Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Comparison-based interactive collaborative filtering

Yuval Carmel, Boaz Patt-Shamir [*,1]

School of Electrical Engineering, Tel Aviv University, Tel Aviv 6997801, Israel

A R T I C L E I N F O

Article history:
Received 7 September 2015
Received in revised form 20 January 2016
Accepted 7 March 2016
Available online 10 March 2016
Communicated by M. Latapy

Keywords:
Parallel algorithms
Recommender systems
Social networks
Collaborative filtering

A B S T R A C T

In this work we study the interactive model of comparison-based collaborative filtering. Each *player* prefers one *object* from each pair of objects. However, revealing what is a player's preference between two objects can be done only by asking the player specifically about that pair, an action called *probing*. The goal is to (approximately) reconstruct the players' preferences with the smallest possible number of probes per player. The per-player number of probes can be reduced if there are many players who share a similar taste, but a priori, players do not know who to collaborate with. In this work, we present the model of comparison-based interactive collaborative filtering, analyze a few possible taste models and present distributed algorithms whose output is close to the best possible approximation to the players' taste.

1. Introduction

Recommendation systems have become a significant part of our lives in the past few years. Most people encounter recommendation systems on a daily basis, while buying a book, choosing which movie to watch, buying groceries in the supermarket, or even finding a life mate. Collaborative Filtering is one of the prevalent approaches to recommendation systems, especially large scale systems (such as Netflix [7]). The idea in collaborative filtering is to take advantage of the existence of many players with similar preferences which can collaborate by sharing the load of trying the objects and identifying objects they perceive as good.

Following [4,5,10,17,20], we distinguish between *interactive* and *non-interactive* recommendation systems, which differ in assumption and usage. In non-interactive recommendation systems, the algorithm is fed all known preferences as collected from the users in the past, and the goal is to output (possibly few) unknown preferences. This model is very popular, and conceptually easy to implement, but it does not take into account the dynamics of the system after the output is made.

In interactive recommendation systems (first introduced as "competitive" in [10]), on the other hand, while the goal remains to predict preferences, it is assumed that no preference is known a priori to the system or the users, but information about preferences can be found by asking the user to try some objects, in an action called *probe*. For example, a probe may ask the user to read a book and ask him[2] for his opinion on the book. The results of probes not only determine the predictions the algorithm makes, but also determine the identity of future objects to be probed. The goal of interactive algorithms is to predict players' preferences while minimizing the number of probes, since probing is assumed to be costly. In the interactive model, the system doesn't suffer from the cold start problem [14,21] since items are being recommended

[*] Corresponding author.
E-mail addresses: yuval9859@gmail.com (Y. Carmel), boaz@tau.ac.il (B. Patt-Shamir).

[1] Supported in part by a grant from the Israel Ministry of Science, Technology and Space and the French Ministry of Higher Education and Research.

[2] Or her. For uniformity, we have arbitrarily chosen to refer to players as males.

http://dx.doi.org/10.1016/j.tcs.2016.03.010
0304-3975/© 2016 Elsevier B.V. All rights reserved.

to the users continuously throughout the running of the algorithm (we treat querying objects in the interactive model as recommending new items to the user while optimizing the next recommendations by the user's probe result).

The interactive model is as follows (cf. Section 2). There are n players and m objects; for each player, there are preferences over the objects, represented by his *preference vector* or *taste*. An entry in the vector of a player can be revealed only by asking that player to preform the appropriate probe. (The player is not assumed to know his preferences in advance, but to discover the result of each probe once it is made.) Probe results are assumed to be posted on a shared "billboard" (modeling eBay feedback records, IMDb reviews, etc.), so that each player can run his algorithm to find which probe to do next, as well as compute preference predictions.

The existence of a billboard does not solve the problem, since players still need to decide whose results to adopt. We assume that some tastes are popular, namely many players have them. Concretely, given a *popularity factor* $0 < \alpha \leq 1$, and a *distance parameter* $D \geq 0$, we say that a preference vector v_j is (α, D)*-popular* if there are at least αn players whose preference vectors differ from v_j by at most D entries. Note that in order to reconstruct any taste (preference vector), $\Omega(m)$ probes are required just to cover all objects, and hence, to reconstruct his preferences to within $O(D)$ errors, the average number of probes per player with an (α, D)-popular taste cannot be less than $\Omega(\frac{m-D}{\alpha n})$.

In most previous work, preferences are simply an absolute grade for each object. The grades are usually binary or decimal with the interpretation of "the user like/dislikes the object a" or "the object a belongs to the user's top x objects" respectively. In this work we introduce a *comparison-based* interactive collaborative filtering model (talking about comparison-based non-interactive CF is out of context, since the comparison-based queries affect the users' ranking), where preferences are expressed only over pairs of objects, with the interpretation of "the user prefers object a over object b". In the interactive model, a comparison-based probe means that the user is presented with two objects, and responds with his preference between them (which may also include "equal" or "incomparable"). We note that it is well known that comparison-based preferences are more intuitive, consistent and accurate than absolute grading (we elaborate below). However, it is not quite clear what can be assumed about the structure of comparison-based preferences. In this work we study a few simple user models. The simplest model is that the user preference between a pair of objects is independent of his preferences over other pairs, and possibly the most structured model is when the pairwise preferences are induced by an underlying total order over all objects. In between, one can consider pairwise preference induced by partial order.

Our contribution The main technical contribution of this work is a comparison-based algorithm for reconstructing preferences induced by an underlying total order. First we present Algorithm $\mathcal{DP}$ (Section 3) for instances with distance parameter $D = 0$. With high probability, Algorithm $\mathcal{DP}$ reconstructs $(\alpha, 0)$-popular preference vectors exactly, incurring at most $O(\frac{1}{\alpha}\log n(\log\log n + \log\frac{1}{\alpha}))$ probes per player, assuming $m = n$.[3]

Our main result is Algorithm $\mathcal{DPD}$ (Section 4), that uses Algorithm $\mathcal{DP}$ as a subroutine, and solves problem instances with distance parameter $D > 0$ w.h.p. Algorithm $\mathcal{DPD}$ reconstructs (α, D)-popular preference vectors with at most $O(D)$ errors and using at most $O(\frac{D^2}{\alpha}\log^2 n\log\frac{\log n}{\alpha}))$ probes per player. We also consider the case where each user perceives the object set as a disjoint union of a few *object categories*, such that objects within a category are totally ordered, but objects in different classes are incomparable. This model is appropriate in the case that the object set is eclectic, e.g., cars and restaurants. We show how to reconstruct the taste in this case without prior knowledge on the categorizations, and even when different users have different categorizations.

Related work Collaborative filtering is studied quite intensively, but mostly from the non-interactive perspective (see, e.g., [7,11,18], which usually use different algorithms to predict users preference on objects based on previous data). The interactive model was introduced by Drineas et al. [10] (referred to there as "competitive recommendation systems") and developed by many over the years [4,5,17,20]. In the absolute grade model, where a preference vector specifies a grade for each object, it was shown by Awerbuch et al. [5], that all $(\alpha, 0)$-popular preferences can be reconstructed exactly, using $O(\frac{1}{\alpha}\log n)$ user probes (assuming for simplicity that the number of objects roughly equals the number of users, i.e., $m = \Theta(n)$). Alon et al. [4] extend this result with instances in which the distance parameter is $D > 0$ and provide a reconstruction for all (α, D)-popular users preferences using $O(\frac{1}{\alpha}D^{3/2}\log^2 n)$ user probes with at most $O(D)$ errors. In addition, [4] provides an algorithm which reconstructs all (α, D)-popular tastes with $O(D/\alpha)$ errors using $O(\log^{3.5} n/\alpha^2))$ user probes. Nisgav and Patt-Shamir [17] improve these results, presenting algorithms for reconstructing (α, D)-popular users preferences with $O(D)$ errors and probe complexity either $O(\frac{D}{\alpha}\log^2 n)$ or $O(\frac{1}{\alpha}\log^3 n)$.

Comparison-based recommendation systems are considered superior to absolute grading systems in stability and more natural for human interaction based ratings. For example, in an experiment held by Jones et al. [12], user preferences expressed by comparisons were measured as 20% more stable over time than preferences expressed by 5-star grading scale. Since a 10% increase in accuracy is considered significant (that was the goal of the million-dollar Netflix challenge [7]), a 20% reduction in inconsistency is clearly meaningful in this context. Moreover, users tend to prefer comparison-based grading, finding it more intuitive than 5-star rating-based grades [8,12].

It is therefore not surprising that there are some proposals for comparison-based model recommendation systems. Loepp et al. [15,16] provide an interactive collaborative filtering algorithm which uses a priori knowledge about the objects to op-

[3] In general, the probe complexity results should be multiplied by $\lceil m/n \rceil$. For simplicity, we usually omit this factor and assume $m = n$.

timize the estimations of user preferences. Desarkar et al. [9] provide a comparison-based non-interactive algorithm to predict preferences using matrix factorization. Both [9] and [15] present experimental results, which show that comparison-based preferences perform well against other user correlation methods in collaborative filtering algorithms. From the theoretical-algorithmic viewpoint, Ailon [1] gives an active learning algorithm to produce approximate ranking from pairwise preferences, which reconstructs a single vector (equivalent to a single taste) using $O(n \cdot \mathrm{polylog}(n, \epsilon^{-1}))$ queries, where $0 < \epsilon < 1$ is the error tolerance parameter. However, to the best of our knowledge, the current work is the first to present an interactive comparison-based recommendation system with worst-case guarantees.

Our techniques Our algorithms use ideas from the interactive non-comparison-based algorithms of [3,5]. We also use the parallel merge sort algorithm of Valiant [19] to speed up computation, and the approximation algorithm for minimum feedback arc set in tournaments from [2,13] to obtain approximate total order.

Organization The remainder of this paper is organized as follows. In Section 2 we define the comparison-based model and on the different possible user models. Then, focusing on the total order case, in Section 3 we give our algorithm $\mathcal{DP}$ for problem instances with distance parameter $D = 0$, and in Section 4 we give our second algorithm $\mathcal{DPD}$, which solves problem instances with distance parameter $D > 0$. In Section 5 we consider the case where objects can be classified into disjoint categories.

2. Preliminaries

2.1. Problem definition

Instances An instance of the comparison-based reconstruction problem is defined as follows. There is a set P of n *players* (a.k.a. *users*), and a set O of m *objects*. For every two objects $i, i' \in O$ and player $j \in P$, there is a *comparison value* $c_j(i, i') \in \{-1, 0, 1, \perp\}$, called the *personal preference* of player j over objects i, i', interpreted as follows. If $c_j(i, i') = 1$ then player j prefers i over i', and if it is -1 then player j prefers i'; if $c_j(i, i') = 0$ then player j likes i and i' the same, and if $c_j(i, i') = \perp$ then player j cannot compare objects i and i'. We define the vector $v_j := (c_j(1, 2), c_j(1, 3), \ldots, c_j(m, m-1))$ of length $\binom{m}{2}$ to be the *preference vector* or *taste* of player j.

Distances and popularity Fix an instance of the problem, and let $O' \subseteq O$ be an object set. The *distance* between two players j, j' w.r.t O' is defined by

$$\mathrm{dist}_{O'}(v_j, v'_j) = |\{(i, i') : i, i' \in O' \wedge c_j(i, i') \neq c'_j(i, i')\}| \, .$$

Namely, $\mathrm{dist}_{O'}(v_j, v'_j)$ is the number of object pairs from O' on which j and j' disagree. A *taste* is a vector v containing a comparison value $c(i, i')$ for each pair $(i, i') \in O$. Given $0 < \alpha \leq 1$ and $D \geq 0$, a taste v is (α, D)*-popular* if there are at least αn players, whose taste is at distance at most D from v. A player is (α, D)*-popular* if his taste is (α, D)-popular. Note that for any taste v and $0 \leq \alpha \leq 1$ there is a $0 \leq D \leq \binom{m}{2}$ such that v is (α, D)-popular, and similarly, for any given $0 \leq D \leq \binom{m}{2}$ there exists an $0 \leq \alpha \leq 1$ such that v is (α, D)-popular.

Algorithms An Algorithm proceeds in parallel rounds, where in each round the algorithm may present to each player j a pair of objects (i, i') and obtain $c_j(i, i')$. This action is called a *probe* by player j. Players communicate through a shared billboard, i.e., all probes results are immediately visible to all players. An algorithm makes output and chooses which objects to probe based on the results of all past probes (including probes taken by other players), and possibly coin tosses. The algorithms we consider in this work are required to output preference vectors that approximate the pairwise preferences of the players, up to some specified number of errors. The maximal number of probes a player is asked to perform in an algorithm execution is the *probe complexity* of the algorithm. Our algorithms are randomized and all guarantees are *with high probability*, i.e., holds with probability $1 - n^{-\Omega(1)}$, where probability is taken over the coin tosses of the algorithm.

We assume random partitioning and sampling is always done uniformly. Splitting a set in two, for instance, is done by choosing, for each element, its part independently with probability 1/2.

2.2. User models

We consider the following variants of the comparison-based preferences model.

General model In the general model, the $\binom{m}{2}$ comparison values of a player are completely independent. In particular, no transitivity of preferences is assumed. This model is equivalent to a model of $\binom{m}{2}$ virtual objects where each virtual object represents a pair of two physical objects, with four possible grades. For an upper bound, one can apply Algorithm 1 of Azar et al. [6] for recommendations with discrete grades, which reduces the problem to the binary grade model. Plugging in that algorithm the algorithm of [17] for binary recommendations, we obtain the following result.

Theorem 2.1. *In the comparison-based model, any (α, D)-popular taste can be reconstructed to within $O(D)$ errors in probe complexity $O\left(\left\lceil \frac{m^2}{n} \right\rceil \frac{1}{\alpha} \log^3(m+n)\right)$.*

The general model also admits easy lower bounds. In particular, the total probe complexity (the sum of individual probe complexity over all players) in the general model is $\Omega\left(\left\lceil \frac{m^2-D}{n} \right\rceil\right)$. Intuitively, even in the case where all players of some (α, D)-popular taste are perfectly coordinated, each individual object (with the exception of at most D objects) must be probed at least once. Therefore, $\Omega(\binom{m}{2} - D)$ total probes must be taken by these players just to cover all pairs. Next, consider the case where all players know all the possible tastes in advance (they are common knowledge). Still, to find his own taste, each user must make at least one probe, giving rise to total probe complexity of at least $\Omega(n)$. Hence the average probe complexity is $\Omega\left(\frac{\binom{m}{2}-D+n}{n}\right) = \Omega\left(\left\lceil \frac{m^2-D}{n} \right\rceil\right)$.

Total order model In this model we assume the player taste is induced by a total order over the objects. Comparisons may not have "$\perp$" as a value: $c_j(i, i') \in \{-1, 0, 1\}$ for all players j and objects i, i'. In this model, a preference vector can be represented by a permutation of the objects. We note that in this case distances, as defined above, are just the number of *transpositions* of one permutation with respect to the other.

We have the following straightforward lower bound on the number of probes in this model.

Theorem 2.2. *Consider an (α, D)-popular taste which is a total order. The number of probes for reconstructing it is $\Omega\left(\frac{(m-2D)\log(m-2D)}{\alpha n}\right)$ probes on average.*

Proof. We use a variant of the sphere packing bound. The number of permutations that are at distance at most $2D$ from a given permutation is at most $\sum_{i=0}^{2D} \binom{m}{i} i! \le (2D+1)m^{2D}$. Therefore, if we cannot specify at least $\frac{m!}{(2D+1)m^{2D}}$ distinct inputs, there will be instances containing two tastes at distance greater than $2D$, and both will have the same output, so necessarily one of them is at distance more than D from its output. Since specifying $\frac{m!}{(2D+1)m^{2D}}$ distinct outcomes requires $\Omega((m - 2D)\log(m-2D))$ bits, and since each probe provides only $O(1)$ bits, the result follows. □

Disjoint categories model In this model we assume that the object set can be broken into disjoint *classes*, or *categories*, where only objects in the same category are comparable. The categorization of objects may be different for different tastes.

Specifically, we assume that for each taste, the object set can be partitioned into k disjoint sets $O = \bigcup_{i=1}^{k} O_i$, and that there is a total order σ_i defined over each O_i. The preference between two objects is either $\perp$ if they belong to different subsets, or given by σ_i if they both belong to O_i for some i. This model is appropriate in cases where the objects may be of incompatible nature, e.g., O_1 consists of films and O_2 is a set of pets: it doesn't make sense to compare a film with a pet. Moreover, different tastes may have different categorizations, which allows our algorithm to be applicable even when the categories are unanimously acceptable.

3. Exact total order: algorithm $\mathcal{DP}$

In this section we consider the case where the algorithm receives as input a popularity factor α, and the goal is to reconstruct the preferences precisely for all $(\alpha, 0)$-popular players, i.e., players whose *exact* taste is shared by at least αn players. Most importantly, we assume that these $(\alpha, 0)$-popular tastes are induced by total orders over the objects. Our solution to this case is a randomized distributed algorithm called $\mathcal{DP}$, based on Algorithm Zero_Radius of [4] for the binary grade model. While $D = 0$ is an interesting case in its own right, we shall use Algorithm $\mathcal{DP}$ as a subroutine in the algorithm for $D > 0$, presented in Section 4.

3.1. Description

The algorithm works as follows (see pseudocode in Algorithm 3.1). The input consists of a set of players P and a set of objects O. If either P or O is small enough, each player in the current player set P sorts all objects in the current object set O, using a straight-forward adaptation of any efficient comparison-based sorting algorithm, such as Quicksort, Heapsort, Mergesort, etc. Otherwise, the algorithm randomly splits the object set and the player set into O', O'' and P', P'' respectively, and calls itself recursively: P' with O' and P'' with O''. When the recursive call returns, each player in P' knows his complete preferences (ordering in our case) over O' and each player in P'' knows his ordering over O''. This is done as follows. Looking at the billboard, a player in P' first identifies the preference vectors over O'' with popularity larger than $\alpha/2$ (line 9) as "candidate tastes". From these candidates, he selects the vector compatible with his taste by probing *controversial pairs*, i.e., pairs on which some candidate vectors differ. Formally, for two vectors v_1 and v_2, there exists a pair (i, i'), for which, $i <_{v_1} i'$ and $i >_{v_2} i'$, namely i is preferred over i' in v_2 but not in v_1 (lines 12–13). Each such probe eliminates at least one candidate. Eventually, the player is left with a single vector. This vector, which is an ordering of O'', is then merged with the vector the player has already computed for O'. This is done using the parallel merging algorithm of Valiant [19] (see Appendix A).

Algorithm 3.1 $\mathcal{DP}(P, O)$ reconstruct exact taste.

```
1: if $\min(|P|, |O|) < \frac{16c \ln n}{\alpha}$ then                      ▷ base case
2:     Sort $O$ by order of preference using any reasonable comparison sorting algorithm.
3:     return the sorted preference vector over $O$.
4: end if
5: Randomly partition $P = P' \cup P''$, and $O = O' \cup O''$.
6: if $j \in P'$ then call $\mathcal{DP}(P', O')$                          ▷ Assuming current player is $j$
7: else call $\mathcal{DP}(P'', O'')$
8: end if
   Assume w.l.o.g. that $j \in P'$. Upon returning, $j$ has $v'$ his complete order over $O'$, and sees the order selected by each player $j' \in P''$ on the shared billboard
9: Let $V$ be a set of vectors of $O''$ chosen by at least $\alpha/2$ players from $P''$.
10: while $|V| > 1$ do
11:     Let $C = \{(i, i') \in O'' \times O'' \mid \exists v_1, v_2 \in V$ s.t. $i <_{v_1} i'$ and $i >_{v_2} i'\}$      ▷ $\alpha$-popular tastes over $O''$ with controversial values
12:     Choose an arbitrary pair $(i, i') \in C$.
13:     Let $c = \text{probe}(i, i')$
14:     Remove from $V$ all vectors whose value on $(i, i')$ is not $c$.
15: end while
16: Suppose $V = \{v''\}$.                                                   ▷ $|V| = 1$ w.h.p. if $j$ is in an $\alpha$-popular taste
17: Let $P_j = \{j' \in P' :$ player $j'$ chose vector $v''\}$.
18: return Merge$(P_j, v', v'')$.
```

3.2. Analysis

Consider an α-popular taste. The algorithm's success critically depends on having more than $|P|\alpha/2$ players of that taste in any invocation with player set P. The following lemma ensures this w.h.p.

Lemma 3.1. *Fix an α popular taste v, and let P_v be the set of players of taste v. Then, with probability $1 - n^{-\Omega(1)}$, in all invocations of $\mathcal{DP}(P, O)$ at least $|P|\alpha/2$ of the players in P have taste v, i.e., $|P_v \cap P| \ge \alpha|P|/2$.*

Proof. Observe that the set of players P in any invocation is a random sample of the set of all players. By the Chernoff bound we have that the number of players in any random sample P is

$$\Pr[\#\text{ players of taste } v < \alpha|P|/2] \le e^{-\frac{\alpha|P|}{8}} \le e^{\frac{2c \ln n}{\alpha}} = n^{-2c},$$

because by the code, $|P| \ge \frac{16c}{\alpha} \ln n$. Since the total number of invocations is bounded by n, the result follows from the union bound. □

Next, we state the correctness of Algorithm Merge(P, v_1, v_2) used in line 18.

Lemma 3.2. *Let v_1, v_2 be two sorted vectors with $|v_1| = \Theta(|v_2|)$ and let P be the set of players agreeing on the joint order of the objects in v_1 and v_2. Then the merged sorted vector can be computed using probe complexity $O(\frac{|v_1|}{|P|} \log\log|v_1|)$ per player.*

The algorithm is based on [19] in a straightforward way. See Appendix A for details.

Theorem 3.3. *Under Algorithm $\mathcal{DP}$, with probability $1 - n^{-\Omega(1)}$, all outputs by α-popular players are correct. The number of probes per player is bounded by $O(Y(\log Y + \log\log m))$, where $Y = \frac{\log n}{\alpha}\lceil \frac{m}{n} \rceil$.*

Proof. Fix an α-popular taste v. By Lemma 3.1, w.h.p., all invocations have at least $\alpha/2$-fraction of players whose taste is v. Hence the set V selected in line 9 contains the projection of v onto O'', and it will be the only vector not eliminated by any player from P' whose taste is v. Since this is true symmetrically also for P'', it follows that with high probability, all outputs are correct.

Regarding complexity, note that probing is done by the algorithm only in lines 3, 13, and 18. The probing of line 3 is done according to the sorting algorithm, at the cost of $O(|O|\log|O|)$ probes. Let P and O be the set of players and objects respectively, when the sorting algorithm is executed. Since the base condition is $\min(|P|, |O|) < \frac{16c \ln n}{\alpha}$ and since both the player and the object sets are approximately halved in every recursive call, we have that, with high probability, $|O| = O(Y)$, and hence the number of probes due to line 3 is $O(Y \log Y)$. Note that line 3 is executed once throughout the algorithm by each player.

Next, consider line 9. Since each vector in V represents at least an $\frac{\alpha}{2}$ fraction of the players of P'', $|V| \le \frac{2}{\alpha}$. Since in each iteration of the while loop, at least one vector is removed from V in line 14, we have that each player makes at most $2/\alpha$ probes in line 13 in an invocation of Algorithm $\mathcal{DP}$. Finally, the probing of line 18 is done using Valiant's merging algorithm [19]. By Lemma 3.2, we have that each player makes $O(\frac{m}{n\alpha} \log\log m)$ probes in line 18. It follows that the total number of probes due to lines 9 and 18 in a single invocation of the algorithm is $O(\frac{1}{\alpha} \log\log m)$. Since the number of recursive levels is $O(\log n)$, the result follows. □

Corollary 3.4. *If $m = \Theta(n)$ and $\alpha = (\log n)^{-\Omega(1)}$, the probe complexity of Algorithm $\mathcal{DP}$ is $O\left(\frac{1}{\alpha}\log n \log\log n\right)$.*

We note that the probe complexity of Algorithm $\mathcal{DP}$ is larger than the lower bound of Theorem 2.2 only by a factor of $O(\log\log n)$.

4. Approximate total order: algorithm $\mathcal{DPD}$

In this section we present our main result. As in Section 3, we assume that the taste of each player is induced by a total order on the objects, but whereas previously we assumed that there are αn players whose taste is *exactly* the same, here we require only that the taste is (α, D)-popular, namely there are αn players such that any two players in the set may disagree on the outcomes of at most D comparisons. We use Algorithm $\mathcal{DP}$ from Section 3 as a subroutine, but we can still reconstruct the taste to within $O(D)$ errors with polylogarithmic overhead. Our solution is a distributed algorithm we call $\mathcal{DPD}$, which extends Algorithm ***S*** from [17] to the comparison-based model.

4.1. Description

The algorithm consists of three conceptual steps as follows (see pseudocode in Algorithm 4.1). In the first step, we split the object set uniformly at random into $S = \Theta(D)$ disjoint subsets. We apply Algorithm $\mathcal{DP}$ to each subset by all players. This splitting and application of $\mathcal{DP}$ is repeated $K = \Theta(D \log m)$ times, thus making sure (w.h.p.) that each pair of objects appears in the same subset $\Theta(\log m)$ times. After this step, we have, for each pair of objects, $\Theta(\log m)$ estimates for each player. Next, using these estimates, each player j builds a directed graph $G_j = (O, E_j)$ as follows. There is a directed edge between every pair of objects, whose direction is determined by the majority of outcomes computed in the first step. Note that G_j is a tournament, but it may be inconsistent, i.e., contain cycles. These are eliminated in the last step by applying the algorithm of Ailon et al. [2] to G_j, which finds a 3-approximation to *MFAST*: MFAST is the problem of deleting the minimum number of edges from a tournament so that the result is acyclic.[4] No probing is required for this step. The result is the output of player j.

Algorithm 4.1 $\mathcal{DPD}$ reconstruct approximate taste.

```
Require: $P, O, \alpha, D$
 1: for all $k \in 1, .., K$ do                                          ▷ $K = \Theta(D \log m)$
 2:     Partition $O$ into $S = cD$ disjoint subsets $O = \bigcup_{s\in 1..S} O_s$.     ▷ $c > 8$ is a constant
 3:     for all $s \in 1, .., S$ do
 4:         Invoke $\mathcal{DP}(P, O_s, \alpha/4)$
 5:     end for
 6:     Let $C_j^k(i, i')$ be the reconstructed output of pair $(i, i')$ for player $j$ on round $k$.
 7: end for
 8: For all object pairs $(i, i')$, let $L(i, i')$ denote the set of iterations in which the objects $i$ and $i'$ are in the same subset.
 9: For all $i, i' \in O$, let $C_j(i, i')$ be the majority of $\{C_j^k(i, i') : k \in L(i, i')\}$
10: Let $G_j = (O, E_j)$ be a directed graph, where, $E_j = \{(i, i') : C_j(i, i') \geq 0\}$
11: Invoke MFAST-Approx$(G_j)$ for each player $j \in P$
12: Output the resulted ranking for each player $j$.
```

4.2. Analysis

We first define some notation. Let v_j denote the taste of player j, and $v_j(i, i')$ denote the preference of player j on the pair (i, i'). For each player j, define $P(j) = \{j' \in P : \text{dist}(v_j, v_{j'}) \leq D\}$, namely $P(j)$ is the set of players whose preference vectors differ from j on at most D object pairs.

Now, define, for each player j, the set of object pairs $O(j)$ to be the pairs on which player j agrees with the majority of the players in $P(j)$, i.e.,

$$O(j) = \left\{(i, i') \in O \times O : \left|\left\{j' : v_{j'}(i, i') = v_j(i, i')\right\}\right| > \frac{|P(j)|}{2}\right\}$$

The following lemma helps to get a lower bound on $|O(j)|$.

Lemma 4.1. *For all $0 < \delta \leq 1$ it holds that any player j disagrees with at most a δ-fraction of the players in $P(j)$ on at most D/δ object pairs.*

[4] MFAST stands for Minimum Feedback Arc Set in Tournaments (an NP-hard problem, according to Alon [3]). The approximation algorithm of Ailon et al. [2] has running time $O(|O|\log|O|)$. Kenyon-Mathieu and Schudy [13] give a PTAS for MFAST with running time $O(|O|^6)$).

Proof. By definition, each player in $P(j)$ disagrees on at most D pairs with player j. Hence the total number of disagreements between j and all players in $P(j)$ is less than $|P(j)|D$. Therefore the number of players in $P(j)$ with which j disagrees on more than D/δ pairs is at most $\delta|P(j)|$. $\square$

Next, we show that for any $(i,i') \in O(j)$, in each iteration $k \in L(i,i')$ of algorithm $\mathcal{DPD}$ (cf. line 8), Algorithm $\mathcal{DP}$ computes a correct estimate of $v_j(i,i')$ in line 4 with "good" probability.

Lemma 4.2. *For all $j \in P$, $(i,i') \in O(j)$ and iteration $k \in L(i,i')$, we have that the following holds* $\Pr[C_j^k(i,i') = v_j(i,i')] \geq 1 - \frac{4}{c} > \frac{1}{2}$ *for $c > 8$.*

Proof. Fix player j and an object pair (i,i'). Consider a random subset O_s for which $i, i' \in O_s$. Let $P_s(j)$ be the set of players that agree with player j on all objects in O_s, i.e., $P_s(j) = \{j' : \mathrm{dist}_{O_s}(j,j') = 0\}$. If $|P_s(j)| \geq \frac{\alpha|P|}{4}$, then $\mathcal{DP}$ will return correct results. It therefore suffices to show that $\Pr[|P_s(j)| \geq \frac{\alpha|P|}{4}] \geq 1 - \frac{4}{c}$, where the probability here is over the choice of O_s.

To do that, we first observe that for any $0 < \beta < 1$, we have that if the following holds $\sum_{j' \in P(j)} \mathrm{dist}_{O_s}(j,j') \leq \beta \cdot |P(j)|$, then $|P_s(j)| \geq (1-\beta)|P(j)|$. It follows that it suffices to bound the probability that the sum of distances for players in $P(j)$ is at most $\frac{3}{4}|P(j)|$, i.e., the probability that $\sum_{j' \in P(j)} \mathrm{dist}_{O_s}(j,j') < \frac{3}{4}|P(j)|$.

Let $O_s^* = O_s \setminus \{(i,i')\}$. Consider the random variable $\sum_{j' \in P(j)} \mathrm{dist}_{O_s^*}(j,j')$, namely, the sum of distances between j and all $P(j)$ players, ignoring the pair (i,i'). As $(i,i') \in O(j)$, we have that $\sum_{j' \in P(j)} |v_j(i,i') - v_{j'}(i,i')| < \frac{|P(j)|}{2}$. Clearly,

$$\Pr\Big[\sum_{j' \in P(j)} \mathrm{dist}_{O_s}(j,j') \leq \frac{3}{4}|P(i)|\Big] \geq \Pr\Big[\sum_{j' \in P(j)} \mathrm{dist}_{O_s^*}(j,j') \leq \frac{|P(j)|}{4}\Big].$$

Since distance is measured by object pairs and from the definition of $P(j)$, we have

$$D|P(j)| \geq \sum_{j' \in P(j)} \mathrm{dist}_O(j,j') \geq \sum_{\ell=1}^{S} \sum_{j' \in P(j)} \mathrm{dist}_{O_\ell}(j,j'),$$

which implies that $\sum_{j' \in P(j)} \mathrm{dist}_{O_\ell}(j,j') > \frac{|P(j)|}{4}$ for at most $4D$ subsets O_ℓ. From that we have

$$\Pr\Big[\sum_{j' \in P(j)} \mathrm{dist}_{O_s^*}(j,j') \leq \frac{|P(j)|}{4}\Big] \geq \frac{S-4D}{S} = 1 - \frac{4}{c}.$$

Since we consider (α, D)-popular players, the definition of $P(j)$ implies that $|P(j)| \geq \alpha n$. Therefore, $|P_s(j)| \geq \frac{|P(j)|}{4} \geq \frac{\alpha n}{4}$ with probability $\geq 1 - \frac{4}{c}$. The result follows. $\square$

Lemma 4.3. *W.h.p., each pair (i,i') occurs together in $\Omega(\log m)$ invocations of $\mathcal{DP}$.*

Proof. Consider object i. The probability that i' is chosen to the same subset as i in a given iteration is $\frac{D}{S} = \frac{1}{c}$. It follows that the expected number of subsets they occur together is $\frac{K}{c} = \frac{rD\log m}{c}$ for some $r \geq 0$.

Let X be the number of iterations in which i and i' are in the same subset. Since the partitioning to different subsets is independent between iterations and from the Chernoff multiplicative bound we have that

$$\Pr\Big[X \leq (1-\delta)\frac{rD\log m}{c}\Big] \leq e^{-\frac{rD\log m\delta^2}{2c}} \leq m^{-\frac{rD\delta^2}{2c}}.$$

Therefore, for any $r > \frac{2c}{D\delta^2}$ and $0 < \delta < \sqrt{3} - 1$, the result follows. $\square$

By Lemma 4.2, the probability that each invocation of $\mathcal{DP}$ is successful is at least $1 - 4/c > 1/2$ because we set $c > 8$. By Lemma 4.3, each pair occurs together in $\Omega(\log m)$ invocations of $\mathcal{DP}$. Therefore, by the Chernoff bound and the union bound, we have that w.h.p, the majority value is correct for all pairs for all players.

Lemma 4.4. $\Pr\big[C_j(i,i') \neq v_j(i,i')$ *for some j and $(i,i') \in O(j)$*$\big] \leq m^{-\Omega(1)}$.

We can now summarize the properties of Algorithm $\mathcal{DPD}$ as follows.

Theorem 4.5. *W.h.p, algorithm $\mathcal{DPD}$ predicts for each player $j \in P$ its preference vector with at most $O(D)$ errors. The probe complexity for each player is $O(D^2 \log m \cdot T_{\mathcal{DP}}(n, m/cD, \alpha/4))$, where $T_{\mathcal{DP}}(n,m,\alpha)$ is the probe complexity of Algorithm $\mathcal{DP}$ with n users, m objects and popularity factor α.*

Corollary 4.6. *For $m = \Omega(nD)$ and $\alpha = (\log n)^{-O(1)}$, the probe complexity of Algorithm $\mathcal{DPD}$ for reconstructing an (α, D)-popular taste is*

$$O\left(\frac{m}{n}\frac{D}{\alpha}\log^2 m \log\log m\right).$$

Proof of Theorem 4.5. Fix a player j. For-every object pair $(i, i') \in O(j)$, we have that $C_j(i, i') = v_j(i, i')$ w.h.p. Since this is guaranteed only for object pairs in $O(j)$, and since $|O(j)| \geq m - 2D$ (by Lemma 4.1), we have $\text{dist}_O(v_j, C_j) \leq 2D$, i.e., the number of object pairs on which the results of the majorities differ from the true preferences of j (which is a consistent total order) is at most $2D$. Since the Algorithm *MFAST-Approx* of [2] finds a 3-approximation to the optimal number of edges that need to be reversed, *MFAST-Approx* finds a permutation whose distance from v_j is at most $6D$ errors. The query complexity follows from Theorem 3.3, along with the fact that $\mathcal{DP}$ is invoked $KS = KcD$ times, with $|P| = n$, $|O_s| = \lceil \frac{m}{D} \rceil$ and popularity factor $\alpha/4$. The probability of correctness follows from Lemma 4.2 and Lemma 4.4. □

5. The case of disjoint categories

In many cases, the object set is eclectic in the sense that it consists of a few kinds of objects, e.g., cats and cars. Typically, in these cases one may have preference over pairs of objects of the same kind, but there is no sense in comparing objects of different kinds. The problem becomes more complicated when there may be ambiguity about object classification: For example, some users may classify a jaguar as a cat, while others may classify it as a car.

In this section we model this situation and present algorithms to reconstruct preferences.

5.1. Preference model

A user j is said to have a *categorized taste* if the object set can be partitioned into k disjoint subsets $O = \bigcup_{\ell=1}^{k} O_\ell^j$ called the *categories* of user j, and if j has a total order σ_ℓ^j over each category O_ℓ^j. Formally, if user j has a categorized taste then

$$c_j(i, i') = \begin{cases} 0 & \text{if } i = i' \\ 1 & \text{if } i, i' \in O_\ell^j \text{ for some } \ell \text{ and } \sigma_\ell^j(i) > \sigma_\ell^j(i') \\ -1 & \text{if } i, i' \in O_\ell^j \text{ for some } \ell \text{ and } \sigma_\ell^j(i) < \sigma_\ell^j(i') \\ \perp & \text{otherwise.} \end{cases}$$

For simplicity, we say that users have the same taste if their preferences are identical. However, in the case of categorized tastes, we also define the notion of *crude taste*: two users j and j' are said to have the same crude taste if they have the same object categorization (but not necessarily the same ordering within each category), i.e., $\left\{O_1^j, \ldots, O_k^j\right\} = \left\{O_1^{j'}, \ldots, O_k^{j'}\right\}$.

5.2. Algorithm $\mathcal{DPC}$

Clearly, if the user categorizations are given, one can run algorithm $\mathcal{DP}$ on each user category with all users that have this category; if a taste is α-popular. Formally, we have the following result.

Lemma 5.1. *Suppose that in a given instance, all tastes are categorized. If the categories of all users are known, then preferences can be reconstructed w.h.p. with probe complexity $O((\frac{m}{\alpha n} + k)\frac{\log n}{\alpha}\log(\lceil \frac{m}{n} \rceil \frac{\log m}{\alpha}))$, where k is the number of categories.*

Proof. Consider an arbitrary user, say j. Let $m_\ell = |O_\ell^j|$, i.e., m_ℓ is the size of category ℓ of user j. Let n_ℓ be the number of users with category O_ℓ^j. Note that $n_\ell \geq \alpha n$ because at least all users with the taste of j have category O_ℓ^j, and possibly others too. User j takes part in k invocations of $\mathcal{DP}$, where invocation ℓ has m_ℓ objects and $n_\ell \geq \alpha n$ players participating. By Theorem 3.3, the total probe complexity for user j is therefore

$$\begin{aligned} \sum_{\ell=1}^{k} T_{\mathcal{DP}}(n, m_\ell, n_\ell/n) &= \sum_{\ell=1}^{k} \left\lceil \frac{m_\ell}{n_\ell} \right\rceil O\left(\frac{\log n}{\alpha}\log\left(\left\lceil \frac{m_\ell}{n_\ell} \right\rceil \frac{\log(m+n)}{\alpha}\right)\right) \\ &\leq O\left(\left(\frac{m}{\alpha n} + k\right)\frac{\log n}{\alpha}\log\left(\left\lceil \frac{m}{n} \right\rceil \frac{\log(m+n)}{\alpha}\right)\right). \end{aligned}$$ □

Note that by using algorithm $\mathcal{DPD}$, Lemma 5.1 can be extended to handle (α, D)-popular tastes, so long as each category is shared by at least αn users.

In view of Lemma 5.1, we now consider the question of reconstructing the *crude* taste of the users. This can be done by adapting algorithm $\mathcal{DP}$ to *crude probes*. A crude probe, or c-probe for short, is just a regular comparison probe, except that its return value is either 1 if the two object probed are comparable (no matter what is the comparison result), or 0 is the pair if incomparable (i.e., the comparison probe returns $\perp$). Algorithm $\mathcal{DPC}$ (see Algorithm 5.1 for pseudocode) for reconstructing crude tastes has the same structure as Algorithm $\mathcal{DP}$, but with different base procedure (called Classify, line 2) and different merging procedure (called Combine, line 17). We explain these two now.

Procedure Classify works by picking an unclassified object i_0 and applying c-probe of i_0 against all other unclassified objects. If the result of c-probe $(i_0, i) = 1$ then i is in the same category as i_0, and otherwise i remains unclassified. Clearly the probe complexity of a set O is $O(k|O|)$.

Procedure Combine gets as input two classifications v', v'', each with at most k categories. It applies c-probe for each pair of categories: one from v' and the other from v''. These probes are done using representatives from the categories, i.e., $O(k^2)$ probes need to be executed in total. Moreover, these probes can be split among all players in P_j, resulting in individual probe complexity of $O(k^2/|P_j|)$.

Algorithm 5.1 $\mathcal{DPC}(P, O)$ reconstruct crude taste.

```
 1: if $\min(|P|, |O|) < \frac{16c\ln n}{\alpha}$ then                         ▹ base case
 2:     return Classify($O$).                                                 ▹ see text
 3: end if
 4: Randomly partition $P = P' \cup P''$, and $O = O' \cup O''$.
 5: if $j \in P'$ then call $\mathcal{DPC}(P', O')$                       ▹ Assuming current player is $j$
 6: else call $\mathcal{DPC}(P'', O'')$
 7: end if
    Assume w.l.o.g. that $j \in P'$. Upon returning, $j$ has $v'$ as his classification of $O'$, and sees the classifications of each player $j' \in P''$
 8: Let $V$ be a set of classifications of $O''$ chosen by at least $\alpha/2$ players from $P''$.
 9: while $|V| > 1$ do
10:     Let $C$ be the set of object pairs $(i, i') \in O'' \times O''$ for which there are classifications $v_1, v_2 \in V$ such that $v_1$ classifies $i, i'$ together and $v_2$ classifies
    them in different categories.
11:     Choose an arbitrary pair $(i, i') \in C$.
12:     Let $c =$ c-probe$(i, i')$
13:     Remove from $V$ all classifications whose value on $(i, i')$ does not agree with $c$.
14: end while
15: Suppose $V = \{v''\}$.                                  ▹ $|V| = 1$ w.h.p. if $j$ is in an $\alpha$-popular taste
16: Let $P_j = \{j' \in P' :$ player $j'$ chose classification $v''\}$.
17: return Combine($P_j, v', v''$).                                          ▹ see text
```

We can therefore summarize the properties of Algorithm $\mathcal{DPC}$ as follows.

Theorem 5.2. *W.h.p., Algorithm $\mathcal{DPC}$ outputs the α-popular crude taste for each user with probe complexity $O(k(\lceil \frac{m}{n} \rceil \log n + k) + \frac{\log n}{\alpha})$.*

Proof. The probing of the base case (line 2) costs $O(\lceil m/n \rceil k \log n)$ because in the base case, the number of objects is $O(\lceil m/n \rceil \log n)$. The elimination step (line 12) costs $O(1/\alpha)$ probes in each iteration as before. Regarding the probing due to Combine (line 17), let n_t denote the number of users in P_j in iteration t. Then the total cost of Combine for user j over all iterations is

$$\sum_{t=1}^{\log(\alpha n)} \left\lceil \frac{k^2}{n_t} \right\rceil = O(k^2 + \log(\alpha n)) \sum_{t=1}^{\log(\alpha n)} \frac{1}{\alpha n 2^{-t}} \le O(k^2 + \log n) ,$$

because w.h.p., $n_t = O(\alpha n / 2^t)$. The result follows. □

Corollary 5.3. *For $m = \Theta(n)$, the probe complexity of Algorithm $\mathcal{DPC}$ for reconstructing an α-popular crude taste is $O((\frac{1}{\alpha} + k) \log n + k^2)$.*

6. Conclusions and open problems

In this work we introduced the comparison-based interactive CF model and developed several algorithms for reconstructing full user preferences in some user models, using probe complexity close to the best possible.

More specifically, we showed that user preferences can be fully reconstructed (with high probability) if the user tastes constitute a total order of the objects. The probe complexity of our algorithms (i.e., the number of objects pairs any user is asked to compare) are polylogarithmic, with an exception of algorithm $\mathcal{DPD}$, whose complexity has an extra factor of $O(D^2)$ (assuming that the number of objects is comparable to the number of users).

On the other hand, we showed that if the pairwise preferences are arbitrary, the probe complexity of taste reconstruction is $\Omega(\lceil m^2/n \rceil)$ in the worst case. We leave open the question of the cost of reconstructing tastes whose underlying pairwise preferences represent a general partial order (i.e., assuming only transitivity).

Many other problems remain wide open. For example, it is interesting to investigate the implication of user privacy, which is in contrast to our "billboard" model, where all probe results are readable by all. It is also very interesting to put the algorithms to the test, i.e., to see what are the results in a more realistic setting, rather than the worst-case which we analyzed. An online experiment with real users interactions would be ideal, however this is not always possible. We note that some schemes of using offline data in an unbiased manner were considered [20].

Appendix A. Parallel merge algorithm

Valiant [19] gives a parallel merge algorithm which works as follows. Let v_1 and v_2 be two sorted vectors, when $|v_1| = |v_2| = n$ and let $p = n$ be the number of processors available. The algorithm *marks* the elements of v_1 and v_2 that are indexed by $i \cdot \lceil\sqrt{n}\rceil$ for $i = 1, 2, \ldots$ Sublists between successive marked elements and after the last marked elements are called *segments*. Each marked element in v_1 is then compared with each marked element of v_2. This can be done by $p = n$ processors in $O(1)$ time. Next, after determining in which segment of v_2 each marked element of v_1 needs to be inserted, the algorithm compares each marked element of v_1 with every element of the segment of v_2 found for it. This again can be done by the p processors in constant time. After these steps, we have identified where each of the marked elements of v_1 belongs in v_2. This reduces the problem of merging 2 lists of length n using n processors into merging $\sqrt{n}$ pairs of lists, and each pair can be allocated $\sqrt{n}$ processors, resulting in overall time of $O(\log\log n)$.

In our context, the algorithm is modified as follows. Players play the role of processors, and probes replace comparisons. However, the number of players we have available is αn, which results in an increase of factor $1/\alpha$ in the running time, which in our case corresponds to probe complexity.

References

[1] N. Ailon, Active learning ranking from pairwise preferences with almost optimal query complexity, in: Proc. NIPS, 2011, pp. 810–818.
[2] N. Ailon, M. Charikar, A. Newman, Aggregating inconsistent information: ranking and clustering, J. ACM 55 (5) (2008) 23.
[3] N. Alon, Ranking tournaments, SIAM J. Discrete Math. 20 (1) (2006) 137–142.
[4] N. Alon, B. Awerbuch, Y. Azar, B. Patt-Shamir, Tell me who I am: an interactive recommendation system, Theory Comput. Syst. 45 (2) (2009) 261–279.
[5] B. Awerbuch, Y. Azar, Z. Lotker, B. Patt-Shamir, M.R. Tuttle, Collaborate with strangers to find own preferences, Theory Comput. Syst. 42 (1) (2008) 27–41.
[6] Y. Azar, A. Nisgav, B. Patt-Shamir, Recommender systems with non-binary grades, in: Proc. 23rd SPAA, ACM, 2011, pp. 245–252.
[7] R.M. Bell, Y. Koren, Lessons from the Netflix prize challenge, SIGKDD Explor. 9 (2) (2007) 75–79.
[8] B. Carterette, P.N. Bennett, D.M. Chickering, S.T. Dumais, Here or there, in: Proc. 30th European Conf. on Advances in Information Retrieval, 2008, pp. 16–27.
[9] M. Desarkar, R. Saxena, S. Sarkar, Preference relation based matrix factorization for recommender systems, in: User Modeling, Adaptation, and Personalization, in: LNCS, vol. 7379, Springer, 2012, pp. 63–75.
[10] P. Drineas, I. Kerenidis, P. Raghavan, Competitive recommendation systems, in: Proc. 34th Ann. ACM Symp. on Theory of Computing, ACM, 2002, pp. 82–90.
[11] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Inf. Retr. 4 (2) (2001) 133–151.
[12] N. Jones, A. Brun, A. Boyer, Comparisons instead of ratings: towards more stable preferences, in: Proc. Int. Conf. on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, 2011, pp. 451–456.
[13] C. Kenyon-Mathieu, W. Schudy, How to rank with few errors, in: Proc. 39th Ann. ACM Symp. on Theory of Computing, ACM, 2007, pp. 95–103.
[14] L. Li, W. Chu, J. Langford, R.E. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 661–670.
[15] B. Loepp, T. Hussein, J. Ziegler, Choice-based preference elicitation for collaborative filtering recommender systems, in: Proc. 32nd Ann. ACM Conf. on Human Factors in Computing Systems, 2014, pp. 3085–3094.
[16] B. Loepp, K. Herrmanny, J. Ziegler, Blended recommending: integrating interactive information filtering and algorithmic recommender techniques, in: Proc. of the 33rd Annual ACM Conf. on Human Factors in Computing Systems, 2015, pp. 975–984.
[17] A. Nisgav, B. Patt-Shamir, Improved collaborative filtering, in: Proc. Algorithms and Computation (ISAAC), Springer, 2011, pp. 425–434.
[18] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: Proc. 2nd ACM Conf. on Electronic Commerce, ACM, 2000, pp. 158–167.
[19] L.G. Valiant, Parallelism in comparison problems, SIAM J. Comput. 4 (3) (1975) 348–355.
[20] X. Zhao, W. Zhang, Jun Wang, Interactive collaborative filtering, in: Proc. of the 22nd ACM International Conference on Conference on Information and Knowledge Management, 2013, pp. 1411–1420.
[21] K. Zhou, S. Yang, H. Zha, Functional matrix factorizations for cold-start recommendation, in: Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2011, pp. 315–324.