i=1...n} w_i \cdot sim_i(q_i, c_i)}{\sum_{i=1...n} w_i}$$

[13]. In our case, the traveling type and the traveling period are weighted higher than other features like crime level because the assumption is that the first two features are more important for the decision.

[13] presents two examples of similarity metrics which consider deviation of a case feature from the user query. A symmetric similarity metric reduces the similarity by the same value if the query feature is lower or higher than the case feature. An asymmetric metric prefers either higher or lower values. For example, the price of an item usually corresponds to an asymmetric metric. If a user is prepared to pay 1000 euros for an item, a price of 800 euros satisfies her requirement better than a price of 1200 euros.

For our recommender, we use the following asymmetric similarity metric. All features of the regions are rated in a range between 0 and 1. For example, the perfect month to visit a region is rated with 1, while the worst possible rating is 0 (the particular month is not recommended at all). The deviation of the query from a case reduces the overall similarity if the case feature is rated lower than the expected value which usually is 1, the best value. If a user expects a region only to fit somewhat with regard to a certain feature, the query feature will be rated with a value lower than 1. A higher value of the case feature always leads to a similarity of 1 since no user would complain about a feature being better served by a region than expected. Figure 2 illustrates this metric. The similarity reaches the maximum when the case feature is rated greater or equal 1.

At the end of step 2, regions with a low rating are removed from consideration in order to reduce the number of items for step 3. In our implementation, we exclude regions with a rating lower than 0.7 on the scale from 0 to 1. This approach also prevents composite trips including regions which do not satisfy the user's demands but could be cheap enough to be considered in a recommendation only for the use of free capacities.

4.3 Calculate the best combination of regions

In step 3, the remaining regions in the recommendation process are combined in a way that maximizes the value for the user while still respecting her limitations in time and money.

The algorithm extends the classic knapsack problem in order to achieve two additional objectives. First, the value of a composite trip should be decreased according to the distances of regions. After testing several variants, we implemented a penalty function $t(i, e)$ which decreases the total value of the composite trip by a percentage depending on the connection costs between the regions i and e. This implementation is evaluated in the expert study in Section 5.

The second objective is the determination of the optimal duration of the stay per region while calculating the best combination of regions. We assume that travelers stay longer in regions with a significant higher value than in other regions of the same composite trip. On the other hand, simply recommending the maximum possible duration of stay in the best-rated region of phase two is not the best solution either because this could decrease the diversity of the recommended trip when visiting only a low number of regions.

Therefore, we suggest regarding every week of every region separately. The value of staying in a region for an additional week is lower than in the week before. We decided to decrease the value after every week by a constant between 5 and 10 percent. The lower the maximum duration of traveling, the higher the deduction. This ensures diversity even for short trips. Splitting regions in one week blocks means that the algorithm is executed on an increased number of items which extends the runtime. Nevertheless, this approach allows determining the optimal durations of stay. If diversity is only determined by the duration of stay in each region, there is no need to extend the penalty function $t(i, e)$ because deductions are already stored in the one week blocks. If, for instance, routes characterized by activities are recommended, the penalty function can be extended by a formula which calculates the diversity of routes.

The knapsack problem itself can be solved by different approximation. We decided to implement a dynamic programming solution which promises good results by splitting the problem into smaller subproblems [6]. This approach iterates over the number of available regions n as well over all limits, in our case the budget B and maximum duration of stay D. The runtime of this algorithm is $O(n \cdot B \cdot D)$. When executing, it creates a three dimensional matrix with $n \cdot B \cdot D$ subproblems. The maximum possible value for a composite trip can then be derived from this matrix. We store the selected regions in every entry of the matrix to access the regions of our final recommendation after the algorithm has terminated.

Shortest path algorithms can be applied on the final recommendation in order to optimize the routing. For our scenario, we implemented a simple approach for finding the best sequence of regions in the trip. For each added region in a knapsack with at least one region, the algorithm calculates at which position the new region includes the lowest additional costs. This region is then inserted at the identified position.

4.4 Implementation

We developed a Java 1.7 application in order to implement the algorithm and to test it in a real scenario. The travel data of our data model is stored in a NoSQL database. The application accesses regions, connections and the corresponding data by parsing an online provided JSON file.

The application offers a simple user interface which allows specifying individual queries for composite trip recommendation. The user can enter one predefined traveling type, her preferred month of traveling, her budget and average spending as well as the maximum possible duration of traveling. Furthermore, regions can be excluded from the recommendation process.

Table 1: Recommended composite trip for an example query

North Argentina and Paraguay	2 weeks	EUR 560
Bolivia	3 weeks	EUR 525
Peru	3 weeks	EUR 630
Sum	8 weeks	EUR 1715

Table 1 illustrates the outcome of an example query[2]. Imagine, a user sees herself as cultural explorer who wants to travel in August. She has a budget of 2000 euro and usually spends a low amount of money when traveling. Her maximum time of traveling is eight weeks. As she already knows Europe and Asia very well, she excludes these regions in her query. Every query is composed of this information and it is automatically extended by an expectation of the lowest possible crime rate. In this example, the best recommendation is composed of three different regions with total costs of 1715 euro. The costs already include local transport within the region and to the borders. The selected regions can be visited overland by passing mutual borders. This is why there are no connection costs in this case. Additional costs would occur if the recommendation demands traveling to a non-neighboring country or region. For future work, we suggest extending the model by specific costs for other means of transportation like buses or trains in order to calculate the overall costs more accurately. The long-distance travel from the starting point of the trip to the first region is never included.

5. EXPERT STUDY

The research question of this paper is whether an algorithm which recognizes dependencies between items and is based on a hierarchically structured data model can improve travel recommendations. We developed an algorithm that is able to consider distances between single regions and ensure diversity by calculating optimal durations of stay. We conducted an expert study to evaluate its performance. The expert had to be familiar with the available traveler types and has knowledge in the travel domain.

5.1 Procedure of the study

Our user study is composed of 56 sample queries. Basically, the queries are selected randomly but we tried to define 7 queries per traveler type. Furthermore, each query in all of the 8 traveling type groups changes only one or two features compared to the precedent query like an increased budget or less time to travel. This allows to understand how a recommendation changes if you modify certain features.

The developed Java application executes each query and presents the best-rated recommendation, based on the explained procedure. In addition, two additional (baseline) algorithms deliver two further recommendations. All of these

[2]The stated cost covers the minimum a traveler would need to spend in these regions, the cost is (much) higher when the user specifies average or high amount of spending

three recommendations are presented in a random order in order to prevent the expert from relating the recommendations to the algorithms. The recommendations are presented to the expert like in table 1, with the duration of stay per region and the total costs of the trip.

The expert rated all recommendations in these four categories on a 5-point Likert scale: 1. General satisfaction with the recommendation, 2. the diversity of the recommendation, 3. how well the single recommendation items fit together and 4. if the routing between the singles items is reasonable.

5.2 Comparative algorithms

Two further algorithms deliver recommendations based on two different approaches: A baseline algorithm that is a standard implementation to solve the knapsack problem, and a top-k algorithm that selects regions sequentially from an ordered list of rated regions. Both algorithms are used for comparison in the expert study.

Related works already implement variations of the classical knapsack problem [17] or refer travel package recommendation to the orienteering problem [14]. The baseline algorithm works like our presented travel recommendation algorithm but does not include our extensions. Thus, the values of items in a composite trip are not dependent on the presence or absence of other items and no weekly calculated penalties are considered. This algorithm allows evaluating the influence our extensions have on the quality of composite trip recommendations.

The top-k algorithm ranks all available regions by their value in a list. It is able to implement weekly calculated penalties and to decrease the total value by the distances to be covered, but it tries to find the best combination of regions in a simpler approach. The algorithm inserts every region from top to bottom in the ordered list into the recommendation sequence. If a region cannot be inserted because of the constraints with regard to money or time, the next region in the list is checked. Hence, the algorithm goes through the list of regions exactly once. The top-k algorithm allows determining if a simpler approach of the knapsack problem can lead to similar results as our more complex travel recommendation algorithm.

5.3 Results

Figure 3 illustrates the performance of the three algorithms in each of the four categories. Our travel recommendation algorithm resulted in the highest overall satisfaction to the expert (∅: 4.02, σ: 0.82). Furthermore, it is best-rated in the categories *regions fit together* (∅: 4.29, σ: 0.76) and *routing* (∅: 4.16, σ: 0.95). In terms of diversity, our travel recommendation algorithm (∅: 3.11, σ: 0.91) is rated somewhat lower than the two comparative algorithms. In this category, the baseline algorithm is ranked first place (∅: 3.57, σ: 0.84). The top-k algorithm is ranked second in every category. The results show that in 3 out of 4 categories, our extensions lead to significant improvements. Some improvements can also be observed when applying the top-k algorithm over the baseline, but the more complex knapsack based travel recommendation algorithm performs better than both.

The expert study delivers further insights into our travel recommendation algorithm. 29 queries asked for trips with a maximum duration of less or equal than six weeks, 27 queries

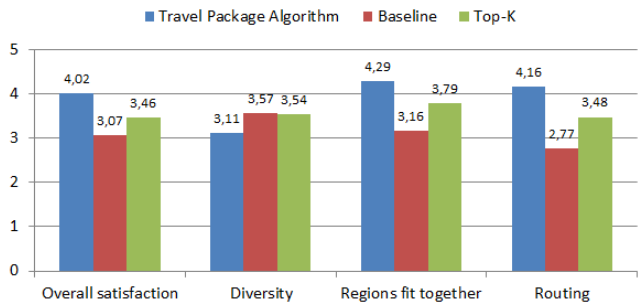

Figure 3: Evaluated travel recommendation algorithms

for more than 6 weeks. For higher durations of stay, the difference in the overall satisfaction with the recommendations is comparable, but our travel recommender algorithm rates better in how regions fit together (∅: 4.41, σ: 0.84) and in terms of the routing (∅: 4.26, σ: 1.06). On the other hand, queries with a higher maximum duration of stay are rated lower in terms of diversity (∅: 3.00, σ: 0.92).

25 queries asked for recommendations with a maximum budget per week of lower or equal than 500 euro. This does not have a significant impact on the quality of the recommendation except in terms of routing which is a bit better for lower budget inputs (∅: 4.28, σ: 0.94) than for higher inputs (∅: 4.06, σ: 0.96).

Our travel recommendation algorithm offers the possibility to limit the recommendation process to preselected regions (step 1 in the process). In the expert study, 16 of the 56 queries had some constraints on the regions. These queries deliver recommendations that satisfied the expert less than average (∅: 3.75, σ: 1.00). Furthermore, single regions fit together less (∅: 3.81, σ: 0.83) and the routing gets worse (∅: 3.75, σ: 1.06) in these cases.

6. CONCLUSION

Recommending composite trips can be understood as a knapsack problem with extensions. Single recommendation items like regions or routes offer value to the user, depending on the similarity between the items and the user query. Multiple regions can be combined to a composite trip which respect the user's limitations in time and money. The total value for the user can not be determined by summing up the single values of all travel stages. Regions in a composite trip are dependent on the presence or absence of other regions in the same recommendation. The distance between regions decreases the possible value for a specified user query because of the costs of the connection.

In this paper, we present a travel recommendation algorithm for composite trips that addresses the Oregon Trail Knapsack Problem and applies the problem to the travel domain. The recommendations generated by our algorithm satisfies an expert user more than comparative algorithms. It is able to combine regions and can determine the optimal duration of stay per region. We showed that the recommended regions fit together and with the availability of connection costs between regions, a decent routing can be ensured. A high maximum duration of stay allows choosing among a higher number of regions and this improves the selection of regions and allows better routing. A low max-

imum budget avoids high connection costs and thus also promises better routing. In terms of diversity, our algorithm performs below average. We integrated mechanisms which penalize long stays in the same region. Nevertheless, the recommended trips offer less diversity than the baseline algorithms. Further effort is necessary in order to understand the preferences of potential users and to find out how a better diversity can be ensured without recommending regions which fit less together. One possibility is extending the penalty function in our algorithm.

Restrictions are useful for users when they want to ignore specific regions in the recommendation process. Reducing the number of regions for the recommendation process improves the algorithm's runtime but also reduces the quality of recommendations. Regional constraints made it more difficult to find regions which match the user's preferences. Moreover, a limited choice of regions makes it more difficult to find regions which fit together and allow a decent routing.

Future work is to extend the system by recommending routes or concrete itineraries in addition to travel regions. In addition, an improved version of the algorithm could also take possible individual activities of travelers such as hiking or shopping into account. In this case, different routes with similar activities could result in additional deduction of the rating and thus reducing the value of this trip. Furthermore, we plan to conduct a more extensive user study with potential users in order to test the recommendations of the travel recommendation algorithm for composite trips.

7. REFERENCES

[1] A. Angel, S. Chaudhuri, G. Das, and N. Koudas. Ranking Objects Based on Relationships and Fixed Associations. In *EDBT '09 Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 910–921, 2009.

[2] J. J. Burg, J. Ainsworth, B. Casto, and S.-D. Lang. Experiments with the "Oregon Trail Knapsack Problem". *Electronic Notes in Discrete Mathematics*, 1:26–35, 1999.

[3] J.-H. Chen, K.-M. Chao, and N. Shah. Hybrid Recommendation System for Tourism. In *2013 IEEE 10th International Conference on e-Business Engineering (ICEBE)*, pages 156–161. Ieee, Sept. 2013.

[4] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *HT '10 Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 35–44, 2010.

[5] Y. Ge, Q. Liu, H. Xiong, A. Tuzhilin, and J. Chen. Cost-aware travel tour recommendation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, pages 983–991, New York, New York, USA, 2011. ACM Press.

[6] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[7] Q. Liu, Y. Ge, Z. Li, E. Chen, and H. Xiong. Personalized Travel Package Recommendation. In *2011 IEEE 11th International Conference on Data Mining*, pages 407–416. Ieee, Dec. 2011.

[8] E. H.-C. Lu, C.-Y. Lin, and V. S. Tseng. Trip-Mine: An Efficient Trip Planning Approach with Travel Time Constraints. In *2011 IEEE 12th International Conference on Mobile Data Management*, pages 152–161. Ieee, June 2011.

[9] F. Ricci, D. Cavada, N. Mirzadeh, and A. Venturini. Case-based travel recommendations. In *Destination recommendation systems: behavioural foundations and applications.*, pages 67–93. CABI, 2006.

[10] F. Ricci, D. R. Fesenmeier, N. Mirzadeh, H. Rumetshofer, E. Schaumlechner, A. Venturini, K. W. Wöber, and A. H. Zins. DieToRecs: a Case-based Travel Advisory System. In *Destination recommendation systems: behavioural foundations and applications.*, pages 227–239. CABI, 2006.

[11] A. Savir, R. Brafman, and G. Shani. Recommending improved configurations for complex objects with an application in travel planning. In *Proceedings of the 7th ACM conference on Recommender systems - RecSys '13*, pages 391–394, New York, New York, USA, 2013. ACM Press.

[12] M. Schumacher and J.-P. Rey. Recommender systems for dynamic packaging of tourism services. In R. Law, M. Fuchs, and F. Ricci, editors, *ENTER*, pages 13–23. Springer Vienna, 2011.

[13] B. Smyth. Case-based recommendation. *The adaptive web*, 4321:342–376, 2007.

[14] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. V. Berghe, and D. V. Oudheusden. a Personalized Tourist Trip Design Algorithm for Mobile Tourist Guides. *Applied Artificial Intelligence*, 22(10):964–985, Oct. 2008.

[15] C. Tan, Q. Liu, E. Chen, H. Xiong, and X. Wu. Object-oriented Travel Package Recommendation. *ACM Transactions on Intelligent Systems and Technology*, 5(3):26, 2014.

[16] P. Vansteenwegen and D. V. Oudheusden. The mobile tourist guide: an OR opportunity. *OR Insights*, 20(3):21–27, 2007.

[17] M. Xie, L. V. Lakshmanan, and P. T. Wood. Breaking out of the box of recommendations: from items to packages. In *Proceedings of the fourth ACM conference on Recommender systems - RecSys '10*, pages 151–158. ACM Press, 2010.