IPS: An Interactive Package Configuration System for Trip Planning

Min Xie
Dept. of Computer Science,
Univ. of British Columbia
minxie@cs.ubc.ca

Laks V.S. Lakshmanan
Dept. of Computer Science,
Univ. of British Columbia
laks@cs.ubc.ca

Peter T. Wood
Dept. of CS and Inf. Syst.,
Birkbeck, U. of London
ptw@dcs.bbk.ac.uk

ABSTRACT

When planning a trip, one essential task is to find a set of Places-of-Interest (POIs) which can be visited during the trip. Using existing travel guides or websites such as Lonely Planet and TripAdvisor, the user has to either manually work out a desirable set of POIs or take pre-configured travel packages; the former can be time consuming while the latter lacks flexibility. In this demonstration, we propose an *Interactive Package configuration System* (IPS), which visualizes different candidate packages on a map, and enables users to configure a travel package through simple interactions, i.e., comparing packages and fixing/removing POIs from a package. Compared with existing trip planning systems, we believe IPS strikes the right balance between flexibility and manual effort.

1. INTRODUCTION

As an essential part of everyone's life, travel is an important way for people to socialize, relax, discover, and explore new cultures. According to a recent report from the U.S. Travel Association [1], the U.S. travel industry plans to create 1.3 million American jobs and add $859 billion to the U.S. economy by 2020. Thus, besides the increasing popularity of existing online travel websites such as Lonely Planet, TripAdvisor, and Yahoo! Travel, many startups such as Tripit have recently been created to help users explore places and trips around the world.

For the same reason, recently a number of research projects have also been initiated around the travel business. For example, using user-uploaded travel-related photos on Flickr, researchers have proposed ways to discover landmarks [19], search landmark images [8], and mine travel patterns [5]. In addition, various algorithms have been developed to mine online user travelogues [7].

In this work, we consider one particularly important task when planning a trip: find the most desirable package of Places-of-Interest (POIs) to be visited during the trip. Previous works on this topic, such as [5] and [14], usually define a user's preferences using *hard constraints*, e.g., a cost budget of $1000 for all the POIs to be visited, and a fixed *utility function*, which is usually a linear combination of utilities of items in the package. These approaches will work when users know *exactly* what they want in a desirable package. However, this may be challenging for the following reasons. Firstly, users often only have a rough idea of what they want in a desirable package. E.g., when summing costs of all POIs, smaller is better; and for the average rating of all POIs, larger is better. Thus hard constraints on a POI feature may result in, either sub-optimal packages when the budget is set too low, or a huge number of candidate packages when the budget is set too high. Secondly, the importance of each criterion specified by the user is usually unknown. E.g., for some users, the monetary budget may not be that important and they can afford to pursue a high quality package while sacrificing a "reasonable" amount of cost budget; whereas other users may be very sensitive to the overall cost of the trip, and may have limited flexibility in terms of monetary budget. We note that the system is informed about the relative importance of the different criteria to the user solely in terms of the weights given to them. However, it is not realistic to expect the user to know (and tell the system) that she/he is 0.8 interested in the overall cost, and 0.2 interested in the overall quality of the package.

To address the first issue, one intuitive solution explored in [17, 10] is first to ask for users' preferences with regard to each criterion, and then present to the user all *skyline packages*, i.e. packages which cannot be *dominated* on every criterion. However, the number of such skyline packages, as shown in the empirical results of [17] and [10], can be in the hundreds or even thousands for a reasonably sized dataset. So presenting all of these packages to a user is impractical.

Instead, following recent work on multi-dimensional ranking of items [16], we take a *quantitative* approach to rank packages based on multiple criteria, i.e., we consider that for each traveler u, there is an intuitive "implicit" linear utility function f_u, which depicts u's preference or trade-off over different criteria for configuring a desirable package. By leveraging this utility function, we can easily rank all packages, and present the best one to the user. E.g., for a user u who has equal preference on cost budget and average item rating of a package, we can use the function $f_u(p) = 0.5SUM(p_{cost}) + 0.5AVG(p_{rating})$, where $SUM(p_{cost})$ is the cost of a package p, and $AVG(p_{rating})$ is the average item rating of p.

However, as mentioned in the second issue above, one challenge for this approach is that we cannot assume that

Proceedings of the VLDB Endowment, Vol. 6, No. 12

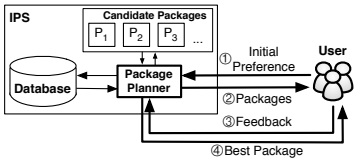

Figure 1: Architecture of IPS.

the user will define the utility function exactly for the system. Thus, similar to some recent work such as [13], [12], and [11], *we will assume the weights of the utility function are hidden* and take an interactive approach to obtain these weights from user feedback.

Specifically, our proposed system initially assumes each user has equal preference over each desired criterion. Then the system iteratively recommends to the user a small number of packages, and the user can select any of her/his favorite packages from among them. These packages are chosen by the system to learn more about the user, and thus enable the system to recommend better packages as time progresses. A key attractive property of our approach is that while the interactive process needs to exploit a user's preferences by suggesting packages, every suggested package is also a *recommendation* to the user based on the current inferred knowledge; this can help the system keep the user engaged through the interaction process.

In addition to packages which are recommended in each iteration, the system also has a separate component which keeps and presents to the user a set of current best packages according to the partially inferred utility function, and the user can directly select any of these to inform the system that the shown package is indeed desirable compared with other "best" packages presented. As we shall discuss in Section 2, every such feedback from the user can also help the system refine the user's utility function.

Compared with [12] and [11] which consider preferences over *items*, our package configuration problem needs to explore a much larger space of all possible packages. In addition, the aggregation-based feature space (e.g., cost budget) further complicates the underlying problem. On the other hand, [13] also considers an interactive way of ranking travel packages. However, the user feedback model in [13] is defined in such a way that for each iteration, the user is asked to rank a set of POIs instead of packages. Thus every decision the user makes is "local" in the sense that the user is not able to personalize her/his preference over *packages* as a whole; it is possible that POIs favored in one iteration might become less desirable after seeing some additional POIs.

Based on the above idea, in this work, we propose an *Interactive Package configuration System* (IPS) which can be utilized to help the user make POI decisions upon planning a trip. Compared with existing work with a similar purpose, IPS has to address the following challenges: 1) Unlike previous work which is either concerned with items, or considers packages of fixed size [17, 10], IPS considers packages which, if considered fully, may require a much larger search space; 2) Instead of assuming an exact utility function to be specified by a user, which is not realistic, IPS adopts an interactive way to explore a large space of utility functions by asking for users' preferences among a small set of packages. Because of the large space of packages, picking the right packages to present to the user is extremely critical.

In the rest of this proposal, we first formulate the interactive package configuration problem, and discuss our proposed algorithms and strategies for solving it (Section 2). Then in Section 3, we describe our demonstration proposal, and explain how the audience can interact with IPS. Finally, we discuss related work in Section 4.

2. PROBLEM SETTING

2.1 System Architecture

The basic framework of IPS is depicted in Figure 1. As can be seen from this figure, a user u of IPS first selects from a predefined list of package features those that she/he is interested in, and the preferred order for each aggregate value. E.g., u can select the cost budget, and indicate that the smaller the cost budget, the better. Then the system will iteratively recommend small batches of packages to u which exploit the user's preferences. User u can select from these packages those she/he likes. Meanwhile, in a separate panel, the user can also interact with the system at any time by noting that a current candidate best package shown to her/him in this panel is preferred, or the user can also inform the system that a particular POI must be included in or excluded from the final package (see Figure 2). The system will keep refining its knowledge about the user through investigating the received feedback, and the user can stop the interactive process at any moment when a desirable package has been found.

2.2 Problem Definition

Consider a set R of n POIs, $R = \{s_1, \ldots, s_n\}$. If $\mathcal{F} = \{f_1, ..., f_o\}$ denotes the universe of all possible features, then each POI $s \in R$ is associated with a subset F_s of features, $F_s \subseteq \mathcal{F}$. Similar to previous work [17], we assume all feature values are non-negative rational numbers. We use $f(s)$ to denote the value of POI s on feature f, while $f(s) = nil$ if f is not defined on s.

Packages of POIs can be defined as subsets of R, e.g., $p = \{s_1, s_2, s_3\}$, $s_1, s_2, s_3 \in R$, is a package of three POIs. We define the feature set F_p of a package p as the union of the feature sets of POIs within p, e.g., for the example package p, $F_p = F_{s_1} \cup F_{s_2} \cup F_{s_3}$. Since a feature $f \in F_p$ may not be available for each POI $s \in p$, we define the *affiliated poi-set* of f in p as $p_f = \{s \mid f(s) \neq nil\}$.

As has been discussed in existing work on package configuration [17, 10], a user's preference over packages is usually based on *aggregations* over feature values of POIs within the package. E.g., the sum of the costs of all POIs might define the overall cost of a package, while the average of the ratings of all POIs might define the overall quality of a package. Thus we define the *Aggregate Feature Profile* (or simply *profile*) of a package as follows. We note that for the current system, we do not take order of items in a package into consideration, since this usually leads to problems of much higher complexity [6, 15].

DEFINITION 1. *(Aggregate Feature Profile) The aggregate feature profile of a package p is defined as $V_p = \{\mathcal{A}_1(p_{f_1}), \ldots, \mathcal{A}_m(p_{f_m})\}$, where $\{f_1, \ldots, f_m\} \subseteq \mathcal{F}$, and each $\mathcal{A}_i$, $1 \leq i \leq m$, is one of the aggregation functions MIN, MAX, SUM, or AVG.*

We call $V = \{(\mathcal{A}_1, f_1), \ldots, (\mathcal{A}_m, f_m)\}$ the *package profile schema*. Note that given a fixed set of POIs, we can easily calculate the maximum aggregate value for a feature that can be achieved by any package. E.g., for $MAX(p_{f_1})$, the maximum value on f_1 that can be achieved by any package

is simply the maximum f_1 value from all the POIs. So we normalize each aggregate value in a profile using the maximum possible aggregate value of the corresponding feature. Thus $0 \leq \mathcal{A}_i(p_{f_i}) \leq 1$, $1 \leq i \leq m$.

Given a set $\mathcal{P}$ of packages, following previous work [16], we model a user's preference over $\mathcal{P}$ using a *utility function* g, which is a weighted linear combination of aggregate feature values in a profile V_p.

$$g(V_p) = w_1\mathcal{A}_1(p_{f_1}) + \ldots + w_m\mathcal{A}_m(p_{f_m}) \quad (1)$$

If a user prefers smaller values to larger ones for an aggregate feature $(\mathcal{A}_i, f_i)$ in the package profile schema, we can modify the corresponding aggregate feature to be $(1-\mathcal{A}_i, f_i)$. Thus, in the following discussion, we assume w.l.o.g. that the user prefers larger values for each aggregate feature, and also that $\sum_i w_i = 1$, $0 \leq w_i \leq 1$, $1 \leq i \leq m$.

A framework based on a utility function essentially defines a total order over all packages, so is different from [17, 10] which aim to return all the *skyline packages*, the number of which can be prohibitively large.

Despite being intuitive, there are two major challenges in adopting the utility function based framework for package configuration in practice. First, users are usually not able to specify the exact weights of the utility function g. Thus we need to have interactive mechanisms which enable us to elicit users' preferences over weight values. Second, unlike [17] and [10] which consider packages of fixed size, we allow package size to be flexible in our framework. E.g., given a system-defined maximum package size of say 20, we consider all possible package sizes ranging from 1 to 20.

Clearly the core problem in IPS is to infer the utility function of the user through feedback. We formulate this problem in Definition 2.

DEFINITION 2. **Preference Elicitation***: Given the current value range of each weight in the utility function g, and a constant k, find the best set $\mathcal{P}_k$ of k packages to present to the user, such that the system can gain the maximum expected amount of knowledge about the utility function g.*

By refining the partial knowledge about g through iterative interaction (details will be presented in Section 2.3), the goal of IPS is to find for the user the most desirable package.

DEFINITION 3. **Best Package Search***: Given a package profile schema V and partial knowledge about the utility function g, find the most desirable package $p^* \in \mathbf{P}$, where $\mathbf{P}$ is the set of all possible packages, such that $\forall q \in \mathbf{P} - \{p^*\}$, $\underline{g}(p^*) \geq \overline{g}(q)$, where $\underline{g}(p)/\overline{g}(p)$ denotes the minimum/maximum possible score for a package p w.r.t. g.*

2.3 Algorithms

Initially, when the system knows nothing about the hidden utility function, clearly the space of all possible combinations of weights forms an m-dimensional hyper-cube $[0, 1]^m$.

To solve the preference elicitation problem of Definition 2, we first consider the simple case where the system presents two packages for the user to review in each iteration. Let the two packages be p_1, p_2. Clearly, without any further information from the user, we can only assume that each package has an equal chance to be preferred by the user. Without loss of generality, assume the user picks p_1. This feedback indicates that for the utility function g, $g(V_{p_1}) - g(V_{p_2}) \geq 0$. Thus we can infer that $g(V_{p_1}) - g(V_{p_2}) = 0$ defines an m-dimensional *decision-boundary* which is a hyperplane and separates the desirable weight space from the undesirable weight space. In the following iterations, we only need to look at weights which come from the desirable weight space. We note that if the user chooses both or neither of the two packages recommended, we can simply ignore the current feedback and try to recommend two new packages.

But what is the guideline for choosing the best two packages in each iteration? Ideally the best two packages should be selected in a way such that the the desirable weight space left after each iteration is minimized. Assuming each package has an equal chance to be selected, it can be shown that we should select packages such that the decision-boundary divides the current weight space into two halves of equal volume. Then by implementing an efficient algorithm for finding candidate packages which can potentially become the best package given a specified weight space, IPS can iteratively apply the above idea to interact with the user and refine the weight space. The algorithm stops when it can already determine the most desirable package using the refined weight space.

We note that there are two additional issues with the above solution. Firstly, arbitrarily chosen packages may result in irregularly shaped desirable weight spaces, which in turn will increase the complexity of the remaining computation. Thus in IPS we only consider packages such that the resulting refined weight spaces are hyper-rectangles. Secondly, in order to satisfy the above properties, IPS may have to show the user "fake" packages. But in order to guarantee a good user experience, similar to the *truthfulness* property discussed in [12], IPS will ensure that whenever fake packages are presented to the user, there exist actual packages which either are equal to or can dominate the fake ones with respect to quality.

Finally, consider the case of showing more than two packages in each iteration. The decision-boundary is no longer a hyperplane, but instead is formed by the *Voronoi boundary* which separates each package profile from the others. To ensure that the refined space after the user's feedback is still a hyper-rectangle, it can be shown that we need to present 2^i packages in each iteration, where i is a small integer. Note that showing more packages in each iteration can result in a much smaller refined weight space after each iteration, but the price will be that users need to make more complex decisions in each iteration.

3. DEMO SCENARIO

For the implementation of IPS, we crawled Yelp for real ratings, monetary cost, and location information of POIs from the most visited cities in North America. For time duration (cost) information, we use synthetic data by making a simple assumption that the time spent on each POI is proportional to its area/size (this information can be crawled from Wikipedia), and multiplied by a factor which depends on the type of the POI. An alternative way to get time cost information is to analyze online user-generated content as described by [5].

IPS has two major components: (1) initial preference customization interface, and (2) interactive package planner.

Initial Preference Customization Interface: Through this interface, users of IPS can specify the travel destination and customize the aggregation functions to be used in the package profile schema. E.g., for the rating of items in the package, the user can prefer either larger minimum rating or larger average rating; similarly, for monetary cost of the

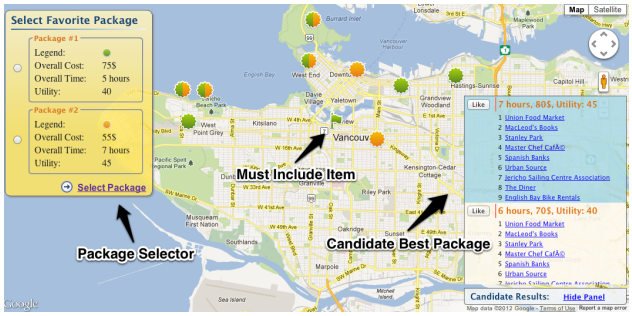

Figure 2: The main interface for reviewing packages in IPS.

package, the user can prefer either smaller maximum cost or smaller total cost. Optionally, users can turn on or off a specific type of POI, e.g., parks, to be considered by IPS.

Interactive Package Planner: Once a user has specified her/his initial preference, IPS will start the interactive package planner, the interface of which is shown in Figure 2. There are three types of interactions that are supported in IPS. (1) The system presents a small set of packages to the user, which are visualized on the map interface, and POIs from different packages can be distinguished using different colors. Then the user can select her/his favorite package from the list, which enables IPS to gain further information about the hidden utility function of the user. (2) Meanwhile, the user can click on each POI shown on the map to obtain its detailed description. If the user really likes/hates a specific POI, she/he can tell IPS this, so that the corresponding POI will be included/excluded from all future package configurations. A list of POIs which have been included/excluded is shown on the map interface using unique markers, and the user can choose to change this behavior by clicking on them. (3) Finally, a list of current best packages is also shown on the interface, and the user can select any of them as favorite to let the system gain further information about the hidden utility function, and also find candidate packages which can beat this favored package.

4. RELATED WORK

There has been much investigation into handling preferences of items in the database community, e.g., general preference frameworks [9, 4], skyline queries [2], and top-k queries [16]. However only recently have researchers started considering preference handling for sets of items. Some initial work on this by AI researchers (e.g., see [3]) typically focuses on the formal aspects of this problem, e.g., expressiveness of the preference language. Unfortunately, the proposed preference model is often not suitable in practice. In [14], we model preferences on packages using hard constraints and a fixed score function, thus turning the problem of finding the top preferable sets into an optimization problem. A similar approach has also been studied in [5]. However, hard constraints may not be intuitive in practice, since users are often flexible when considering budgets, and may be willing to trade costs for better result quality. Recent work in [17] and [10] represents an alternative approach of finding all skyline packages of fixed cardinality. However, as mentioned in the introduction, a severe drawback of this approach is that the number of skyline packages usually tends to be prohibitive.

In IPS we adopt a more practical framework of modeling set preferences using a personalized utility function. However, unlike previous work in top-k query processing [16], we assume that the parameters of the utility function are unknown, and they need to be inferred from online user feedback. Compared with existing work on interactive preference elicitation [12, 18, 11], our search space of all candidate packages is much larger, and we need to consider features which are based on complex aggregations of item attribute values. Finally, the motivation for IPS is similar in spirit to [13]; however, the feedback model in [13] is based on items, instead of packages as a whole, and so may result in sub-optimal decisions for the user.

5. REFERENCES

[1] `http://www.ustravel.org/news/press-releases/us-travel-industry-reveals-plan-create-13-million-american-jobs`.

[2] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, pages 421–430, 2001.

[3] G. Brewka, M. Truszczynski, and S. Woltran. Representing preferences among sets. In *AAAI*, 2010.

[4] J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.

[5] 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*, pages 35–44, 2010.

[6] M. De Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. Automatic construction of travel itineraries using social breadcrumbs. In *ACM Hypertext*, pages 35–44, 2010.

[7] Q. Hao, R. Cai, C. Wang, R. Xiao, J.-M. Yang, Y. Pang, and L. Zhang. Equip tourists with knowledge mined from travelogues. In *WWW*, page 401, 2010.

[8] L. S. Kennedy and M. Naaman. Generating diverse and representative image search results for landmarks. In *WWW*, pages 297–306, 2008.

[9] W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.

[10] C. Li, N. Zhang, N. Hassan, S. Rajasekaran, and G. Das. On skyline groups. In *CIKM*, 2012.

[11] D. Mindolin and J. Chomicki. Discovering relative importance of skyline attributes. *PVLDB*, 2(1):610–621, 2009.

[12] D. Nanongkai, A. Lall, A. D. Sarma, and K. Makino. Interactive regret minimization. In *SIGMOD*, pages 109–120, 2012.

[13] S. B. Roy, G. Das, S. Amer-Yahia, and C. Yu. Interactive itinerary planning. In *ICDE*, pages 15–26, 2011.

[14] M. Xie, L. V. S. Lakshmanan, and P. T. Wood. Breaking out of the box of recommendations: From items to packages. In *RecSys*, page 151. ACM, 2010.

[15] M. Xie, L. V. S. Lakshmanan, and P. T. Wood. Comprec-trip: A composite recommendation system for travel planning. In *ICDE*, pages 1352–1355, 2011.

[16] A. Yu, P. K. Agarwal, and J. Yang. Processing a large number of continuous preference top-k queries. In *SIGMOD*, pages 397–408, 2012.

[17] X. Zhang and J. Chomicki. Preference queries over sets. In *ICDE*, pages 1019–1030, 2011.

[18] F. Zhao, G. Das, K.-L. Tan, and A. K. H. Tung. Call to order: A hierarchical browsing approach to eliciting users preference. In *SIGMOD*, pages 27–38, 2010.

[19] Y. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *CVPR*, pages 1085–1092, 2009.