A package recommendation framework based on collaborative filtering and preference score maximization

Panagiotis Kouris[1] [2], Iraklis Varlamis[2], and Georgios Alexandridis[1]

[1] School of Electrical and Computer Engineering, National Technical University of Athens, Greece
{pkouris,gealexandri}@islab.ntua.gr
[2] Department of Informatics and Telematics, Harokopio University of Athens, Greece
varlamis@hua.gr

Abstract. The popularity of recommendation systems has made them a substantial component of many applications and projects. This work proposes a framework for package recommendations that try to meet users' preferences as much as possible through the satisfaction of several criteria. This is achieved by modeling the relation between the items and the categories these items belong to aiming to recommend to each user the top-k packages which cover her preferred categories and the restriction of a maximum package cost. Our contribution includes an optimal and a greedy solution. The novelty of the optimal solution is that it combines the collaborative filtering predictions with a graph based model to produce recommendations. The problem is expressed through a minimum cost flow network and is solved by integer linear programming. The greedy solution performs with a low computational complexity and provides recommendations which are close to the optimal solution. We have evaluated and compared our framework with a baseline method by using two popular recommendation datasets and we have obtained promising results on a set of widely accepted evaluation metrics.

Keywords: Recommendation system; Package recommendations; top-k packages; Collaborative filtering

1 Introduction

Recommendation systems (RSs) have become popular since they can personalize user experience by providing automated recommendations. RSs operate by analyzing user preference data, trying to identify correlations between them. User preference is expressed in various forms such as the history of purchases, usage logs and numerical ratings in a predefined scale (e.g. five star rating system). The RSs, in return, may propose a variety of items; for example, an on-line shop could suggest books or movies to users based on their profile, their previous purchases, the preferences of his friends and other users with similar interests.

Package Recommender Systems extend the classical RSs by proposing to their users sets of items (packages) instead of single items. Package recommendation

is extremely useful in a number of application domains (e.g. recommendation of packages of academic courses to students, packages of meals for a weekly diet, travel packages and sets of movies or books). This work focuses precisely on package recommender systems that are a specific category of RSs.

This work presents a recommendation framework which can be applied on systems that group items into categories and make package recommendations to users under various constraints (e.g. time, money). The proposed framework composes packages by selecting from a pool of items that may belong to multiple categories, using as input only the users previous preference data (ratings of the items), the package size and a package cost threshold. The recommended packages comprise of items that match user preferences and satisfy the package cost restriction, as well as the package composition criteria that concern the categories of the selected items. The recommended packages are expected to achieve the highest preference score of the user to whom they are proposed to, while satisfying all user restrictions. Some examples are movie packages (a number of movies of different categories with a maximum total duration), weekly diets (a number of meals per day comprising of plates of a nutritional value and predefined calories) etc.

An optimal and a greedy solution, which are based on the user preferences and given restrictions in order to recommend the top-k packages, are proposed. The prediction of a user's preference score for an item is performed using the technique of item-based collaborative filtering [1]. The optimal solution matches items to categories so as to obtain the top-k packages of items with maximum preference score and at the same time it satisfies the constraint of maximum package cost. This optimal approach tries to solve the problem of recommendations by modeling it as a minimum cost flow problem that is solved by integer linear programming. On the other hand, the greedy solution performs in low running time due to low computation complexity and its recommendations are close to those of optimal solution. The proposed solutions were evaluated by estimating a set of measures and compared with a baseline approach, which is based on the popularity of items, without taking into account the current user preferences. For evaluation purposes, we have used two popular datasets and we have confirmed the robustness of the solution.

2 Related Work

Package recommendation has attracted the attention of the scientific community in recent years. In [2], the authors propose a system for recommending a team of experts, who have a set of predefined skills and a minimum communication cost. These experts are connected and communicate with each other within a social network. Their approach bears a similarity to our methodology as the skills may correspond to the categories of our work.

Other systems try to satisfy hard constraints among proposed packages of fixed size [3, 4]. The packages can be the top-k tuples of entities that match user queries [3], or the result of rank join queries that formulate user aggrega-

tion constraints [4]. When a maximum cost (budget) restriction is added [5–7], the algorithms create packages that maximize user preference score satisfying the given constraints. In [8, 9], authors introduce restrictions on prerequisite items of package recommendations. These restrictions are valid in certain application domains, e.g. in academic course recommendation, where an advanced course is offered only if other elementary courses have been completed successfully. Another course recommendation system that is based on the maximum flow algorithm is presented in [10]. Our proposed methodology differs from the aforementioned systems as it identifies the minimum cost flow, it includes the restriction of maximum package cost and it is more generic; it can be adapted and applied to a wide range of recommendation domains.

Flexible recommender systems, a special case of package recommender systems, do not obey hard constraints and can adapt to the application domain. In [11] authors introduced an intuitive user interface for travel package recommendation and in [12], they proposed a top-k package recommender that performs user preference elicitation. Our system is different since it is capable of directly producing recommendations based on past user evaluations, without requiring any additional user interaction. In [13], a versatile recommendation system for proposing packages of items has been developed. This system is flexible and does not refer to any particular application, but it is adapted to the requirements of each user. It is a *content-based* recommender system [14] that proposes packages adhering to user-provided constraints. A main difference of this system from our work is that it is based on content while we use collaborative filtering predictions.

Our model and system differentiates the other approaches in the field, since the compatibility constraints apply to the categories in which the items belong. The fact that an item may belong to more than one categories (i.e. item may fit to different packages with different roles), formulates a new package composition problem, which has not yet been discussed in the literature.

3 Package Recommendation

3.1 Problem Definition

The proposed recommender system assumes that a user u from a set of users U is interested in packages of fixed size, composed of items t_i from a set T. All items in T are considered to be of the same type (e.g. movies, plates or food portions, POIs) but may belong to one or more categories from a set C of l categories. Each user has provided ratings for several items of T and we can use these ratings to understand the user preferences, for both the specific items and the categories these items belong to. The rating information can be typically represented with a rating matrix R of size $n \times m$, where n is the total number of users and m is the total number of items. This matrix is usually sparse, since users typically rate only a few items, but by using a collaborative filtering algorithm, ratings for items that users have not rated yet are possible to be predicted [15]. Each item t_i has an associated cost i_{cost} (e.g. the movie duration, the distance to reach a POI or the money which is spent there or a combination of the two). Also each

item has an item value i_{value}, which can be the item rating (actual or predicted). As a result, each package is a fixed-size set of s items that has a total cost p_{cost} (the sum of the costs of the items that each package contains) and a total value p_{value} (the sum of the selected item values).

The input parameters are the package size s, a maximum package cost max_{pcost}, optionally the number of alternative packages k and the user preferences which are represented by the number of categories per package or specifically by the number of items from each category (n_{cj}). The system recommends the package (or the $top-k$ packages in non increasing p_{value} order) that covers user preferences, not surpassing max_{pcost} and maximizing the total package value p_{value}. Table 1 summarizes the notations used in the proposed model.

Table 1: Notation summary

Symbol	Description
U	The set of all users, $U = \{u_1, ..., u_n\}$, $\|U\| = n$
T	The set of items, $T = \{t_1, ..., t_m\}$, $\|T\| = m$
C	The set of categories $C = \{c_1, ..., c_l\}$, $\|C\| = l$
R	The *user-item* rating matrix (dimension $n \times m$)
i_{cost}	The cost of having item t_i in the package
i_{value}	The value that item t_i adds to the package
p	A recommended package with items $p = \{t_1, ..., t_s\}$
p_{cost}	The total package cost
p_{value}	The total package value
s	The package size (number of items in the package)
max_{pcost}	The maximum allowed package cost
k	The maximum number of recommended packages
n_{cj}	The number of items of category c_j in the package.

A package p is a set of s items $\{t_1, ..., t_s\}$, where each item t_i belongs to one or more categories from C, but within the package each item represents a specific category c_j. The package can be recommended when it satisfies the compatibility and aggregation constraints of Equation 1, where $|t_{cj}|$ is the number of items that represent the category c_j.

$$\begin{aligned} |t_{cj}| = n_{cj},\ \forall c_j \in \{c_1, ..., c_l\} \\ p_{cost} \leq max_{pcost},\ where\ p_{cost} = \sum_{t_i \in p} i_{cost} \\ |p| = s \end{aligned} \tag{1}$$

The total package value is given by Equation 2 and the recommender system proposes the $top-k$ packages in non increasing p_{value} order.

$$p_{value} = \sum_{t_i \in p} i_{value} \tag{2}$$

Package compatibility constraints: The main package composition constraint according to Equation 1 refers to the number of items per category in the package ($n_{cj} : c_j \in \{c_1, .., c_l\}$). These numbers can be given as input by the

user or they may be derived implicitly from past user preferences. In this case, the popularity of each category is determined first, followed by the distribution of items among the q most popular categories. The popularity of a category c_j may depend on the number of items of this category that the user had rated in the past, the sum of the ratings the user has provided for the items in this category or any other method derived from the actually rated items. Once the category popularity for each user has been computed, then the top-q categories can be selected for composing the package, where q may be provided by the user explicitly. Based on the popularity scores for the top-q categories, we can derive the number of items for each category in the package, which may be proportional to the category popularity, or the same for all popular categories.

Package aggregation constraint: This constraint refers to the total package cost p_{cost}, which must not exceed a maximum threshold value max_{pcost}.

Item value: The value of an item i_{value} can be the rating explicitly provided by the user for this item, or implicitly predicted by the system, e.g. using a collaborative filtering algorithm. An alternative method for defining the value of an item i, that combines the specific user value $r_{u,i}$ with the item popularity is given by the following equation:

$$i_{value} = r_{u,i} + PF \cdot \frac{n_i}{n} \tag{3}$$

where $PF \in \{0, 1, 2, ..., 10\}$ is a popularity factor, n_i is the number of users that have rated item t_i and n is the total number of users.

4 Solving the package composition problem

4.1 Minimum cost flow model

The problem of finding a package of items that satisfies the restrictions of Equation 1 and at the same time maximizes the package value of Equation 2 is an optimization problem which may add a big computational load to a recommendation system. A good algorithm that finds the optimum solution for each user is the key to the efficiency and effectiveness of the recommender system. The key concept is that each category may be optimally matched to a number of items according to the parameters of the problem. Essentially, this is a problem of optimal matching, which can be reduced to a minimum cost flow problem [16].

Since the proposed model aims at maximizing the package value for a predefined package size s, the total flow must be equal to the package size and the edge cost must be inverse to the item value in order for a minimum cost flow model to be used. The basis of the flow network is the bipartite graph $G(T, C, E)$, where T and C are the sets of item and category vertices respectively and E is the set of edges. An edge e_{ij} connects item t_i with category c_j and denotes that the item belongs to the category. The bipartite graph is extended, for each user, to a flow graph (Figure 1), by connecting the source vertex Src to all item vertices that the user has not yet valued as well as all the category vertices that are in the interest of the user to the termination vertex Trm. The edges carry the

information $fc_{i,j}/max_{capij}/min_{capij}$, where $fc_{i,j}$ is the flow cost over the edge e_{ij}, max_{capij} and min_{capij} are the maximum and minimum edge capacity.

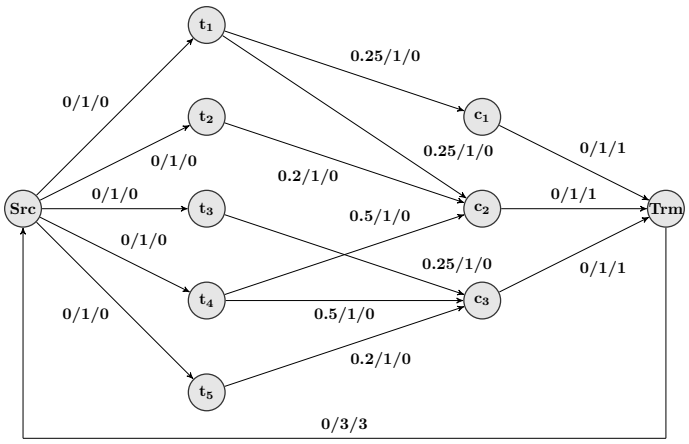


Fig. 1: An example of the minimum cost flow problem formulation

The maximum flow from Src to Trm may be reduced to the optimal matching between items and categories which, in turn, may lead to the creation of the optimal recommendation package. The derivation of this optimal matching between items and categories from the maximal flow problem is due to the validity of the integrality theorem [16] which claims that if all the capacities of the edges of a flow graph have integer values, then the maximum flow also assumes an integer value. Therefore, it is this theorem that allows the optimal matching between an integer number of items and categories. Adding a cost for passing the flow through a graph edge transforms the problem to a *minimum cost flow* one, and its solution results in the recommendation of the top package that maximizes the item value.

The cost for passing the flow through an edge that is connected to the source Src or terminal Trm vertices is zero ($fc_{Src,i} = fc_{j,Trm} = 0,\ \forall i \in T, \forall j \in C$). The flow cost $fc_{i,j}$ through the edge e_{ij} that connects item t_i with category c_j is the inverse of the (predicted) item value (i_{value}) and it is the same for all the edges that connect t_i with a category vertex (see Equation 4).

$$fc_{i,j} = \frac{1}{i_{value}}, \forall i \in T, \forall j \in C \tag{4}$$

The maximum capacity of an edge connecting Src with an item vertex is equal to the maximum number of times the same item can be accessed (e.g. in the movie package this is always 1), while the minimum capacity is 0. The maximum capacity of edges connecting items to categories is always 1 since an item may belong to the same category only once (the minimum is 0). Finally, the

maximum capacity of edges connecting category vertices and the Trm vertex is the number of items per category in the package (the minimum is 0). Figure 1 shows an example of a minimum-cost flow graph for finding the best package of size $s = 3$ with exactly 1 item from each of the categories c_1, c_2 and c_3. The predicted user ratings for items $\{t_1, t_2, t_3, t_4, t_5\}$ are $\{4, 5, 4, 2, 5\}$ respectively in the 0-5 scale. The optimal package for this problem includes items t_1, t_2, t_5, which minimize the cost of the flow with the minimum cost of 0.65 (i.e. 0.25+0.2+0.2). The respective package value $p_{value} = 14$ is the maximum possible.

Integer linear programming formulation: In the general case, the problem of discovering the minimum cost flow of the graph may be re-formulated in terms of linear programming [17]. In addition, the proposed model has an aggregation constraint; the maximum package cost threshold (max_{pcost}), which relates to budget or time restrictions that apply to the recommended package as a whole. The linear programming objective function and the constraints are depicted in Equation 5, where $V = \{T, C, Src, Trc\}$ is the set of min-cost flow graph nodes, f_{ij} is the flow passing from an edge e_{ij} and i_{cost} is the cost of item $t_i \in T$.

$$\begin{aligned} \text{minimize: } & \sum_{i,j \in V,\, i \neq j} fc_{ij} \cdot f_{ij} \\ \text{subject to: } & min_{cap_{ij}} \leq f_{ij} \leq max_{cap_{ij}},\ \forall i, j \in V \,\&\, i \neq j \\ & \sum_{j \in V,\, i \neq j} f_{ij} = 0,\ \forall i \in V \\ & \sum_{i \in T,\, j \in C} f_{ij} \cdot i_{cost} \leq max_{pcost} \\ & f_{ij} \in \mathbb{Z}_{\geq 0} \end{aligned} \tag{5}$$

4.2 Greedy solution and baseline

A greedy alternative begins by adding to the package items from the most wanted category (as defined by the user preferences) and continues with the other categories in non increasing i_{value} order, satisfying the composition restrictions and without violating the aggregation restriction (i.e. maximum package cost). If the package cost exceeds max_{pcost}, then the item with the minimum $i_{value}/cost$ value is replaced with the next item (in non increasing rating order) of the same category (in order to keep composition restrictions and improve towards the aggregation restriction). The algorithm terminates successfully when all the composition restrictions have been met (i.e. the requested number of items from each category has been added to the package) without exceeding max_{pcost}. It fails when all the items of a requested category have been examined but they cannot fit for the package.

Finally, a baseline method, which ignores user preferences and assumes that an item value (i_{value}) is proportional only to the overall popularity of the item, replace the predicted ratings for each user in the greedy algorithm, giving us a non-personalized baseline.

Data: u, T, C, R, s, max_{pcost}, k
Result: top-k packages
forall $c_j \in C$ **do**
  $T_{cj} \leftarrow$sort items in decreasing value (rating) order for category c_j;
end
forall Pi, $i = \{1, ..., k\}$ **do**
  $Pi \leftarrow$ top n_{cj} items for each category c_j;
  while $Pi_{cost} > max_{pcost}$ **do**
    replace t_x of minimum $\frac{i_{value}}{i_{cost}}$ with t_{x+1}: $category_{t_x} = category_{t_{x+1}}$
  end
  Update T_{cj} sets to produce the next package;
end

Algorithm 1: Greedy algorithm for the top-k package creation.

4.3 Composing top-k packages

In the minimum cost flow model, the problem of composing the $top - k$ packages can be solved by repetitively using the minimum cost flow formulation for slightly modified flow networks. Since it produces the optimum solution each time, it is necessary to update the graph on each iteration by removing vertices and edges based on what was selected in the previous packages. The Greedy algorithm follows a similar strategy (e.g. items with the minimum i_{value}/i_{cost} scores that have already been recommended can be removed from the set of candidate items).

4.4 Computational complexity

The problem of the Optimal solution which is formulated as integer linear programming is NP-Complete [18]. In our case, the pruning of the equations by using a certain number of items with higher values, the problem is solved in reasonable time as it is demonstrated in section 5. The complexity of the Greedy algorithm for the first loop (sort ratings by category) is $O(|C| \cdot |T| \cdot log|T|)$ and for the second loop is $O(k \cdot |T|)$, in the worst case where all items belong to all categories and all items of the categories are examined for composing packages. Since $|C|$, $k << |T|$, the computational complexity of the Greedy algorithm is $O(|T| \cdot log|T|)$. This complexity allows the algorithm to perform in low running times as shown in the experimental evaluation.

5 Evaluation

In order to implement and evaluate the proposed model, we have developed a Java application[3] that allows us to test the proposed solution in a wide range

[3] The application jar file, source code, usage instructions and a sample dataset, which was also used for the evaluation, are available for downloading at https://goo.gl/IMbxq1.

of application scenarios, by modifying system parameters through a GUI. It also permits an incomplete rating dataset (not all users rate all items) to be imported and the missing ratings to be predicted using a Collaborative Filtering (CF) algorithm[4]. The application is connected to an external linear programming module[5], which solves the optimization problem of the optimal approach.

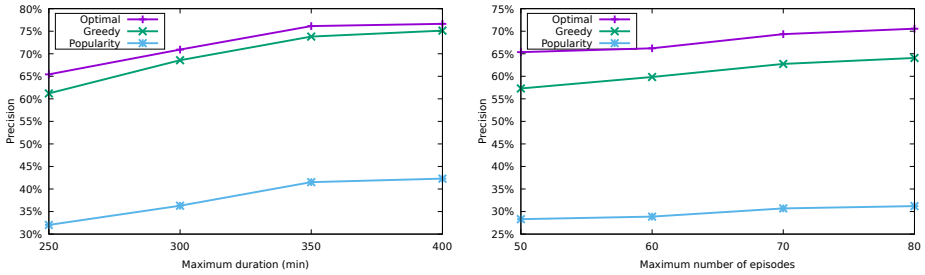
Datasets: The first dataset, Movielens 1M[6], includes almost 1 million ratings provided by 6,040 users for 3,900 movies. The durations of films are retrieved from the OMDB[7] web service and used as the cost of each movie item. The second dataset is a subset (1M ratings) of the Anime[8] dataset, including more than 7 million ratings provided by 73,516 users on 12,294 series. The cost for an Anime series is set to be the number of its episodes.

Experimental parameters: Since the proposed solution is generic, it is not possible to examine its performance in all possible parameter setups (e.g. the number of packages, the package composition strategy etc). Although we evaluated many scenarios, due to space limitation, we fix some parameters, which remain the same in all the experiments and evaluate the effect of the aggregation and composition parameters in system performance. More specifically, we evaluate the top-10 packages, comprising of one item from each of the top-3 most popular categories for each user and the choice is made from the top-500 highest rated items (prediction) for the user.

Evaluation methodology: The item-item CF algorithm, an algorithm with proven performance [19], with the Tanimoto coefficient for measuring item similarity [20] being employed for predicting item ratings. For the evaluation of the package recommender algorithms, we choose randomly 50 users who have rated more than 500 items (there is a total of 396 such users in MovieLens and 223 in Anime dataset). For these users the 40% of their ratings is hidden (random stratified sampling per user) and all the remaining ratings in the dataset are used for training. We repeat this Monte Carlo cross-validation technique five times [21] and report the average performance of the package recommender algorithms. The metrics employed for the performance evaluation of our model are the precision and value of each proposed package. In order to evaluate the time performance, we also report the *total running time*. The *package value*, is given in Equation 2. *Precision*, given in Equation 6 (where T_p is the set of items in the package and T_h is the set of highly rated "hidden items"), counts the number of items in a package p that have been rated by the user, but the respective ratings have not been used for training ("hidden items"). Since we do not want to recommend items that have actually received low ratings by the user, we consider only those "hidden items" that have been rated highly (rating ≥ 4).

$$Precision_p = |T_p \cap T_h|/|T_p| = |T_p \cap T_h|/s \tag{6}$$

[4] Item-based CF implementation of Apache Mahout (https://mahout.apache.org)
[5] IBM ILOG CPLEX solver.
[6] https://grouplens.org/datasets/movielens/
[7] http://www.omdbapi.com/
[8] https://www.kaggle.com/CooperUnion/anime-recommendations-database



(a) Movielens Package Precision. (b) Anime Package Precision.

Fig. 2: Precision with varying maximum package cost.

5.1 Results

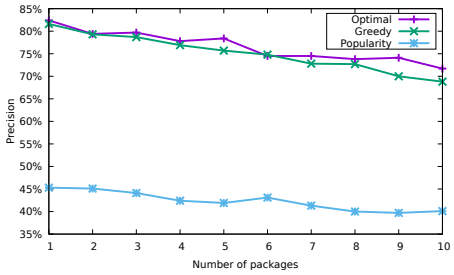
As far as the maximum package cost is concerned, we have experimented with various thresholds and the results in Figure 2 show that the precision increases by relaxing this aggregation constraint as expected. The Optimal algorithm performs better than the Greedy and both algorithms outperform the baseline approach.

A comparison of the three approaches, in recommending the top-k packages of size 3 (1 item from the top-3 categories), as depicted in Figure 3, reveals the superiority of the proposed approach, when compared against the Greedy and the Item-Popularity alternatives. Although the precision of the greedy algorithm is comparable to the optimum in the top-3 packages, the respective package value quickly deteriorates, which means that the greedy algorithm recommends an equal number of "hidden items" but with lower ratings. As for the time complexity of each algorithm is concerned, Figure 4 shows that the greedy solution is much faster due to low computational complexity and returns the top-10 packages in less than 1ms, whereas the Optimal Solution needs 700ms for the top-10 packages.

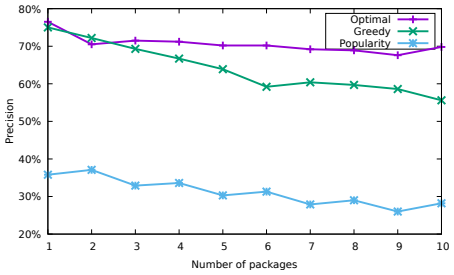
Based on the experimental results, we conclude that both the Optimal and the Greedy solution operate efficiently, achieve low running times and high precision rates. The Greedy solution can be applied in applications that require low running times and low computational complexity even though it may lose in package quality, whereas the Optimal solution can be applied in applications that require a maximum recommended package quality.

6 Conclusions and future work

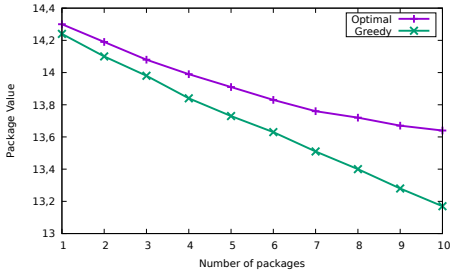
This work proposes a package recommender framework that employs user preferences. Our model is not designed for a particular domain but can be applied widely as it is comprised of a number of parameters which can be configured accordingly, allowing it adapt to various recommendation problems. An integrated recommendation system, which incorporates all the required functionality, was



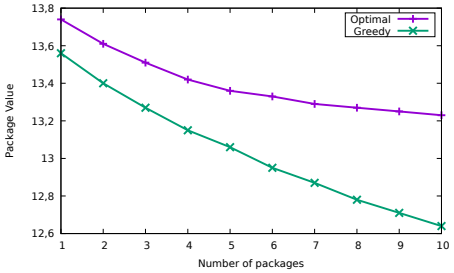
(a) Movielens Package Precision.



(b) Anime Package Precision.

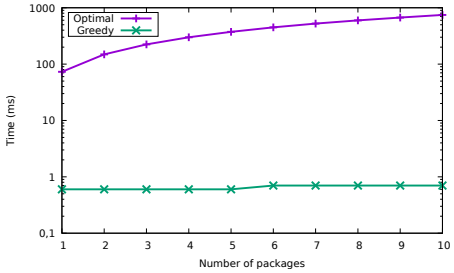


(c) Movielens Package Value.

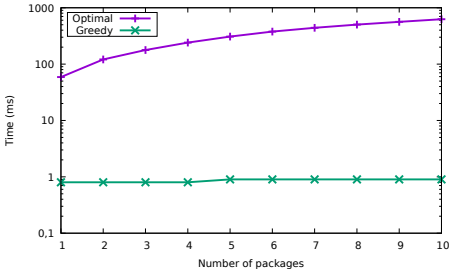


(d) Anime Package Value.

Fig. 3: The Precision and Value of the top-k packages (for various k values).



(a) Running Times for Movielens.



(b) Running Times for Anime dataset.

Fig. 4: Running time for creating the top-k packages (for various k values).

developed in order to support the implementation and evaluation of our approach. The results of the evaluation experiments showed that the proposed model fulfills its purpose and works effectively and efficiently.

The first positive evaluation results of our approach lead us to think about further extensions. It is on our next plans this model to be applied and evaluated in the nutrition domain as it may be a solution for recommending personalized gastronomic diets. We are also working on automatically setting the system pa-

rameters which depend on the application, the dataset and the user preferences. This may allow us to achieve high performance.

References

1. Ekstrand, M.D., Riedl, J.T., Konstan, J.A., et al.: Collaborative filtering recommender systems. Foundations and Trends in HCI **4**(2) (2011) 81–173
2. Lappas, T., Liu, K., Terzi, E.: Finding a team of experts in social networks. In: 15th ACM SIGKDD, ACM (2009) 467–476
3. Angel, A., Chaudhuri, S., Das, G., Koudas, N.: Ranking objects based on relationships and fixed associations. In: 12th EDBT, ACM (2009) 910–921
4. Xie, M., Lakshmanan, L.V., Wood, P.T.: Efficient rank join with aggregation constraints. VLDB Endowment **4**(11) (2011) 1201–1212
5. Xie, M., Lakshmanan, L.V., Wood, P.T.: Breaking out of the box of recommendations: from items to packages. In: 4th RecSys conference, ACM (2010) 151–158
6. Xie, M., Lakshmanan, L.V., Wood, P.T.: Comprec-trip: A composite recommendation system for travel planning. In: 27th ICDE Conference, IEEE (2011) 1352–1355
7. Benouaret, I., Lenne, D.: A package recommendation framework for trip planning activities. In: 10th RecSys Conference, ACM (2016) 203–206
8. Parameswaran, A.G., Garcia-Molina, H.: Recommendations with prerequisites. In: 3rd RecSys conference, ACM (2009) 353–356
9. Parameswaran, A.G., Garcia-Molina, H., Ullman, J.D.: Evaluating, combining and generalizing recommendations with prerequisites. In: 19th ACM CIKM, ACM (2010) 919–928
10. Parameswaran, A., Venetis, P., Garcia-Molina, H.: Recommendation systems with complex constraints: A course recommendation perspective. ACM Transactions on Information Systems (TOIS) **29**(4) (2011) 20
11. Xie, M., Lakshmanan, L.V., Wood, P.T.: Ips: an interactive package configuration system for trip planning. VLDB Endowment **6**(12) (2013) 1362–1365
12. Xie, M., Lakshmanan, L.V., Wood, P.T.: Generating top-k packages via preference elicitation. VLDB Endowment **7**(14) (2014) 1941–1952
13. Interdonato, R., Romeo, S., Tagarelli, A., Karypis, G.: A versatile graph-based approach to package recommendation. In: Tools with Artificial Intelligence (ICTAI), 2013 IEEE 25th International Conference on, IEEE (2013) 857–864
14. Melville, P., Sindhwani, V.: Recommender systems. In: Encyclopedia of machine learning. Springer (2011) 829–838
15. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The adaptive web. Springer (2007) 291–324
16. Dasgupta, S., Papadimitriou, C.H., Vazirani, U.V.: Algorithms. McGraw-Hill (2008)
17. Bertsekas, D.P.: Network optimization: continuous and discrete models. Belmont: Athena Scientific. (1998)
18. Matousek, J., Gärtner, B.: Understanding and using linear programming. Springer Science & Business Media (2007)
19. Seminario, C.E., Wilson, D.C.: Case study evaluation of mahout as a recommender platform. In: RUE@ RecSys. (2012) 45–50
20. Anil, R., Owen, S., Dunning, T., Friedman, E.: Mahout in action. (2012)
21. Dubitzky, W., Granzow, M., Berrar, D.P.: Fundamentals of data mining in genomics and proteomics. Springer Science & Business Media (2007)