CompRec-Trip: a Composite Recommendation System for Travel Planning

Min Xie †, Laks V.S. Lakshmanan †, Peter T. Wood ‡

†*Department of Computer Science, University of British Columbia*
{minxie,laks}@cs.ubc.ca
‡*Department of Computer Science and Information Systems, Birkbeck, University of London*
ptw@dcs.bbk.ac.uk

***Abstract*—Classical recommender systems provide users with a list of recommendations where each recommendation consists of a single item, e.g., a book or a DVD. However, applications such as travel planning can benefit from a system capable of recommending packages of items, under a user-specified budget and in the form of sets or sequences. In this context, there is a need for a system that can recommend top-k packages for the user to choose from. In this paper, we propose a novel system, CompRec-Trip, which can automatically generate *composite recommendations* for travel planning. The system leverages rating information from underlying recommender systems, allows flexible package configuration and incorporates users' cost budgets on both time and money. Furthermore, the proposed CompRec-Trip system has a rich graphical user interface which allows users to customize the returned composite recommendations and take into account external local information.**

I. Introduction

Recommender systems (RS) are increasingly popular and have become an essential driver of many applications including web services [1]. Classical RS provide recommendations consisting of single items, e.g., books or DVDs. However, there are several applications that can benefit from a system capable of recommending packages of items [2].

In travel planning, a user is interested in suggestions for places to visit, or places of interest (POI). There may be a cost to visiting each place (time, price, etc.) which the user may want to constrain with a budget. Optionally, and independently of the budget, there may be a notion of *compatibility* among POIs in a package, modeled in the form of constraints: e.g., "the total distance covered in visiting all POIs in a package should be $\leq$ 10 km.", "no more than three museums in a package", "not more than two parks", etc. The user may have a limited budget and is interested in suggestions of compatible packages of POIs such that the cost of each package is under budget and its value (as judged from ratings) is as high as possible. In these applications, there is a natural need for top-k recommended packages for the user to choose from. Some so-called "third generation" travel planning web sites, such as NileGuide[1] and YourTour[2], are starting to provide some of these features, although in a limited form.

Motivated by such applications, we propose a novel system called CompRec-Trip which can automatically generate *composite recommendations* for travel planning, where each recommendation comprises a set or sequence of POIs. Each POI has both a value (rating or score) and a cost associated with it, and the user specifies a maximum total cost (budget) for any recommended package of POIs. Our CompRec-Trip system consists of one or more recommender systems focusing on different domains (e.g., hotels, museums, and events). These component RS serve (i.e., recommend) top POIs in non-increasing order of their value (explicit or predicted ratings). In addition, the CompRec-Trip system also has access to information sources (which could be databases or web services) which provide the cost associated with each POI.

In our setting, the problem of generating the top recommendation (package) is NP-complete as it models either the Knapsack problem [3] for packages as sets or the Orienteering problem [4] for packages as sequences. Because of this and the fact that we expect the component recommender systems to provide ratings for large numbers of POIs and access to these ratings can be relatively expensive[3], we devise approximation algorithms for generating top-k packages as recommendations.

Compared with previous work on considering complex or composite recommendations[4], our system has the following distinguishing features: Package generation is based on ratings from recommender systems; Allows flexible package configuration; Captures users' budgets on both time and money; Allows flexible resulting package customization by adding/removing POIs; Allows users to incorporate local information from websites such as Yahoo! Upcoming.

II. System Architecture

In a traditional RS, users rate items based on their personal experience, and these ratings are used by the system to predict ratings for items not rated by an active user. The predicted ratings can be used to give the user a ranked recommendation (item) list.

As shown in Figure 1, our composite recommendation system is composed of one or more component RS and has access to external sources that provide the cost of given items. External sources can be local databases or web services. E.g., Yelp.com can be consulted for restaurants' prices, and Google Map can be consulted for distances between places. In terms of computation, we abstract each RS as a system which serves

[1] http://www.nileguide.com
[2] http://www.yourtour.com

[3] Especially when the ratings need to be predicted.
[4] A detailed comparison can be found in [2].

items in non-increasing order of their value (rating or score) upon request. In addition, the system includes a compatibility checker module, which checks whether a package satisfies compatibility constraints, if any. The compatibility checker consults necessary information sources in order to verify compatibility.

The user interacts with the system by specifying cost budgets, an integer k, and optionally compatibility constraints on packages. The system finds the top-k packages of items with the highest total value such that each package has a total cost under budget and is compatible.

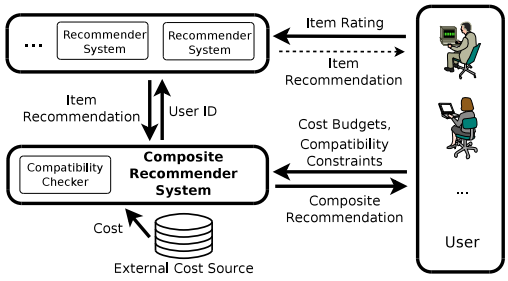

Fig. 1. System Architecture

III. Problem Statement

In this paper, for simplicity, we focus on the composite recommendation problem with one component RS and without compatibility constraints; extensions will be discussed in Section IV-D.

Given a set N of POIs, a set U of users, an active user $u \in U$, and a POI $t \in N$, we denote by $v_u(t)$ the *value* of POI t for user u. We denote the value as $v(t)$ when the active user is understood. A RS predicts $v(t)$ when it is not available, by using the active user's past behaviour and possibly that of other similar users. For $t \in N$, we denote by $tc(t)$ the *time cost* and by $mc(t)$ the *monetary cost*, of POI t. Given a set of POIs $R \subseteq N$, we define $v(R) = \Sigma_{t \in R}\, v(t)$, $tc(R) = \Sigma_{t \in R}\, tc(t)$ and $mc(R) = \Sigma_{t \in R}\, mc(t)$.

Definition 1: **Top-k Composite Set Recommendations:** Given an instance I of a composite recommendation system consisting of one component RS and an external information source, a cost budget B_t on time, a cost budget B_m on money and an integer k, find the top-k packages $P_1, ..., P_k$ such that each P_i has $mc(P_i) \leq B_m$, $tc(P_i) \leq B_t$, and among all feasible packages[5], $P_1, ..., P_k$ have the k highest total values, i.e., $v(P) \leq v(P_i)$ for all feasible packages $P \notin \{P_1, ..., P_k\}$.

When $k = 1$, the top-k composite set recommendation problem (CompRec-Set) can be viewed as a variation of the classical 0/1 2-dimensional knapsack problem [3] with the restriction that POIs can be accessed only in non-increasing order of their value. Without loss of generality, we assume all POIs have (time and money) costs smaller than the cost budgets B_t and B_m.

Note that ratings of POIs from the component RS are retrieved using sorted access, while the cost of a given POI is obtained via random access. Let c_s and c_r be the costs associated with these accesses. Then the total *access cost* of processing n POIs is $n \times (c_s + c_r)$. Notice that the number of POIs in the system is often huge, and c_s and c_r can be large compared to the cost of in-memory operations, as often for both accesses, information may need to be transmitted through the Internet.

So, well-known algorithms for the knapsack problem which need to access *all* POIs [3] may not be realistic. Thus, an efficient algorithm for top-k CompRec-Set should minimize the total access cost, i.e., it should minimize the number of POIs accessed.

In our system, we also assume some background cost information about each POI is available. The background cost information can be a histogram $\mathcal{H}$ collected from the cost database or just the minimum POI monetary (time) cost mc_{min} (tc_{min}). This information can be materialized in our system and be refreshed regularly by rescanning the cost database. We denote the background cost information as $\mathcal{BG}$.

In addition to the cost associated with each POI, we may also need to consider the cost spent on traveling to corresponding POIs. For each POI pair (t_1, t_2), $t_1, t_2 \in N$, we denote by $d(t_1, t_2)$ the shortest distance between t_1 and t_2. And given a set of POIs $R \subseteq N$, we define $w(R)$ as the minimum distance walk which covers all POIs in R, and let $tcw(R)$ and $mcw(R)$ be the corresponding time and monetary cost for taking $w(R)$ by assuming an average speed/cost per unit of distance.

Definition 2: **Top-k Composite Sequence Recommendations:** Given an instance I of a composite recommendation system consisting of one component RS and an external information source, a cost budget B_t on time, a cost budget B_m on money and an integer k, find the top-k packages $P_1, ..., P_k$ such that each P_i has $mc(P_i) + mcw(P_i) \leq B_m$, $tc(P_i) + tcw(P_i) \leq B_t$, and among all feasible packages, $P_1, ..., P_k$ have the k highest total values, i.e., $v(P) \leq v(P_i), 1 \leq i \leq k$ for all feasible packages $P \notin \{P_1, ..., P_k\}$.

When $k = 1$, the top-k composite sequence recommendation problem (CompRec-Seq) can be viewed as a variation of the orienteering problem [4] with the restriction that POIs can be accessed only in non-increasing order of their value. Similar to the composite set recommendation problem, we assume we have some simple background cost information such as the minimum shortest distance d_{min} between POIs.

IV. Implementation

A. Composite Set Recommendation

For the composite set recommendation problem, we can adopt the algorithm template studied in our previous work [2], which is shown in Algorithm 1.

In this algorithm template, one POI is retrieved from an underlying RS at each iteration (lines 3–4). After accessing this new POI, we can use a pseudo-polynomial time algorithm [3] to find an optimal solution R^o *over the accessed POI-set* S (line 5). We can also derive a *tight upper bound*[6] V^* of the true optimal solution from the calculation of R^o

[5]We say a package if feasible if it satisfies all the budgetary constraints.

[6]An upper bound is tight iff there exists a possible unseen POI configuration for which the upper bound is achievable by using a subset of accessed POIs and valid unseen POIs. Here, an unseen POI t is valid iff $v(t) \leq v_{min}$, $tc(t) \geq tc_{min}$ and $mc(t) \geq mc_{min}$.

using MaxValBound [2] (line 6). The algorithm terminates if $v(R^o) \geq \frac{1}{\alpha} \times V^*$; if not, it continues to access the next POI (lines 7–8) [7].

Algorithm 1: CR-Set-Top1(N, B_t, B_m, $\mathcal{BG}$)

```
1 S ← An empty buffer
2 while TRUE do
3     t ← N.getNext()
4     S.Insert(t)
5     R^o ← Optimal2DKP(S, B_t, B_m)
6     V* = MaxValBound(S,BG)
7     if v(R^o) ≥ 1/α × V*
8         return R^o
```

Similar to [2], it can be shown that the proposed CR-Set-Top1 is both correct and *instance optimal* [5]. However, CR-Set-Top1 relies on an exact algorithm for the knapsack problem which may lead to high computational cost. To remedy this, we can develop more efficient heuristic algorithms but may have to give up instance optimality. The idea is that instead of using an exact algorithm to get the best package for the currently accessed set of POIs S, we can use a simple greedy heuristic to form a high quality package R^G from S and then test whether R^G is globally a high quality package. We have shown in previous work [2] that algorithms using the greedy heuristic often have performance very close to the instance optimal algorithm.

B. Composite Sequence Recommendation

Composite sequence recommendation is closely related to the orienteering problem [4], which seeks a maximum value walk on a graph subject to a budget constraint on the cost. To simplify the presentation, we will ignore discussing cost on each POI (i.e., the so-called node cost) as this can be handled by a reduction to the original orienteering problem [6].

Similar to the composite set recommendation problem, to minimize the number of POIs retrieved while having the result quality guaranteed, we need to adapt the algorithm template as described by Algorithm 1 and iteratively calculate the optimal solution for the accessed POI-set S along with a *tight upper bound* on the possible true optimal solution to the underlying composite sequence recommendation problem instance.

Given an (exponential-time) exact orienteering solver, we can calculate the optimal solution for the subgraph G of the original POI graph, induced by the accessed POI-set S. However, it is more challenging to get a *tight* upper bound on the value of the true optimal solution for the composite sequence recommendation problem.

Let d_{min} be the background cost information about the minimum distance between POIs, $v_{min} = \min_{t \in S} v(t)$ and t^* be an "imaginary" unseen POI which has value v_{min}. A tight upper bound on the possible true optimal solution can be computed by the procedure MaxOriValBound shown in Algorithm 2.

It can be proven that, using procedure MaxOriValBound, we can give a correct instance-optimal α-approximation algorithm CR-Seq-Top1 for the composite sequence recommendation problem. Similar to composite set recommendation, to get better practical performance, we can utilize approximation algorithms for the orienteering problem such as [4] instead of exact algorithms. However, the resulting algorithm will not be instance optimal.

Algorithm 2: MaxOriValBound(G,B_t,B_m,$\mathcal{BG}$)

```
1 foreach Edge (v1, v2) ∈ G do
2     n* = ⌊ d(v1,v2) / d_min ⌋
3     Add to G a path P between v1 and v2 which is composed of n*
      "imaginary" POIs, each a copy of t*, and d(P) = d(v1, v2)
4 S = OptimalOrienteer(G,B_t,B_m)
5 Augment S with "imaginary" POIs t* if possible
6 return v(S)
```

C. Top-k Composite Recommendation

In addition to the best composite recommendation, it is often useful to provide the user with the top-k composite recommendations, where k is a small constant. Similar to the top-1 case, due to the hardness of the underlying problem, we seek an efficient approximation algorithm which can give us high quality recommendations.

Given an instance I of the top-k composite recommendation problem, assume $\mathcal{R}^I$ is the set of all *feasible composite recommendations*, i.e., $\mathcal{R}^I = \{R \mid R \subseteq N \wedge mc(R) \leq B_m, tc(R) \leq B_t\}$ for composite set recommendation and $\mathcal{R}^I = \{R \mid R \subseteq N \wedge mc(R) + mcw(R) \leq B_m, tc(R) + tcw(R) \leq B_t\}$ for composite sequence recommendation. Following Fagin et al. [5], we define an α-approximation of the top-k composite recommendations to be any set $\mathcal{R}_k$ of min(k, $|\mathcal{R}^I|$) composite recommendations, such that, for all $R \in \mathcal{R}_k$ and $R' \in \mathcal{R}^I \backslash \mathcal{R}_k$, $v(R) \geq \frac{1}{\alpha} \times v(R')$.

Lawler's procedure [7] is a general technique for enumerating optimal top-k answers to an optimization problem, which relies on an efficient algorithm to find the optimal solution to the problem. We have proven in our previous work [2] that for the top-k composite recommendation problem, if the algorithm for the optimal solution in Lawler's procedure is replaced by an α-approximation algorithm, instead of optimal top-k answers, we get an α-approximation to the optimal set of top-k composite recommendations. So in our system we leverage Lawler's procedure and CR-Set-Top1/CR-Seq-Top1 to produce top-k approximate composite recommendations.

D. Discussion

To capture further constraints such as "not more than two parks" in our algorithms, we can define a Boolean *compatibility function* $\mathcal{C}$ over the packages under consideration. Given a package P, $\mathcal{C}(P) = true$ iff all constraints on POIs in P are satisfied. We can add a call to $\mathcal{C}$ in the previously proposed algorithms after each candidate package has been found. If the package fails the compatibility check, we just discard it and search for the next candidate package.

It is worth noting that the Boolean compatibility function defined here allows for great generality. However, depending on the application needs, for scenarios where only one specific type of constraint is considered, e.g., having one POI from

[7] Decreasing the approximation factor α can lead to higher quality packages; however, it will also result in higher computational cost. In our system, we set $\alpha = 2$.

each of three predefined categories, more efficient algorithms like Rank Join [8] can be leveraged.

Furthermore, although in the previous algorithms we assume there is only one component RS, it is straightforward to combine recommendation lists from several component RS into a single list, based on ratings.

E. Datasets

For the implementation of our system, ratings, monetary cost and location information are crawled from Yelp.com. For time cost information, we assume that the time spent on each POI should be proportional to its area/size (this information is crawled from Wikipedia), and multiplied by a factor which depends on the type of the POI. For example, parks are often larger in area than museums; however, visitors tend to spend more time in museums per unit area, so we may apply a larger factor to museums to take this into consideration. For POIs such as restaurants, we assume a default time cost. An alternative way to get time cost information is to analyze online user-generated content as described by [6].

We can obtain all the distance information using the geographic coordinates of POIs crawled from Yelp.com. We can also obtain better quality distance information from the Google map service. To estimate the time/money to be spent on the trip, depending on which transportation method the user chooses, we assume an average speed/cost for the travel.

V. Graphical User Interface

Compared with existing trip recommendation systems, one of the highlights of our CompRec-Trip system is that users have great power in interacting with the recommended packages and in further tuning of the results. An illustration of the system interface is shown in Figure 2. As can be seen, there are three major components of our system: 1. Destination and customization interface; 2. Recommendation list; 3. Package contents view.

1. Destination and Customization Interface: Users of CompRec-Trip can get composite set or sequence recommendations through the top tab bar by specifying the travel destination, cost budgets, number k of top packages to be generated and also various other features including: preferences about POIs such as number of different types of POI to be included, and the method of transport. Furthermore, from the tab bar, users can choose to include the latest updates from external websites such as Yahoo! Upcoming.

2. Recommendation List: Once a user has (fully or partially) specified the travel targets and preferences about the trip, the CompRec-Trip system automatically generates the top-k composite recommendations. The recommended packages will be listed on the right-hand panel, and the user can click on each to get a live view of the package on the left hand map interface. If the user is not satisfied with the recommended packages, he/she can click the "Add POI (Remove POI)" button on the current active package. A local search interface will be shown and the user can use this to filter POIs he/she is (is not) interested in. Any modification to the package takes effect immediately and the change is reflected on either the current

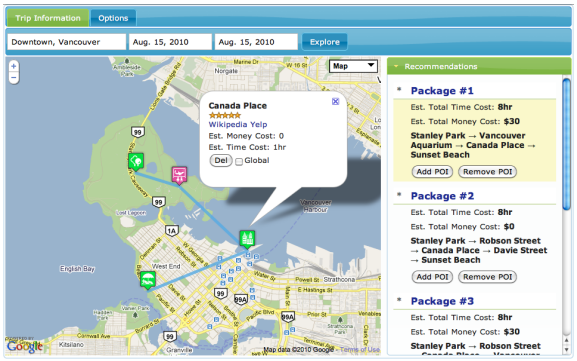

Fig. 2. Graphical User Interface

active recommendation or all recommendations, depending on whether the user wants to make this fix global.

3. Package Contents View: A large part of the user interface is occupied by the visualization of the current active recommended package. A map control is shown according to the selected package, the contents of the package are visualized on the map and nearby interesting POIs or updates from external information sources can be shown as an option. The user can click on each of the POIs contained in the package to get a brief overview of this POI and obtain links for getting further information. Also the user can interact with this package by removing some POIs from the package or clicking on the map to search for some other local interesting POIs nearby, which she can add to the current package.

VI. Demonstration

Our demo will show how top-k composite set/sequence recommendations will be constructed by specifying various preferences such as cost budgets, transportation method and other constraints. We will visualize each package on a map interface and show how users can interact with the resulting packages by adding and removing POIs. Furthermore, we will also show how external local information can be seamlessly incorporated into the recommendation results and be used to help users explore local interesting activities. The importance of efficient top-k composite recommendation algorithms along with incremental computing will be highlighted.

References

[1] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE TKDE*, vol. 17, no. 6, pp. 734–749, 2005.

[2] M. Xie, L. V. S. Lakshmanan, and P. T. Wood, "Breaking out of the box of recommendations: From items to packages," in *ACM RecSys*, 2010, pp. 151–158.

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

[4] C. Chekuri and M. Pál, "A recursive greedy algorithm for walks in directed graphs," in *FOCS*, 2005, pp. 245–253.

[5] R. Fagin, A. Lotem, and M. Naor, "Optimal aggregation algorithms for middleware," *J. Comput. Syst. Sci.*, vol. 66, no. 4, pp. 614–656, 2003.

[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*, 2010, pp. 35–44.

[7] E. L. Lawler, "A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem," *Man. Sci.*, vol. 18, no. 7, pp. 401–405, 1972.

[8] J. Finger and N. Polyzotis, "Robust and efficient algorithms for rank join evaluation," in *ACM SIGMOD*, 2009, pp. 415–428.