2015 IEEE Symposium Series on Computational Intelligence

Optimizing Seed Set for New User Cold Start

He-Da Wang* and Ji Wu†
Multimedia Signal and Intelligent Information Processing Laboratory
Department of Electronic Engineering
Tsinghua University, Beijing, China
Email: *wang-hd12@mails.tsinghua.edu.cn, †wuji_ee@mail.tsinghua.edu.cn

***Abstract*—Users newly enter a recommender system can not get personalized recommendation due to the lack of personal profiles. An interview process that asks new users to rate a set of items (the seed set) will help user modeling and improve user experience. Traditional seed set generation approaches often concentrate on item-wise properties instead of aiming at finding the optimal seed set. We propose a simple random optimization technique to search for the optimal seed set, which considers the seed set as a whole and performs a random search by reducing the prediction error on validation set. By off-line experiments on the MovieLens 10M data set, we show that the proposed approach performs as well as the state-of-the-art method called GreedyExtend, and the proposed approach needs significantly less computational cost to reach the same prediction error as the best baseline on validation set.**

I. INTRODUCTION

The rapid growth of Internet sees an overproduction of online information. The abundance of web content causes people harder to make decisions about which one to buy, which one to read, and which one to watch. Recommendation system (RS) provides an effective way to handle this problem. RS records the behaviors of the users and learns profiles from the feedback collected from the users. It makes recommendations to the users based on these profiles.

A recommender system needs the previous reviews from the users to build profiles for them. A new user just enters the system has no such history information and can not get personalized recommendations. This intrinsic problem of recommender system is called the new user cold start problem [1], [2], [3].

New user problem is common for websites experiencing growth. For the number of active users to grow, the website must take in new users to neutralize the impact of customer churn. Proper welcoming process will allow the system to understand the users better and to improve user satisfaction.

A bootstrapping process called preference eliciting is usually used to get the system to know new users. During such process the system interviews the users with a few questions in order to get feedback, which is used to construct the initial user profiles for them.

One of the most popular forms of preference eliciting is a non-personalized interview, in which every new user will be asked to rate every item from the same list called the *seed set*, as shown in Figure 1. Feedbacks reflecting the preferences of a new user towards the items in the seed set are then used to construct the user profile. Therefore, a good seed set is important for accurate new user modeling.

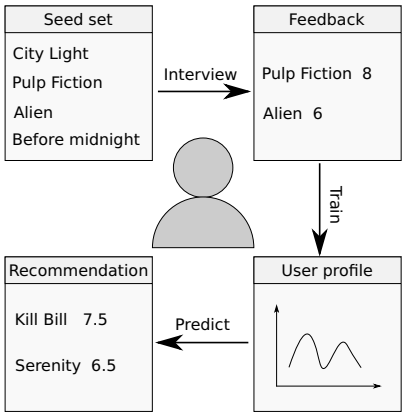

Fig. 1. Preference eliciting for new users allows the system to know new users better and helps the system make more accurate recommendations.

Many researches have been targeted on seed set generation. Some propose useful heuristics [4]. Others address the problem by intuitions from information theory [4], [5], [6]. A more direct approach uses greedy algorithm to search for seed sets with lower training prediction error [7].

The main challenge of seed set generation is how to reduce the redundancy of the seed set and maximize its utility as a whole. Few researches have focused on such optimization problem. To our best knowledge, the state-of-the-art approach is the *GreedyExtend* algorithm [7].

To address the optimization problem of seed set, we propose an hill climbing approach based on random optimization algorithms. We find this algorithm generally performs as well as *GreedyExtend* with significant less computarion cost in an off-line experiment on MovieLens 10M data set.

II. PRELIMINARIES

In this section, we will survey the works in the field of seed set generation. The following notations are used thorough the rest of this paper: u denotes a certain user; i denotes a certain item; r_{ui} denotes the rating that user u expresses to item i; v denotes the value of a rating; U denotes the set containing all users; I denotes the set containing all items; U_i denotes the set containing all users who rate item i; I_u denotes the set containing all items rated by user u; S denotes a seed set containing a certain number of items for interviewing new users.

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.140

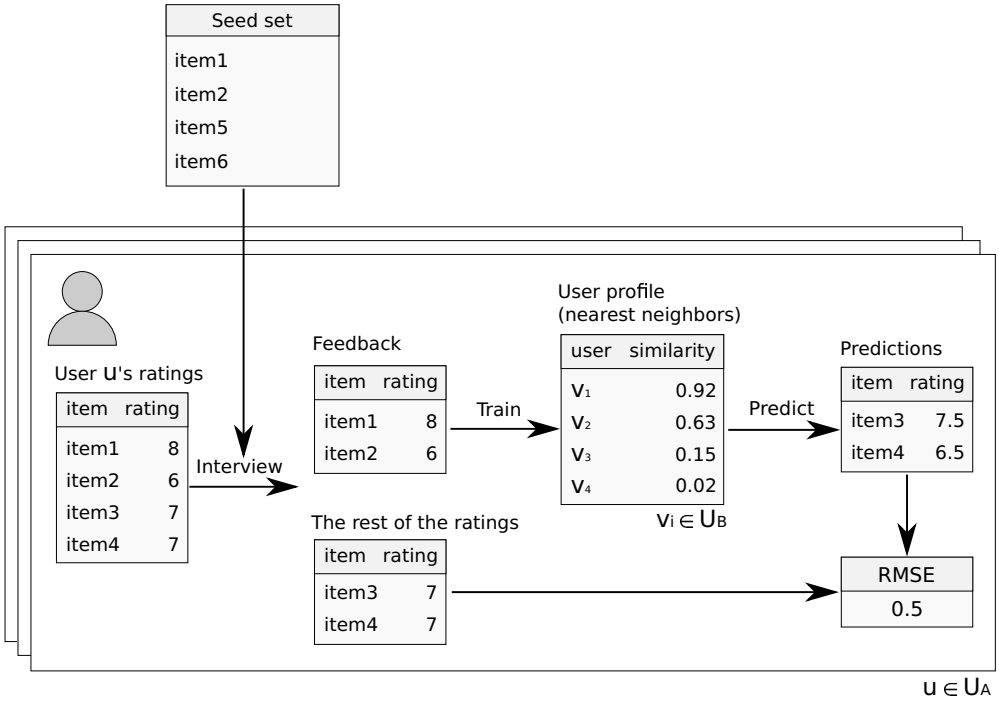

Fig. 2. An example showing how seed set is evaluated in an off-line new user interview. In an evaluation on test set, $U_A = U_{test}$ and $U_B = U_{train}$. In an evaluation on validation set, $U_A = U_{valid}$ and $U_B = U_{train}$

A. New User Cold Start

A typical cold start process for a new user consists of 3 procedures:

1. A set of items (the seed set) are presented to the user for rating. The user rates every item she knows, and leaves the rest of the item unrated.
2. The system trains an initial profile for the user based on her feedback in step 1.
3. The system makes recommendations for the user based on the profile.

The utility of seed set is usually measured by the test prediction error of the recommender system after preference eliciting.

In an online experiment, the system can use other feedback the user expresses after the interview to evaluate the utility of the seed set. One can carry on an A/B test to compare different seed set generating strategy.

However, such an experiment is resource-demanding and requires a large number of human participants. Thus, we conduct an agent-based off-line experiment to simulate this process.

In an off-line experiment, as shown in Figure 2, we make use of an explicit feedback dataset in which every record is a triple (u, i, r) which means a user u rates an item i with value r. The users are split into two different set: U_{train} for system training and U_{test} for testing. We use agents representing users in the new user interview. An agent that simulates a user knows all her ratings. The agent rates every item that the user rates in the seed set with the same value as the user. The system uses such feedback to build an initial profile for the agent and make predictions based on it. The ratings not used in the interview are used in the evaluation of the prediction error.

We use the root-mean-square error (RMSE), a frequently used measurement of the differences between predicted values and the sample values, to measure the prediction error. RMSE for a user u is computed by Eq. 1, in which r denotes the actual ratings while $\hat{r}$ denotes the predicted values. The computation is limited to items rated by user u but not in the seed set S.

$$RMSE(r_u, \hat{r}_u) = \sqrt{\frac{1}{|I_u \setminus S|} \sum_{i \in I_u \setminus S} (r_{ui} - \hat{r}_{ui})^2} \quad (1)$$

The prediction RMSE only depends on the seed set S if the recommender system RS, the user set U_{train} and U_{test} are fixed. Thus, we denote the aggregated prediction error by $RMSE(S)$, as shown in Eq. 2.

$$RMSE(S) = \frac{1}{|U_{test}|} \sum_{u \in U_{test}} RMSE(r_u, \hat{r}_u) \quad (2)$$

Lower prediction RMSE is a sign of more accurate user modeling which means that the seed set allows the system to know the new users better.

B. Generating Seed Set

Good seed set can improve the quality of recommendation and at the same time lessen user efforts. A lot of heuristic cri-

teria are proposed to select items by their item-wise properties such as *Popularity* [4], *Entropy* [4], and *Entropy0* [5].

1) Popularity: New users are more likely to give the system feedback on popular items. Therefore, presenting popular items in the interview might increase the number of ratings collected. This is the intuition of *Popularity* criteria. The downside of this criteria is that popular items might be loved by everyone and carry less information than less popular but more controversial ones.

2) Entropy: Entropy (Eq. 3) is the amount of uncertainty that one variable contains. *Entropy* strategy favors items that are controversial; the rating distribution of such items tend to be very polemic. *Entropy* is flawed in that it overlooks the possibility that a new user might not know of such item. For example, *Entropy* strategy will not distinguish an item with 1000 1-star and 1000 5-star ratings from one with one 1-star and one 5-star ratings, despite the intuition that the former one is more suitable for new user interview.

$$H(i) = -\sum_{v=1}^{5} p_{r_i=v} log(p_{r_i=v}) \tag{3}$$

3) Entropy0: In a system where rating scales from 1 to 5, a 5-star rating usually stands for fondness; on the opposite, a 1-star usually means dislike. But what if there is no rating from a user to an item? Such ratings are called missing values. Missing values convey unclear but useful information. *Entropy0* takes missing values into consideration in the computation of information entropy as shown in Eq. 4. Every possible value v of a rating is weighted by w_v. A missing value is denoted by 0 since common rating scale is from 1 to 5. The weight w_0 for missing values is set to a smaller value than the other weights to limit the influence of missing values. Note that in Eq. 4 if we set w_0 to 0 and other weights to 1, and multiply the equation with a factor of 5, we get Eq. 3.

$$Entropy0(i) = -\frac{1}{\sum_{v=0}^{5} w_v} \sum_{v=0}^{5} p_{r_i=v} w_v log(p_{r_i=v}) \tag{4}$$

4) Var: This strategy is proposed in a post in Netflix prize forum[1]. The original intention is to find movies that are "both universally loved and universally hated by different subgroups of people". The measure uses standard deviation of ratings regarding to item i to select movies that are both love and hated, and amplifies it with the square root of popularity of item i to introduce universality.

$$Var(i) = \sqrt{|U_i|} \times \frac{1}{|U_i| - 1} \sqrt{\sum_{u \in U_i} (r_{ui} - \bar{r_i})^2} \tag{5}$$

The heuristic is named "contention" in the original post, and "*Var*" in [7]. Experiment [7] shows that this heuristic outperform all other baseline methods. We follow such annotation and call this measure *Var*.

[1]http://netflixprize.com/community/viewtopic.php?id=164

5) GreedyExtend: This strategy considers the set as a whole and optimizes the seed set towards a direct goal [7]. It minimizes the training prediction error of recommender system bootstrapped by the seed set on the training set. To generate a seed set containing k items, we hope to find a seed set S that the prediction RMSE of recommender system RS after bootstrapping with seed set S is minimized (Eq. 6). In Eq. 6, $RMSE(S)_{valid}$ stands for the prediction error on validation set after new user interview using the seed set S in cold start process. An illustration of the evaluation of prediction error on validation set is shown in Figure 2. Such an evaluation is similar to one conducted on test set. This optimization strategy implies that a seed set that performs well on training set will also does well on test set.

$$S_k = \underset{S \subset I, |S|=k}{argmin}\, RMSE(S)_{valid} \tag{6}$$

Such optimization is hard in reality, since the time complexity of a traversal is $O(\binom{|U|}{k})$. Therefore, [7] uses a greedy algorithm that extends the current "best" seed set by searching for one "best" additional item at a time and names it *GreedyExtend* (Eq. 7-8).

$$S_0 = \{\} \tag{7}$$

$$S_{k+1} = \underset{i \in I \setminus S_k}{argmin} RMSE(S_k \cup \{i\})_{train} \tag{8}$$

Experiment shows [7] that it can achieve lower RMSE than heuristic strategies including *Popularity*, *Entropy*, *Entropy0*, and *Var*.

III. OUR APPROACH

Heuristic approaches tend to select items by their item-wise properties. However, correlations between items lead to redundancy in seed sets. Take strategy *Var* for an example, it chooses the top ones by an item-wise property. Figure 3 presents an example extracted from MovieLens data set. The figure shows how *Var* strategy introduces redundancy and fails to find the optimal seed set.

GreedyExtend addresses this problem by greedily iterating over every item and finding the best additional item to the currently found "best" seed set. However, once an item is selected, it has no way out. This prevents *GreedyExtend* from finding better seed sets which do not contain certain items already in the "best" seed set, as shown in Figure 4.

Note that in the example shown in Figure 4, the optimal two-item seed set is not a superset of the optimal one-item seed set. This shows that the optimization problem of seed set does not have optimal substructure. Therefore, dynamic programming methods are also not applicable in finding the optimal seed set.

To address this problem, we decide to employ a random searching algorithm, which enables item replacement and is able to iteratively adjust the seed set. Our approach is based on random mutant hill climbing.

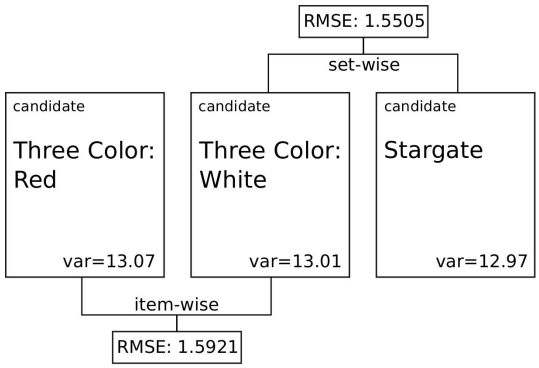

Fig. 3. How an item-wise approach as *Var* generates a redundant seed set when we select two items as the seed set. *Three Color: White* and *Three Color: Red* are series movies that are highly correlated. But *Var* selects them since they rank the highest. A set-wise approach that treats the seed set as a whole might avoid such redundancy.

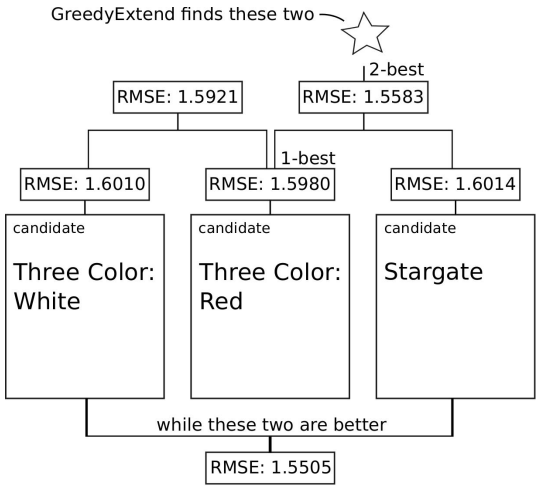

Fig. 4. How *GreedyExtend* will miss better seed sets due to greedy selection. *GreedyExtend* selects *Three Color: Red* for the best seed set with one item and it cannot discard this item in the selection of seed set with two items, so it misses a better option.

Random Mutant Hill Climbing (RMHC)

RMHC is a simple hill-climbing algorithm that outperforms a genetic algorithm on a special problem called "Royal Road" [8]. The algorithm keeps record of one best individual and replace it with a randomly mutated one if the mutated one is better. Mutation introduces random changes to the current best individual and allows the algorithm to constantly adjust the individual.

Our random search method is based on *RMHC*. It adopts the same optimization goal that reduce training prediction error of recommender system using the seed set as described in [7]. The algorithm keeps one seed set as the currently found best one. It mutates a seed set by replacing one item in the seed set with one out of the set. If the mutation leads to a lower prediction error on training set, the new seed set will replace the old one, otherwise it is discarded. In this way redundant items have a chance to be replaced by better ones and other items in the seed set will remain intact. Instead of choosing a randomly selected seed set, we use the seed set generated by *Entropy0* as the starting point of hill climbing for the sake of computational efficiency. The detailed process is described in Algorithm 1.

generate an initial seed set $S_{Entropy0}$;
while *stop criteria not met* **do**
  randomly select an item from the set $i_d \in S_{best}$;
  randomly select an item out of the set $i_a \in I \setminus S_{best}$;
  $S_{new} = (S_{best} \setminus \{i_d\}) \cup \{i_a\}$;
  if $Err(RS(S_{new})) < Err(RS(S_{best}))$ **then**
    $S_{best} \leftarrow S_{new}$;
  end
end
return S_{best};

Algorithm 1: Random Mutant Hill Climbing for Seed Set Optimization

For most hill climbing algorithm, the process stops when the algorithm can not make enough improvement in a certain amount of steps. But we adopt a simpler stop criteria: we limit the total steps of hill climbing. This allow us to control the budget of computational cost.

IV. EXPERIMENTS

A. Evaluation Framework

The data used in our experiments is the MovieLens 10M data set. It contains 10,000,000 ratings (on a scale of 1 to 5) from 72,000 users on 10,000 movies. We randomly select 60% of the users and use their ratings as the training set R_{train}. Ratings from another 10% of the users are used as the validation set R_{valid} and ratings from the rest 30% users are used as the test set R_{test}.

We implement several baseline approaches: *GreedyExtend*, *Var*, *Entropy0*, and *Popularity*. They are compared to our approach *RMHC* with regard to both prediction error and computational cost.

Our evaluation framework can be illustrated by Figure 2. Every user in the test set is evaluated separately. We assume that the agent simulating a user will complete the interview. Every item in the seed set in the interview will be assigned either a rating (if the user has rated the item) or a blank value. The ratings to the items outside the seed set are used to evaluate the prediction error of the system. Such assumption is similar to the real bootstrapping process for new users, except that in real systems new users may abort the interview, which is beyond our topic.

Since the user does not need to rate items that she does not know about, the rating effort needed in the interview is often much less than the size of the seed set. Thus, such bootstrapping process does not necessarily place a burden on the user.

We use root mean square error (RMSE) to evaluate our seed set selection algorithms. It is the same metrics used by *GreedyExtend* [7].

In our experiment, the predictions in both training and test phases are made with factored item-based collaborative

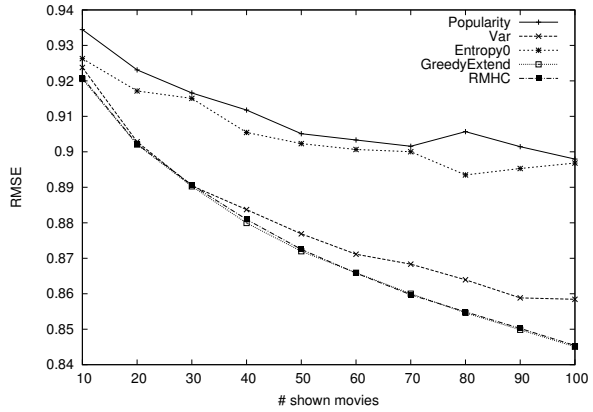

Fig. 5. The test prediction error vs. number of movies shown, which is also the size of seed set. Among the baselines, *GreedyExtend* is superior to the others. *RMHC* performs as well as *GreedyExtend* does.

filtering[9]. Whereas, our experiment framework is still valid using other prediction algorithms.

B. Experiment Setup

We introduce all parameter used in the experiment in this section. Heuristic approaches including *Var*, *Entropy0*, *Popularity*, and *Random* do not need any parameter. *GreedyExtend* cost a fixed number of computation and need no parameter. And *RMHC* has a tunable climbing steps limit C, which is related to the performance of the algorithm.

Since *RMHC* performs one evaluation in every climbing step, the limit of steps in *RMHC* is set to $C = 1500 \times size(seedset)$. This makes the steps of evaluation of *GreedyExtend* and *RMHC* comparable.

C. Results and Discussions

We compare *RMHC* with various baselines at 10 different seed set length from 10 to 100 to trade off between detailed comparison and costly computation.

Figure 5 shows the performances of five baselines and *RMHC* under different seed set lengths. It turns out that generally *GreedyExtend* is the best baseline method in our experiment. *RMHC* consistently reaches very low prediction error as *GreedyExtend* does.

Unlike *GreedyExtend* whose computational steps are limited by the number of items and the size of the seed set, *RMHC* can stop optimizing early to compromise between performance and computational cost. Figure 6 shows the prediction error on validation set with different amount of computational steps. It takes *RMHC* much less computation steps to lower the prediction error from the starting point to a good point.

We conclude that *RMHC* is more efficient in generating high performance seed set for initial user modeling. We attribute the efficiency to a good selection of starting point and the effectiveness to the random optimization that allows for constant adjustments.

Our approach can also be deployed in parallel. It takes a few hundred steps of evaluation in average to search for

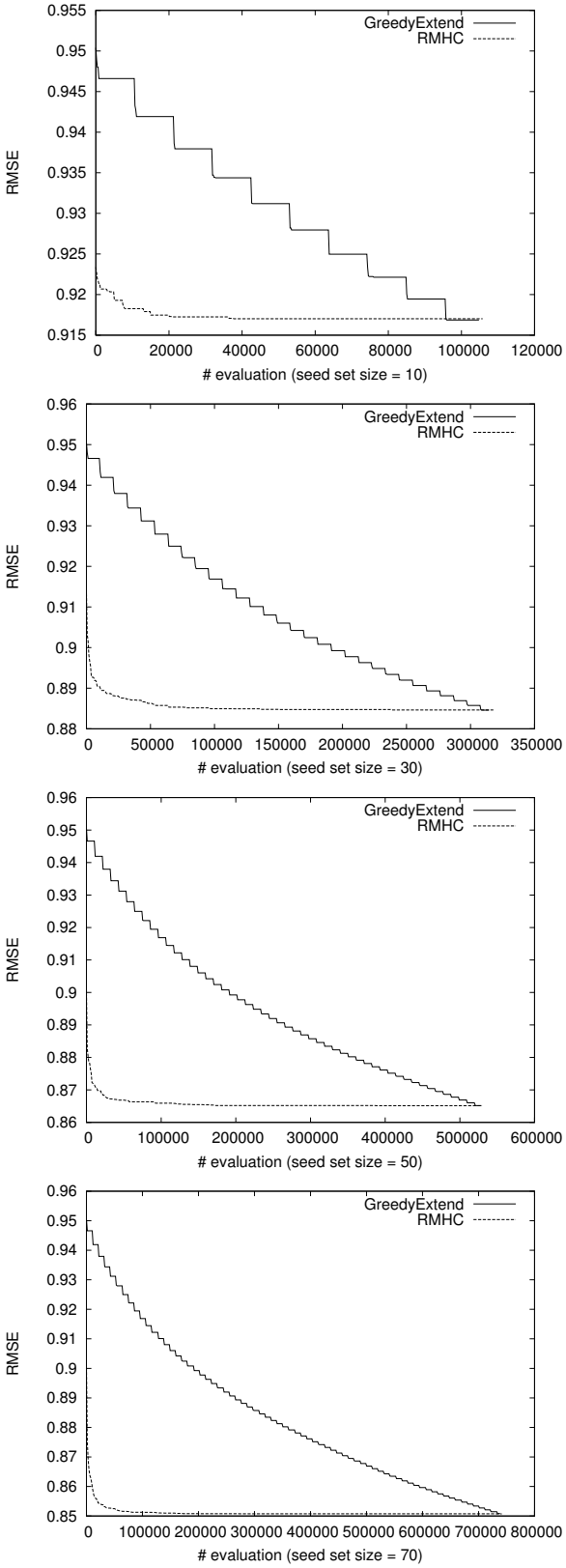

Fig. 6. The prediction error on validation set versus the number of evaluation steps taken when generating a seed set.

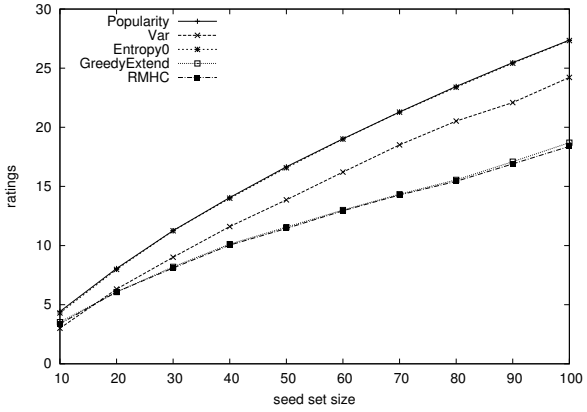

Fig. 7. Average number of ratings expressed in new user cold start. Typically, it is considered acceptable to collect 20 to 30 ratings.

one item replacement that leads to a better seed set. These searching steps are independent of each other and can run in parallel.

D. User Effort Analysis

A usually expressed concern about preference eliciting for new users is that it may place extra burden on the user. A first impression is that any barrier or bootstrapping process during user sign-up may be annoying.

However, various researches [5], [10], [11] shows that what the users want is not minimized burden, but better recommendations. According to an online experiment conducted by [5], most of the users reported that after rated 20 movies they found it "easy" to rate movies. The experiment also found that its users rather understand the benefit of rating more items as the only way to learn their preferences. Another interesting experiment [10] shows that "if a more demanding rating process is balanced by significantly better recommendations, the global satisfaction is not affected negatively by the increased effort".

Also, some researches shows that preference eliciting is beneficial for improving user engagement. In an experiment conducted by [11], the users actually report that they find the rating process entertaining with a statistical significance of 1%. Another user study [2] shows that the users in the group going through a more time-consuming process is more likely to be active users after 25 days, which is quite against the common opinion.

The average number of ratings collected from a user is often 20 [5] or 30 [2]. Thus we assume that 20 to 30 ratings are acceptable burden in preference eliciting. We compared the average numbers of ratings collected while bootstrapping using different seed sets(Figure 7). As we see, all the measures meet such a standard when the size of seed set is less than 100. *RMHC* and *GreedyExtend* are the least burdensome methods among the five ones.

V. CONCLUSIONS

In this paper, we discuss the problem of seed set optimization. We point out that heuristic method will not generate quality seed set since it overlooks the correlation between items. We also point out that a greedy algorithm as *GreedyExtend* may not work well since the optimization problem of seed set does not have optimal substructure. We address this problem by a random optimization method based on random mutant hill climbing. Experiments on MovieLens data set show that our approach performs as well as the state-of-the-art *GreedyExtend* method with significantly less computation. To achieve a prediction performance as good as *GreedyExtend*, our approach needs significantly less computational cost. Our work provides the evidence that random optimization is more efficient than greedy selection in seed set optimization.

REFERENCES

[1] W. S. Lee, "Collaborative learning for recommender systems," in *In Proc. 18th International Conf. on Machine Learning*. Morgan Kaufmann, 2001, pp. 314–321.

[2] S. McNee, S. Lam, J. Konstan, and J. Riedl, "Interfaces for eliciting new user preferences in recommender systems," in *User Modeling 2003*, ser. Lecture Notes in Computer Science, P. Brusilovsky, A. Corbett, and F. de Rosis, Eds. Springer Berlin Heidelberg, 2003, vol. 2702, pp. 178–187. [Online]. Available: http://dx.doi.org/10.1007/3-540-44963-9_24

[3] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel, "Probabilistic memory-based collaborative filtering," *IEEE Trans. on Knowl. and Data Eng.*, vol. 16, no. 1, pp. 56–69, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2004.1264822

[4] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: Learning new user preferences in recommender systems," in *Proceedings of the 7th International Conference on Intelligent User Interfaces*, ser. IUI '02. New York, NY, USA: ACM, 2002, pp. 127–134. [Online]. Available: http://doi.acm.org/10.1145/502716.502737

[5] A. M. Rashid, G. Karypis, and J. Riedl, "Learning preferences of new users in recommender systems: An information theoretic approach," *SIGKDD Explor. Newsl.*, vol. 10, no. 2, pp. 90–100, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1540276.1540302

[6] H. J. Ahn, H. Kang, and J. Lee, "Selecting a small number of products for effective user profiling in collaborative filtering," *Expert Systems with Applications*, vol. 37, no. 4, pp. 3055 – 3062, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417409008069

[7] N. Golbandi, Y. Koren, and R. Lempel, "On bootstrapping recommender systems," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ser. CIKM '10. New York, NY, USA: ACM, 2010, pp. 1805–1808. [Online]. Available: http://doi.acm.org/10.1145/1871437.1871734

[8] S. Forrest and M. Mitchell, "Relative building-block fitness and the building-block hypothesis," *Ann Arbor*, vol. 1001, p. 48109, 1993.

[9] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discov. Data*, vol. 4, no. 1, pp. 1:1–1:24, Jan. 2010. [Online]. Available: http://doi.acm.org/10.1145/1644873.1644874

[10] P. Cremonesi, F. Garzottto, and R. Turrin, "User effort vs. accuracy in rating-based elicitation," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, ser. RecSys '12. New York, NY, USA: ACM, 2012, pp. 27–34. [Online]. Available: http://doi.acm.org/10.1145/2365952.2365963

[11] F. M. Harper, X. Li, Y. Chen, and J. A. Konstan, "An economic model of user rating in an online recommender system," in *Proceedings of the 10th International Conference on User Modeling*, ser. UM'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 307–316. [Online]. Available: http://dx.doi.org/10.1007/11527886_40