Operations Research and Recommender Systems

Thomas Asikis and George Lekakos

Department of Management Science and Technology,
Athens University of Economics and Business, Athens, Greece
asikis.thomas@gmail.com, glekakos@aueb.gr

Abstract. Nowadays, Recommender Systems (RS) are being widely and successfully used in online applications. A successful Recommender System can help in increasing the revenue of a web-site as well as helping it to maintain and increase its users. Until now, research in recommendation algorithms is mainly based on machine learning and AI techniques. In this article we aim to develop recommendation algorithms utilizing Operations Research (OR) methods that provide the ability to move towards an optimized set of items to be recommended. We focus on expressing the Collaborative Filtering Algorithm (CF or CFA) as a Greedy Construction Algorithm as well as implementing and testing a Collaborative Metaheuristic Algorithm (CMA) for providing recommendations. The empirical findings suggest that the recommendation problem can indeed be defined as an optimization problem, which provides new opportunities for the application of powerful and effective OR algorithms on recommendation problems.

Keywords: Recommender Systems, Personalization algorithms, Operational Research, Metaheuristic.

1 Introduction

The number of possible choices in various sites that offer products or services increases rapidly day by day. Indeed, a typical user may have to make decisions such as: “With whom should I connect in a social network?”, “What song should I hear on Soundcloud?”, “Which product should I buy on Amazon?”. Websites, offer a vast amount of possible choices to questions such as the above ones, so users need support in their decision making process in order to make the best possible selections [1].

Recommender Systems (RS) represent a type of information filtering systems that aim at predicting a user’s items of interest in a large space of possible options, based on his previous preferences [2]. Typical recommendation approaches include collaborative, content based filtering and hybrid methods. Collaborative Filtering RS rely mostly on the behavioral similarity between users. Content Based Filtering RS focus on the similarities between item features that a user has favored in the past. In addition, several hybrid methods utilizing collaborative, content-based, demographic, and knowledge-based criteria have been developed in the last years [3]. In this article, the focus is on the collaborative approach due to its popularity, simplicity and intuition.

S. Yamamoto (Ed.): HIMI 2014, Part II, LNCS 8522, pp. 579–589, 2014.

Operations research (OR) provides methods and techniques that support the decision making process, by evaluating every possible alternative and estimating the potential outcome [4]. The main idea underlying the work presented in this paper is to exploit OR methods towards an optimized set of items to be recommended to the user.

Combinatorial Optimization is a method used in OR for the identification of optimal object/objects in a finite collection of objects [5]. It provides a suitable framework for the definition of a recommendation problem as an OR "Selection" problem. Recommender Systems aim at creating a selection of items that a user is most likely to respond positively (and eventually purchase or use the proposed item/items). In OR Selection problems, the aim is to select each item for the recommendation based on a choice criterion and evaluate the solution based on a specific evaluation criterion.

Along the above line of thinking, the objective of this paper is twofold: (a) to demonstrate that the recommendation problem can be defined and treated as an OR problem and (b) to design a well-performing recommendation algorithm based on OR techniques.

In the following, we firstly present extant related research and then we define the recommendation problem as an OR problem. At the next section we describe the implementation of a Collaborative-filtering Greedy-construction Algorithm (CGA) and a Collaborative Metaheuristic Algorithm (CMA) to determine if the combinatorial approach is suitable for the Social Recommender System Problem (SRSP). Finally, an empirical evaluation of the above algorithms is presented and the results are discussed in the final section of the paper.

2 Related Work

In this section we present some relevant RS techniques and discuss if they can be used to enhance the OR implementation in RS. In general, there are a lot of criteria that can be used for optimizing a recommendation. This provides us the flexibility to construct more reliable and accurate OR optimization methods. Trust is a concept that in general reflects the probability of someone doing an action, based on the actions performed from another person [6]. People are naturally grouped by trust. People who trust one another, usually can influence the behavior of each other and a high level of trust usually benefits all the parties in a transaction, by reducing the transaction cost between the seller and the buyer of an item. Moreover, trust can be aggregated and propagated though the members of a system. Those features have made trust one of the key components of some RS [7].

The advantages of the trust-based recommender systems can be found in a number of aspects: invulnerability to malicious attacks, greater control of the recommendation process, an explanation can be provided to users, for each item recommended to them. However, an important problem of trust-based algorithms, is data sparseness. Trust-based algorithms tend to face difficulty in sparse datasets [7] and their ability to produce good recommendations is limited.

On the other hand, features (mostly used in e-commerce recommenders where online purchases are enabled) such as the item's price may be used as a selection

criterion. Such criteria are particularly useful for optimization problems that aim at maximizing product supplier's profits. Therefore the optimization problem can be rephrased as the following problem: How can we provide recommendations that match user's interests while ensuring the maximum profit for the provider? [8;12].

Novelty and diversity of recommendations can also serve as selection criteria as they may lead to quite effective recommendations [9]. Novel and diverse recommendations refer to items beyond the typical spectrum of items previously seen or consumed by the user and therefore they are perceived as an unexpected option that they user may have never considered in the past [10].

3 Defining Recommendation as an OR Problem

3.1 Problem Definition

A RS can be defined by the following:

$U = \{u_1, u_2 \ldots u_N\}$ a set of users.
$A = \{a_1, a_2 \ldots a_n\}$ a set of items.
$C_i = \{c_1, c_2 \ldots c_m\}$ the set of items that user i has already chosen.
$P_t = \{p_1, p_2 \ldots p_k\} \rightarrow$ the set of items that are selected to be recommended to U_t (recommendations).

From the above formalization we have:

$u_t \rightarrow$ the user who will receive the recommendations.
$C_t = \{t_1, t_2 \ldots t_m\} \rightarrow$ the set of items that U_t has already used.
$Ri = \{r_{ck} \ldots r_{cj}\} \rightarrow$ the set of ratings that a user i has given to an item c_j.
The above problem in combinatorial terms, can be expressed as follows:
"Which is the optimal[1] set of items that we can recommend to a user?"

In order to operationalize the above problem in Operations Research terms, the following elements are defined:

Form of Problem Solution: The solution to the above problem is a selection of items from set A, that represents the set of items to be recommended (P_t).

Element of Solution: The element of solution is the single item r_j that will be recommended to the user.

Criterion of Choice: The criterion for selecting an item from A and use it as a recommendation in P_t. It can be expressed by a function of multiple criteria $f(c_1, c_2, \ldots c_3)$. Depending on the value of this function, it can be determined if the item can be used as a recommendation.

Evaluation Criterion: The criterion, which that will be used in order to evaluate the solution/recommendations provided to Ut. One of the most difficult and crucial tasks towards the solution of the recommendation problem is to define an evaluation criterion that could increase the quality of recommendations, without increasing the time

[1] Optimal means the set of items that will best match the user interests.

consumed for the algorithm execution. In general the evaluation criterion is used for evaluating the solutions produced from an algorithm. A metaheuristic can produce a vast number of possible solutions. The evaluation criterion is used by the metaheuristic to evaluate the solutions and choose the optimal one[15].

3.2 Algorithms

Collaborative-Filtering as a Greedy-Construction Algorithm (CGA). The first algorithm implemented is a Greedy Construction algorithm, which is based on the Collaborative Filtering. The algorithm has the following characteristics:

Criterion of Choice: The criterion upon which an item c from A will be selected and used as a recommendation in P_t for the user U_t is the "collaborative rating":

$$r'_{tc} = \bar{r}_t + \frac{\sum correl(U_t,U_i)*(r_{ic}-\bar{r}_i)}{\sum correl(U_t,U_i)} \quad (1)$$

Where:
r'_{tc}: is the predicted rating for the U_t for item c.
$\bar{r}_t$: is the average rating score of U_t.
$\bar{r}_i$: is the average rating of a user U_i.
$correl(U_t, U_i)$: is the correlation between users U_t and U_i, based on their known ratings.
r_{ic}: is the rating of item c from user i.

The algorithm is executed for a target user U_t from U, as follows:

1. Pick the target user U_t from U.
2. Calculate all correlations between U_t and the other users of U.
3. Create neighbors of users that share correlation higher than 0.5, in a set $U_{similars}$.
4. Based on those users, calculate the collaborative score for each item they have rated, and user U_t hasn't rated.
5. Create the set A_{recs}, consisting of the above (step 4) rating
6. For each item in A_{recs}: if it has score greater than 4, put the item in the recommendations set P_t.
7. After putting all the appropriate items in P_t, recommend the items to the user.
8. Evaluate the solution. Compare the average of the correlations the user had with all the users, with the one he has now.
9. End.

Collaborative Metaheuristic Algorithm (CMA). This is the second algorithm that was developed and tested. CF can be used both as a metaheuristic and as a constructive algorithm. Constructive or Heuristic Algorithms, construct a solution from zero, building it element by element. To use a constructive CF, we created a solution from zero, adding items based on their collaborative score. To implement a collaborative model in a Metaheuristic Algorithm. An algorithm which uses already existent solutions, to find better ones. The basic elements of the CFA are the following:

Criterion of Choice: The criterion upon which an item c is selected from A for inclusion or exclusion to P_t is the following:

Inclusion Criterion:

$$g_c = \frac{0{,}5*d_t+0{,}4*correl'^{(U_t,U_i)}+0{,}1*r_{ic}}{d_t+1+correl'^{(U_t,U_i)}+r_{ic}} * random > y \quad (2)$$

Exclusion criterion:

$$h_c = \frac{0{,}5*d_t+0{,}4*\frac{1}{correl'^{(U_t,U_i)}}+0{,}1*\frac{1}{r_{ic}}}{d_t+\frac{1}{correl'^{(U_t,U_i)}+}+\frac{1}{r_{ic}}} * random > v \quad (3)$$

Where:

$$d_t = \left|\frac{\sum_i^n correl(U_t,U_i)}{n} - \frac{\sum_i^n correl'^{(U_t,U_i)}}{n}\right| = \left|\overline{correl(U_t)} - \overline{correl'(U_t)}\right| \quad (4)$$

d_t : is the absolute difference between the initial average correlation the user had, and the current average correlation he has now.

$\overline{correl(U_t)}$: The initial average correlation of U_t with other users.

$\overline{correl'(U_t)}$: The current average correlation of U_t with other users.

$correl'(U_t, U_i)$: The new correlation between U_t and U_i, after the algorithm has removed and imported new items.

$correl(U_t, U_i)$: The initial correlation user U_t had with other users, before the start of the algorithm.

r_{ic}: is the rating of item c from user U_i.

$random$: A randomly generated number between 0 and 1. This variable is used to make the CMA a probabilistic algorithm. This is used to express the vagueness and randomness in human behavior.

y, v: Those are the intensification/diversity factors, in this article we are going to call them scope values, because they determine the scope of the solutions area that the algorithm is going to check. With higher values of y and v, less often the algorithm will allow a change to happen in P_t, be it either an import or removal.

Evaluation Criterion: The difference between the initial value of average correlation and the present value of average correlation, of the same user. The purpose of this criterion is to describe the user based on the average correlation he has with the users that are similar to him (correl(U_t,U_i)>0.5). Our aim is to reconstruct the target user's ratings, with new items and see if he still remains similar with the users that used to be highly correlated with him. Every time the algorithm produces a solution, we evaluate it based on the following:

$$\min(d_t) = \min\left(\left|Avg\left(correl(t,i)\right)' - Avg\left(correl(t,i)\right)\right|\right) \quad (5)$$

The algorithm is being executed in the following way:

1. Calculate the initial average correlation $\overline{correl(U_t)}$ for U_t.

2. Create U'_t[2].
3. Calculate[3] a current average correlation value $\overline{correl'(U_t)}$, either randomly or based on U'_t.
4. Calculate d_t. $d_{best}=d_t$.
5. Set $P_{best}=P_t$.
6. For counter = 0, counter<x*[4], counter++
    a. Pick a user U_i from $U_{similars}$.
        i. For each item of U_i
        ii. Pick an item c that U_i has already rated
        iii. Check if c belongs to C_t or P_t.
            1. If it does, check if the removal criterion is fulfilled.
                a. If it is remove the item from P_t.
                b. Else do nothing.
            2. If it does not, check if the removal criterion is fulfilled.
                a. If it is, import the item in P_t.
                b. Else do nothing.
    b. After all removals and imports are done for P_t, calculate the new $\overline{correl'(U_t)}$.
    c. Calculate d'_t .
    d. Compare d'_t with d_{best} .
        i. If $d'_t < d_{best}$. $d_{best}=d'$, $P_{best}=P_t$.
    e. $d_t=d'_t$.
7. Propose the P_{best} as the new recommendation.
8. End

The above Metaheuristic algorithm takes an imported set of chosen items or a set of recommendations, and sets it as Pt. Each time the metaheuristic executes a loop, it changes the contents of P_t and it compares the changes that happened on the average correlation value -Avg(Correl(U_t,U_i))- of the user. The random variable and the constants y and v are the tuning factors that decide if the algorithm is intensifying or diversifying the search. If y and v are low (around 0.4), the algorithm has a bigger scope in the solution area. This means that the algorithm is less likely to get stuck in a local minimum, but it is harder for it to find one.

The number of times the algorithm executes, is also another tuning factor for intensification. The more iterations the algorithm executes before proposing a solution, the more accurate the solution is. As the number of iterations increases, the program

[2] U'_t is the new user. This user can be "created" in various ways. It can be the input from another algorithm. We can create him also by removing/importing randomly solution elements from Pt.

[3] This value (average correlation) doesn't always need an U'_t to be calculated. It can be set with a random value between 1 and 0. We will explain the significance of this later on.

[4] X can be any positive number. It indicates the number of times we want the metaheuristic to execute. Each time the metaheuristic checks a new solution.

spends even more time in finding the solution. CMA is an algorithm that can use any set of choices as P_t, meaning that we can even import a random solution in it, and still expect it to produce better ones.

4 Empirical Evaluation

4.1 Dataset

The empirical evaluation of the above algorithms will be based upon the well-known dataset from epinions.com. This dataset consists of:

- A **set U** of 49.290 users.
- A **set A** of total of 139.738 items.
- In total 664824 ratings were given, as a set of $\Sigma R=r_1+r_2+\ldots+r_{664824}$
- Regarding trust, 487.181 statements were made.

This dataset has also been used to describe the ways of setting up a recommender system for new users, based on trust [11]. The dataset consists of 2 files. The first file provides the ratings data, to be used by the collaborative filtering. The data is represented as:

{User_id Item_id Rating}

- User_id is an integer, with positive values, which provides us the id of the user that gave the rating.
- Item_id is a positive integer also, which provides us the id of the rated item.
- Rating is a positive integer, ranging from 1 to 5. Its value provides us how much the user liked the item in ascending order. 5 means the user liked the item very much. 1 means the user didn't like the item.

4.2 Experimental Environment

In order to successfully test the algorithms' performance, we set up the experimental environment using the following:

- Java programming language for constructing and executing the algorithms.
- The datasets as well as the output data were stored in space delimited text files.
- Statistical processing was partially done by the Java applications and by the use of Microsoft Office Excel®.
- The hardware used for executing the above experiments, were 2 computers with following specs. A desktop with: Intel® i7 960 quad-core processor at 3.2 GHz, 6 GB of Ram and Windows 8 Professional OS. A laptop with: Intel® i7 2670QM quad-core processor at 2.2 GHz and 4 GB of RAM.

4.3 Algorithm Implementation and Tuning

The Collaborative-filtering Greedy-construction Algorithm (CGA), was implemented as it was described above. On the contrary, for the CMA implementation, some special tuning was performed. In order to decide the values of v and y, as well as the number of the algorithm iterations, we had to test it in some simulated datasets and some smaller samples of the Epinions Dataset. It is very important to note that as an input of the CMA after trying several scope values for this dataset - we determined that the following values should be used[5]:

- y=0.15, which means that the algorithm is likely to add new items in the recommendations set easily.
- v=0.9, which means that the algorithm will avoid deleting items often.
- Iterations' number = 50, which means that the algorithm will try 50 times to make the solution more accurate by changing the set of recommendations.

4.4 Results

We executed both algorithms for each user of the dataset, considered as the target user. Each time an algorithm was fully executed, it provided a set of recommendations for the target user. We evaluated the precision and recall of this set, using a test set consisting of the 30% of user ratings which we removed and treated as unknown to the algorithm. The results of the evaluation are summarized in the table below:

	CGA	CMA
Average Recall	0,004872652	0,00808811
Average Precision	0,441938921	0,572996974
Average F	0,008215866	0,012658391

Except from the average values, we checked how both algorithms performed throughout the dataset. The following charts (Fig 1, Fig 2) demonstrate that the CMA performed better, throughout the whole dataset, for all the evaluation metrics. Each chart represents the average score the algorithm achieved in the corresponding metric. The average total choices, show us how many items the user has rated, when he was used as target user. It must be noted that the collaborative approach cannot respond well to a cold start situation[13;14], where the target user has provided no ratings. For cold-start conditions a content-based metaheuristic algorithms is a more efficient solution.

[5] In a real-time applied recommender system, those values would be determined in a training set, and then applied to the production. Also this values, could be changed in real-time execution from the system.

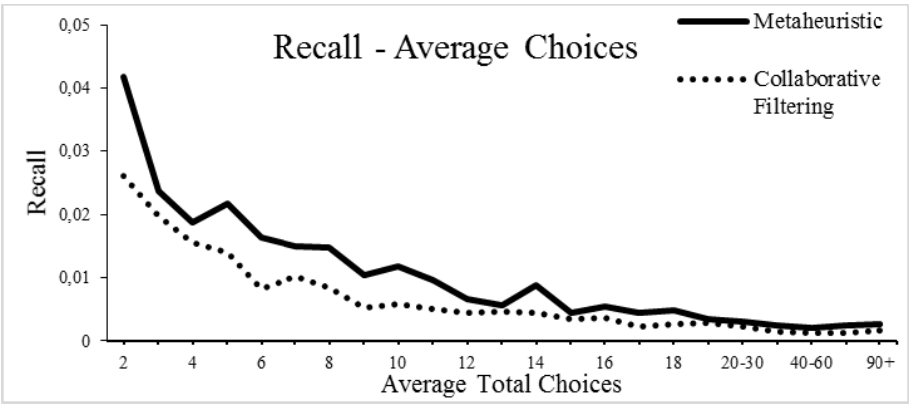

Fig. 1. Algorithm Comparison in terms of Recall

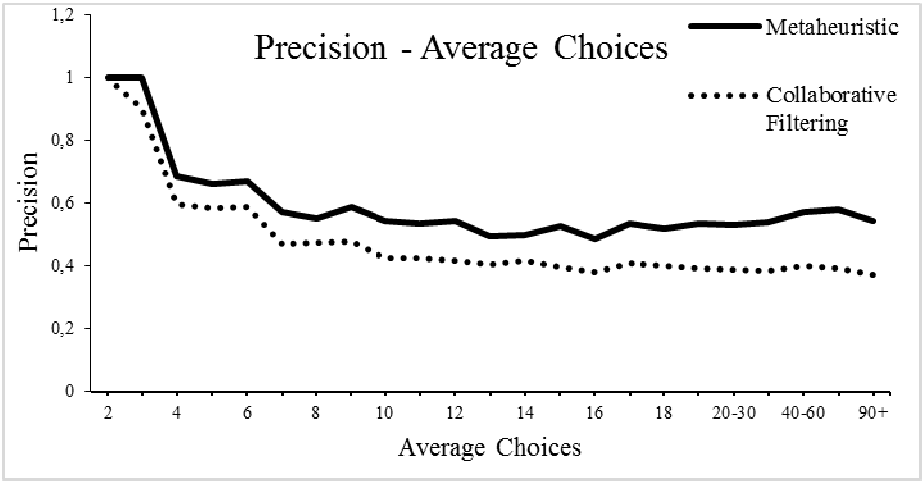

Fig. 2. Algorithm Comparison in terms of Precision

5 Conclusions and Future Work

Recommendation algorithms typically aim at predicting a set of items that the target user is most likely to be interested in. Even considering a highly accurate algorithm, there is no guarantee that the set of recommended items (produced by that algorithm) represents the optimum solution for the given recommendation problem. On the other hand, OR methods aim at optimizing the solutions to selection problems. Thus, the main idea underlying this paper is to utilize OR methods to find the optimum combination of items among the various "good" recommendation sets that can be produced.

We managed to implement the Collaborative filtering algorithm as a greedy constructive algorithm, thus demonstrating that Recommender systems can be treated as

an OR problem. Furthermore we created a metaheuristic algorithm for providing recommendations. Both algorithms were tested on a large and real user dataset. They both gave efficient and precise recommendations, though the metaheuristic being more efficient in the overall.

After analyzing the dataset and the recommendations that the algorithms provided, we realized that there is definitely a correlation between some of the user characteristics and the performance of each algorithm. The metaheuristic proved to be more promising on dealing with datasets and users that provide us sparse information, compared to the Collaborative Filtering.

Other OR algorithms and methods using fuzzy sets and neural networks were taken in notice, and seemed very promising. Combining this with the conclusions of this article we can say that recommendation problems can be definitely be treated as Operational Research problem.

The conclusion of this article can help us research new methods of facing the RS. With the use of OR, we can definitely create new algorithms and systems that can help us provide more accurate and efficient recommendations, such as:

- Designing of constructive algorithms and metaheuristic algorithms in RS using as recommendation criteria trust, diversity, novelty and content based metrics.
- Designing of more complex metaheuristic algorithms such as neural networks, swarm intelligence and genetic algorithms for handling RS.
- Use of evaluation metrics other than recall and precision for the evaluation of the algorithms.

References

1. Hosein, J., Hiang, S.A.T., Robab, S.: A Naive Recommendation Model for Large Databases. Internation Journal of Information an Education Technology, 216–219 (2012)
2. Fransesco, R., Lior, R., Brach, S.: Introduction to Recommender Systems Handbook. Recommender Systems Handbook. Springer (2011)
3. Peter, B., Alfred, K., Wolfgang, N.: The Adaptive Web (2007)
4. Sharma, S.C.: Introductory Operation Research. Discovery Publishing House (2006)
5. Alexander, S.: Combinatorial Optimization. Springer (2003)
6. Gambetta, D.: Can We Trust Trust? Trust: Making and Breaking and Breaking Cooperative Relations (2000)
7. Qiu, Q., Annika, H.: Trust Based Recommendations for mobile Tourists in TIP. Hamiltou: [s.n.] (2008)
8. Fan, W.H., Cheng-Ting, W.: A strategy oriented operation module for recommender systems in E-commerce. Computers nad Operations Research (2010)
9. Paolo, C., Franca, G., Roberto, T.: Investigating the Persuasion Potential of Recommender Systems from a Quality Perspective: An Empirical Study. ACM Transactions on Interactive Intelligent Systems 2 (2012)
10. Saul, V., Pablo, C.: Rank and Relevance in Novelty and Diversity Metrics for Recommender Systems

11. Messa, P., Avesani, P.: Trust aware bootsraping of recommender systems. In: Proceedings of ECAI 2006 Workshop on Recommender Systems, pp. 29–32 (2006)
12. Chen, L.-S., Hsu, F.-H., Chen, M.-C., Hsu, Y.-C.: Developing recommender systems with the consideration of product profitability for sellers (2008)
13. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and Metrics for Cold-Start Recommendations (2002)
14. Lashkari, Y., Metral, M., Maes, P.: Collaborative Interface Agents (1994)
15. Gonzalez, T.F.: Handbook of Approximation Algorithms and Metaheuristics (2007)