Adversarial Online Collaborative Filtering

Stephen Pasteris   SPASTERIS@TURING.AC.UK
The Alan Turing Institute, UK

Fabio Vitale   FABIO.VITALE@CENTAI.EU
CENTAI, Italy

Mark Herbster   M.HERBSTER@UCL.AC.UK
University College London, UK

Claudio Gentile   CGENTILE@GOOGLE.COM
Google Research, USA

Andre' Panisson   ANDRE.PANISSON@CENTAI.EU
CENTAI, Italy

Abstract

We investigate the problem of online collaborative filtering under no-repetition constraints, whereby users need to be served content in an online fashion and a given user cannot be recommended the same content item more than once. We start by designing and analyzing an algorithm that works under biclustering assumptions on the user-item preference matrix, and show that this algorithm exhibits an optimal regret guarantee, while being fully adaptive, in that it is oblivious to any prior knowledge about the sequence of users, the universe of items, as well as the biclustering parameters of the preference matrix. We then propose a more robust version of this algorithm which operates with general matrices. Also this algorithm is parameter free, and we prove regret guarantees that scale with the amount by which the preference matrix deviates from a biclustered structure. To our knowledge, these are the first results on online collaborative filtering that hold at this level of generality and adaptivity under no-repetition constraints. Finally, we complement our theoretical findings with simple experiments on real-world datasets aimed at both validating the theory and empirically comparing to standard baselines. This comparison shows the competitive advantage of our approach over these baselines.

Keywords: Online Learning, Collaborative Filtering, No-repetition constraint, Biclustering.

1. Introduction

Helping customers identify their preferences is essential for businesses with a diverse product offering. Many companies rely on recommendation systems (RS) (Resnick and Varian, 1997), which allow users to browse, search, or receive suggestions from online services. Recommendation algorithms let us narrow down massive amounts of information into personalized choices. This is especially relevant in online businesses, where the capabilities of interactive RS have become of paramount importance. Customers can obtain suggestions for movies (e.g., Netflix, YouTube TV), music (e.g., Spotify, YouTube Music), job openings (e.g., LinkedIn, Indeed), or various products (e.g., Amazon, eBay), while their feedback is tracked and exploited to improve future recommendations tailored to specific user interests. In most cases, the goal is to improve user experience as measured by the amount of "likes" given by the user over time.

A standard approach to content recommendation is the one provided by Collaborative Filtering (CF), where personalized recommendations are generated based on both content data and aggregate user activity. CF algorithms are either user-based or item-based. A user-based CF algorithm suggests items that similar users enjoy. Item-based algorithms suggests items that are similar to items enjoyed by the user in the past. In many practical applications, suggesting an item that has already been consumed is often useless (Bresler et al., 2014, 2016; Ariu et al., 2020). For instance, it is pointless to keep advertising the same movie or book to a user after she has already watched or read it. In the recent online CF literature, this assumption is often called the "*no-repetition constraint*" (e.g., Ariu et al. (2020)).

The aim of this paper is to design and analyze novel learning algorithms for online collaborative filtering under the no-repetition constraint assumption, still forcing the algorithms to leverage the collaborative effects in the user-item structure. In this sense, the algorithms we propose combine both user-based and item-based online CF approaches.

Our learning problem can be described as follows. Learning proceeds in a sequence of interactive *trials* (or *rounds*). At each round, a single user shows up, and the RS is compelled to recommend content (an individual item) to them. We make no assumptions whatsoever on the way the sequence of users gets generated across rounds. The user then provides feedback encoding their opinion about the selected item, and the RS uses this signal to update its internal state. The kind of feedback we expect is the binary click/no-click, thumb up/down, like/dislike, which is very common in online media services (TikTok, YouTube, etc.) In any trial, the RS is constrained to recommend to the user at hand an item that *has not* been recommended to that user in the past.

In order to leverage non-trivial collaborative effects, we investigate a latent model of user preferences based on *biclustering* (Hartigan, 1972), or perturbed/noisy versions thereof. Specifically, we shall assume that each user falls under one user *type* (or *cluster*) and that, in the absence of noise, users within the same type prefer the same items. At the same time, items are also clustered so that, in the absence of noise, all items belonging to the same cluster are liked or disliked in the same fashion by each user. Modern user-based (item-based) CF algorithms often operate under these assumptions, which are supported by a number of experiments in the RS literature, for both user (Das et al., 2007; Bellogín and Parapar, 2012; Bresler et al., 2014; Bresler and Karzand, 2021) and item (Sarwar et al., 2001; Bresler et al., 2016; Bresler and Karzand, 2021) clustering. The set of all user preferences is generally viewed as a matrix called a *preference matrix*. A perturbed version of a biclustered matrix is one where the actual preference matrix can be seen as a noisy variant of a biclustered matrix, a scenario we also tackle in our analysis.

Our contributions. We consider a sequential and adversarial learning setting where users and user preferences can be generated *arbitrarily*. We initially investigate the case where the preference matrix is perfectly biclustered (Section 3), and then relax this assumption to allow an adversarial perturbation of the preference matrix (Section 4). In all cases, we quantify online performance in terms of cumulative *regret*, that is, the extent to which the number of recommendation mistakes made by our algorithms across a sequence of rounds exceeds those made by an omniscient oracle that knows the preference matrix beforehand. In the perfect biclustering case, we fully characterize the problem: we both provide a regret lower bound and describe an algorithm that achieves a regret guarantee that matches this lower bound. In the adversarially perturbed case, we introduce a more robust algorithm that operates with general preference matrices, and whose regret performance is expressed in terms of the degree by which the perturbed preference matrix diverges from a perfectly biclustered one. Our algorithms are scalable, *fully adaptive* (that is, *parameter free*), in that they need

not know the parameters of the underlying biclustering structure, the time horizon, or the amount of perturbation in the preference matrix. Moreover, the algorithms can be naturally run in situations where both the set of users and the universe of items increase (arbitrarily) over time.

We then empirically compare (Section 5) versions of our algorithm to three baselines (recommendation based on popularity, the Weighted Regularized Matrix Factorization (WRMF) from Hu and Volinsky (2008), and random recommendations) on a real-world benchmark showing that our algorithms exhibit on this benchmark faster learning curves than its competitors.

Discussion and related work. As far as we are aware, this is the first work that provides theoretical performance guarantees under the no-repetition constraint in a non-stochastic setting where both the sequence of users and the user-item preference matrix are generated adversarially.

Because user preferences can be observed solely for items recommended in the past, we must address the classical problem of how to quickly determine the users' interests without affecting the quality of our recommendations. That is, should we provide recommendations according to the users' interests observed thus far, or should we obtain new feedback signals so as to better profile the users? This well-known exploitation-exploration dilemma has received wide attention in the machine learning and statistics fields. In particular, the research in multi-armed bandit (MAB) problems and methods applied to RS tasks is interested in how to strike an optimal balance between exploitation of current knowledge of user preferences and exploration of new potential interests (see, e.g., the monographs by Bubeck et al. (2012) and Lattimore and Szepesvári (2020)).

One thing to emphasize, though, is that none of the variants of MABs which are readily available in the literature includes all the core elements of the problem we are considering here. An item can only be suggested *once* to a user in our setting, while an arm (item) in MAB problems can typically be recommended multiple times. This is a significant difference between our setting and standard MAB formulations where, once the best arm is discovered for a given user, the problem for that user is deemed to be solved. Clustering (e.g., Bui et al. (2012); Maillard and Mannor (2014); Gentile et al. (2014); Li et al. (2016); Kwon et al. (2017); Jedor et al. (2019)) as well as low-rank (e.g., Katariya et al. (2017); Lu et al. (2018); Hong et al. (2020); Jun et al. (2019); Trinh et al. (2020); Lu et al. (2021); Kang et al. (2022)) assumptions are also widespread in the stochastic MAB literature as applied to recommendation problems, but these works do not consider the no-repetition constraint on the items, nor do they address adversarial perturbations of the preference matrix.

The reader is referred to Appendix A for further discussion on the connections to matrix completion/factorization and structured bandit formulations.

The closest references to our work are perhaps the works by Bresler et al. (2014, 2016); Ariu et al. (2020); Bresler and Karzand (2021), where online CF problems are investigated under the no-repetition constraint assumption. As in our paper, algorithm performance is measured by comparing the proposed algorithm against an omniscient RS that knows the preferences of all users on all items. However, Bresler et al. (2016) assumes the sequence of users is generated in a quite benign way (uniformly at random), while in the papers by Bresler et al. (2014); Ariu et al. (2020); Bresler and Karzand (2021) all users need to receive a recommendation and provide a feedback simultaneously.

Finally, it is worth mentioning that standard ranking problems, where the RS is required to produce a ranked list of diverse items (see, e.g., classical references, like Slivkins et al. (2013)), can be seen as a way to implement the no-repetition constraint, but only within a given user session, not across multiple sessions of the same user. Hence this gives rise a substantially different RS problem than the one we consider here.

2. Preliminaries, Learning Tasks, and Overview of Results

All the tasks considered here involve the recommendation of N items to M users. In order to define our problems we must first introduce what it means for a matrix to be *(bi)clustered*.

Given an $M \times N$ matrix L, we say that two users $i, i' \in [M]$ are *equivalent* if and only if $L_{i,j} = L_{i',j}$ for all $j \in [N]$, that is, if and only if the rows corresponding to the two users are identical. Similarly, two items $j, j' \in [N]$ are *equivalent* if and only if $L_{i,j} = L_{i,j'}$ for all $i \in [M]$. A matrix L is C-user clustered and D-item clustered if the number of equivalence classes under these equivalence relations are no more than C and D, respectively. For brevity, we shall refer to such a matrix as a (C, D)-biclustered matrix.

The Basic Problem. We first describe the simplest version of the problem that we study. We have an unknown binary matrix L which is (C, D)-biclustered for some unknown C and D. We say that user i *likes* item j if and only if $L_{i,j} = 1$. Our learning problem proceeds sequentially in trials (or rounds) $t = 1, 2, \ldots, T$, where on trial t a learning agent (henceforth called "Learner") interacts with its environment as follows:

1. The environment reveals user i_t to Learner;

2. Learner chooses an item j_t to recommend to i_t. However, Learner is restricted in that it cannot have recommended item j_t to user i_t on some earlier trial;

3. L_{i_t,j_t} is revealed to Learner.

Note that the problem restricts the environment in that a given user cannot be queried more than N times. For any trial t, if $L_{i_t,j_t} = 0$ then user i_t does not like item j_t and we say that Learner incurs a *mistake*. The aim of Learner is to minimize the total number of mistakes made throughout the T rounds for the given matrix L and the sequence of users $i_1, \ldots, i_T$ generated by the environment.

In fact, since the binary matrix L can be arbitrary (it may contain a lot of zeros) and the sequence $i_1, \ldots, i_T$ may be generated adversarially, our goal will be to bound the learner's *regret* R, which is defined as the difference between the number of mistakes made by Learner and those which would have been obtained by an *omniscient* oracle that has a-priori knowledge of L. Formally, given a user $i \in [M]$, let ω_i be the number of rounds $t \in [T]$ in which $i = i_t$, and let ξ_i be the number of items in $[N]$ that i likes. Let us denote here and throughout by the brackets $[\![\cdot]\!]$ the indicator function of the predicate at argument. The regret R is then defined as:

$$R := \sum_{t \in [T]} (1 - L_{i_t,j_t}) - \sum_{i \in [M]} (\omega_i - \xi_i)[\![\omega_i \geq \xi_i]\!] \,,$$

where the first summation is the number of mistakes made by Learner and the second summation is the number of mistakes made by the omniscient oracle for the given sequence of users $i_1, \ldots, i_T$. We shall prove for this problem that our randomized algorithm ORCA (**O**ne-time **R**e**C**ommendation **A**lgorithm – Section 3.1) has an expected regret bound of the form:

$$\mathbb{E}[R] = \mathcal{O}\left(\min\{C, D\}(M + N)\right) \,,$$

and a time complexity of only $\mathcal{O}(N)$ per round. We will also prove that the above regret bound is essentially optimal. The algorithm has to be randomized since it is designed to deal with adversarially generated user sequences (the same applies to our second algorithm ORCA*).

General preference matrices. We now turn to the problem of incorporating adversarial perturbation into matrix $\boldsymbol{L}$. In this case we will only exploit similarities among items and leave it as an open problem to adapt our methodology to exploit similarities among users. Our algorithm ORCA* (Section 4.1) takes an integer parameter $\psi \geq 2$. We shall assume now that we have a *hidden D-item clustered* $M \times N$ binary matrix $\boldsymbol{L}^*$ which is perturbed *arbitrarily* to form the matrix $\boldsymbol{L}$. The matrix $\boldsymbol{L}^*$ and parameter ψ induce the following concept of *bad* users and *bad* items. Recall that ω_i and ξ_i are the number of times this user is queried and the number of items that it likes, respectively.

Definition 1 *Given a user $i \in [M]$, its* perturbation level δ_i *is the number of items $j \in [N]$ in which $L_{i,j} \neq L_{i,j}^*$. User i is a* bad *user if and only if both $\delta_i > 0$ and $\omega_i > \xi_i - 2\delta_i$ hold. Given an item $j \in [N]$, its* perturbation level ϵ_j *is the number of users $i \in [M]$ in which $L_{i,j} \neq L_{i,j}^*$. Item j is a* bad *item if and only if $\epsilon_j > \psi$, for the given value of ψ.*

Note that the parameter ψ does affect the definition of a bad item. In particular, when ψ increases, the number of bad items decreases, and it does so in a way that depends on the structure of the (unknown) preference matrix. We can view ψ as a tolerance of the algorithm to item perturbation. A bad user is one which the learner is compelled to serve content "too often". Observe that this notion not only depends on the difference between $\boldsymbol{L}$ and $\boldsymbol{L^*}$, but also on the specific sequence of users $i_1, \ldots, i_T$ generated by the environment. We also stress that in relevant real-world scenarios there will often be no bad users. E.g., there are usually many more books that a person would enjoy than those they have time to read.

Given that we have m bad users and $n = n(\psi)$ bad items, ORCA* enjoys the following regret bound:

$$\mathbb{E}[R] = \mathcal{O}\Big(\min\big\{(D\psi + m + n)(M + N), (D + n + m/\psi)(M + N\psi)\big\}\Big).$$

Note that the two terms in the above minimum are generally incomparable, even when solely viewed as a function of parameter ψ. When $M = \mathcal{O}(N)$ the first term is better due to the reduced influence of n, whilst when $M = \Omega(N\psi)$ the second term is better. Hence the above bound expresses a best-of-both-worlds guarantee which is independent of the relative size of M and N. It is also instructive to consider how the two terms change as a function of ψ. As we said, when ψ increases, n decreases, and vice versa. However, since both terms in the above minimum also exhibit a linear dependence on ψ, there is typically a "sweet spot" for ψ, which can easily be found, again in a fully adaptive way, as explained in Section 4.2.

Dynamic Inventory. A natural extension to our problem is to dynamically allow new users and items over time. Thus on a given trial we may or may not see a (single) new user, but the set of items $\mathcal{I}_t$ that we may recommend from on round t is a superset of the items from previous round, i.e., $\mathcal{I}_{t-1} \subseteq \mathcal{I}_t$. Besides, there is no limit on the number of added items. Conventionally, at the final round T the set of distinct users is $[M]$ and distinct items is $[N] = \mathcal{I}_T$. Our algorithms do not need to know M and N in advance.

For simplicity of presentation, we will only consider here the the perturbation-free case, but the same methodology can also be applied in the adversarially perturbed case. The notion of regret generalizes to this dynamic case as follows.

For all trials $t = 1, \ldots, T$, let $\widehat{\omega}_t$ be defined recursively as $\widehat{\omega}_t = 1 + \sum_{s<t} [\![i_s = i_t]\!][\![\widehat{\omega}_s \leq \widehat{\xi}_s]\!]$, where $\widehat{\xi}_t$ is the number of items in $\mathcal{I}_t$ that user i_t likes. The regret is then defined as

$$R := \sum_{t \in [T]} (1 - L_{i_t, j_t}) - \sum_{t \in [T]} [\![\widehat{\omega}_t > \widehat{\xi}_t]\!].$$

We note that with the above definition $\widehat{\omega}_t$, the regret R is again the difference between the number of mistakes made by Learner and those of an omniscient oracle. For this dynamic case, ORCA enjoys *the same* regret guarantee as it did for the static case. However, its running time increases from $\mathcal{O}(N)$ to $\mathcal{O}(N^2)$ per round. (More precisely, $\mathcal{O}(|\mathcal{I}_T|^2)$ per round, where $|\mathcal{I}|$ is the cardinality of set $\mathcal{I}$.)

3. The Perturbation-Free Case

We start off by describing the basic algorithm ORCA which is designed to work in the absence of perturbation (that is, in the purely biclustered case) when the inventory is either static or dynamic. In the dynamic case the time complexity of the algorithm is $\mathcal{O}(|\mathcal{I}_T|^2) = \mathcal{O}(N^2)$ per trial. On the other hand, in the static case (i.e., when $\mathcal{I}_t = [N]$ for all $t \in [T]$) the time complexity decreases to $\mathcal{O}(N)$ per trial. We shall analyze both the static and dynamic cases together.

Algorithm 1 One time recommendation algorithm (ORCA).

Initialization : $\ell^* \leftarrow 0$; $\ell_i \leftarrow 0$ for all $i \in \mathbb{N}$;

For $t = 1, \ldots, T$:

1. $\ell \leftarrow \ell_{i_t}$;

2. $\mathcal{R} \leftarrow$ set of all items in $\mathcal{I}_t$ not recommended yet to i_t;

3. **If** there exists an item $j_t \in \mathcal{R}$ and a level $k \neq 0$ with $k \leq \ell$, $i_t \in \mathcal{U}_k$ and $j_t \in \mathcal{P}_k$ **then :**

   - Select j_t;   **If** $L_{i_t,j_t} = 0$ **then** remove j_t from the item pool $\mathcal{P}_k$;

4. **Else if** $\ell \neq \ell^*$ **then :**

   - **If** $r_{\ell+1} \in \mathcal{R}$ select item $j_t \leftarrow r_{\ell+1}$ **else** select any item j_t from $\mathcal{R}$;
   - $\ell_{i_t} \leftarrow \ell + 1$;

5. **Else :**

   - Select item j_t uniformly at random from $\mathcal{R}$;
   - **If** $L_{i_t,j_t} = 1$ **then**:   // Create new level
     - $\ell^* \leftarrow \ell^* + 1$; $\ell_{i_t} \leftarrow \ell^*$; $r_{\ell^*} \leftarrow j_t$; $\mathcal{P}_{\ell^*} \leftarrow \mathbb{N}$;   // Define $u_{\ell^*} := i_t$
     - Define $\mathcal{U}_{\ell^*}$ to be the set of all users $i \in [M]$ with:
       ↱ Only for **UC:** $L_{i,r_{\ell'}} = L_{i_t,r_{\ell'}}$ $\forall \ell' \leq \ell^*$
       ↳ Only for **IC:** $L_{i,r_{\ell^*}} = 1$

3.1. The One-Time Recommendation Algorithm ORCA

ORCA's pseudocode is contained in Algorithm 1, and its online functioning is briefly illustrated in Figure 1.

ORCA has two variants – ORCA-UC and ORCA-IC, which exploit user and item clusters, respectively. The two variants share much of the same code. We will refer for brevity to them as UC and IC. These algorithms will be designed in such a way that they can be fused together (into the

resulting ORCA) as follows. We run both UC and IC in parallel and maintain a $0/1$-flag. On any round t, if the flag is set to 0 then we select item j_t with UC and update UC. If the flag is set to 1 we do so with IC. The flag gets flipped if and only if $L_{i_t,j_t} = 0$. The two algorithms run in parallel and do not share any information, except for the items already recommended to each user.

UC and IC also share much of the same analysis. Hence, when we describe and analyze the two algorithms we are considering both at the same time, unless we state otherwise. The pseudo-code of the two algorithms only differ in the lines marked "UC" and "IC" at the very end of Algorithm 1.

The algorithm(s) maintains over time a partitioning of the previously observed users into a sequence of user sets, that we call *levels*. Each observed user is initially assigned to the level 0, and at any trial, can only move forward to the next level or stay in its current one. Informally, the higher is the level of a user, the more we know about her item preferences. Hence, with this method we can profile users based on their observed preferences. The current total number of levels is denoted by ℓ^*, and can only increase over time, when a user belonging to level ℓ^* moves forward to a new level, thereby increasing by 1 the value of ℓ^*. For all levels ℓ', we denote by $u_{\ell'}$ the user i_t on the trial t on which level ℓ' gets created - in that ℓ^* becomes equal to ℓ' (in Line 5 of the pseudo-code). Each level ℓ' is associated with a *representative* item $r_{\ell'}$ and a set of users $\mathcal{U}_{\ell'}$. The only difference between UC and IC is how this set $\mathcal{U}_{\ell'}$ is defined:[1]

- In UC, $\mathcal{U}_{\ell'}$ is the set of all users i in which $L_{i,r_{\ell''}} = L_{u_{\ell'},r_{\ell''}}$ for all $\ell'' \leq \ell'$.

- In IC, $\mathcal{U}_{\ell'}$ is the set of all users i with $L_{i,r_{\ell'}} = 1$.

We note that, for all levels ℓ', the set $\mathcal{U}_{\ell'}$ need not (and cannot) be constructed explicitly. When a user i moves into a level ℓ' – in that ℓ_i becomes equal to ℓ' (during step 4 of the pseudo-code in Algorithm 1) – we check to see if i likes $r_{\ell'}$. This means that at any point in time we know, for all $\ell'' \leq \ell_i$, whether or not $i \in \mathcal{U}_{\ell''}$.

For each level ℓ' we maintain a *pool* $\mathcal{P}_{\ell'}$ which is the set of all items that we believe are liked by all users in $\mathcal{U}_{\ell'}$. In essence, the algorithm is biased towards the initial assumption that all users like all items, and then operates by keeping track of any bias violation to recover the biclustering structure. This is because we initialize $\mathcal{P}_{\ell'}$ to be equal to $\mathbb{N}$ (the universe of potential items). When we encounter a user i that we know to be in $\mathcal{U}_{\ell'}$ and that we know does not like an item j, we then know that j is not liked by all users in $\mathcal{U}_{\ell'}$, and hence we remove it from $\mathcal{P}_{\ell'}$.[2]

On any trial t we call an item $j \in \mathcal{I}_t$ *recommendable* if it has not been recommended yet to user i_t and there exists a level $\ell' \neq 0$ not larger than[3] the current level of i_t, such that $i_t \in \mathcal{U}_{\ell'}$ and $j_t \in \mathcal{P}_{\ell'}$.

On any trial t in which $\ell := \ell_{i_t} > 0$, we recommend our item j_t and update the level of i_t as follows. If there exists a recommendable item (i.e. the condition in Line 3 is true) then we choose j_t to be such an item. If no such item exists (i.e. the condition in Line 3 is false) then, given $\ell < \ell^*$ (so that the condition in Line 4 is true), we move i_t up a level. We do so by first checking to see if i_t likes the next level's representative item $r_{\ell+1}$ (by choosing $j_t := r_{\ell+1}$ if it hasn't already been recommended this item) and then updating $\ell_{i_t} \leftarrow \ell + 1$. Now suppose we want to move i_t up a level but $\ell_{i_t} = \ell^*$ so there is no level to move up to (that is, the condition in Line 5 is true). In this case,

---

1. Note that at initialization, we set $\ell_i \leftarrow 0$ for all i, yet, level 0 is not a real level, and the sets $\mathcal{U}_0$ and $\mathcal{P}_0$ are dummy sets that are only meant to simplify the pseudo-code.

2. To keep the algorithm simple and fast we actually do not always remove this item, but for the purposes of this discussion assume we do.

3. In the static inventory model the condition on ℓ' in this definition is that it is *equal* to current level of i_t.
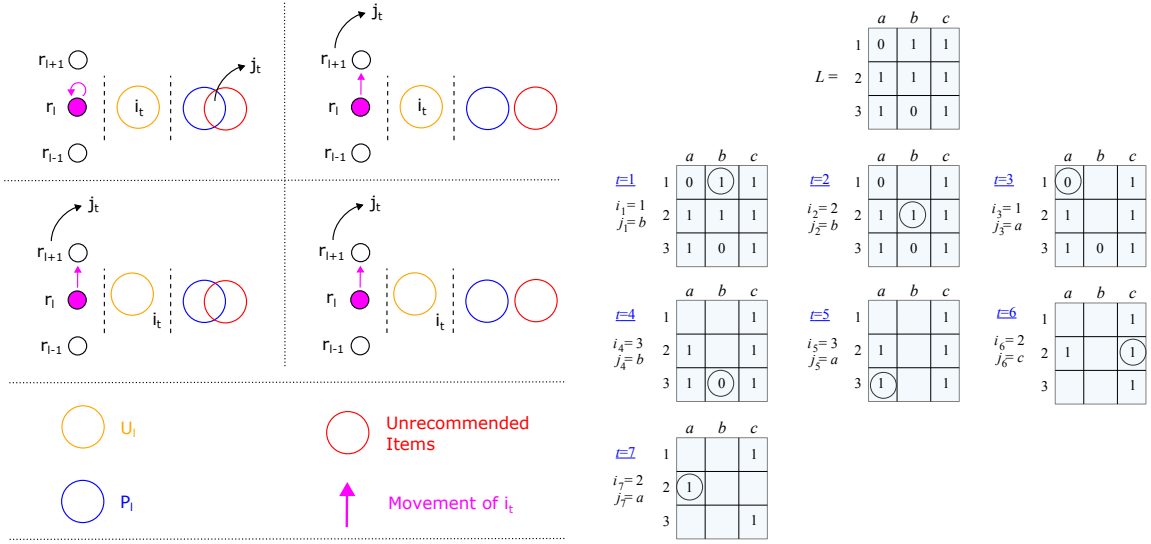
Figure 1: **Left:** The behavior of ORCA (with a static inventory) at trial t when $\ell := \ell_{i_t} < \ell^*$. The four boxed diagrams on top represent the possible cases. In all cases, the yellow circle represents the set $\mathcal{U}_\ell$, the red circle contains the set of so far unrecommended items for user i_t, and the blue circle represents set $\mathcal{P}_\ell$. The figure illustrates the movement of i_t across levels. In the upper-left diagram the condition in Line 3 is true. In all other diagrams the condition in Line 4 is true. Note that in the upper-left diagram j_t is removed from the blue set $\mathcal{P}_\ell$. The item j_t is always removed from the red set of items not yet recommended to user i_t. When $\ell = \ell^*$ the upper-left diagram still applies. **Right:** An example run of ORCA-UC with user set $\{1, 2, 3\}$, item set $\{a, b, c\}$, and a static inventory.

we draw j_t uniformly at random from those items in $\mathcal{I}_t$ not yet recommended to i_t. If i_t likes j_t then we increment ℓ^* by one and then create a new level ℓ^* with $r_{\ell^*} := j_t$ and $u_{\ell^*} := i_t$.

Example run. For further clarity, in Figure 1 (right) we give an example run of ORCA-UC with user set $\{1, 2, 3\}$, item set $\{a, b, c\}$ and a static inventory. We now detail what happens on each trial:

- $t = 1, i_1 = 1$. Since $\ell_1 = 0$ and $\ell^* = 0$ the algorithm attempts to create a new level (level 1) by sampling j_1 uniformly at random from $\mathcal{R} = \{a, b, c\}$. We draw b so set $r_1 = b$. Since $L_{1,b} = 1$ user 1 moves into level 1 so now $\ell_1 = 1$ and $\ell^* = 1$. We initialise $\mathcal{P}_1 = \{a, b, c\}$. We note that if it had been the case that $L_{1,b} = 0$ then user 1 would have stayed at level 0 and no new level would have been created.

- $t = 2, i_2 = 2$. Since $\ell_2 = 0$ and $\ell^* = 1$, user 2 is move into level 1, and hence $j_2 = r_1 = b$, and now $\ell_2 = 1$.

- $t = 3, i_3 = 1$. We have $\ell_1 = 1$. Also $\mathcal{R} = \{a, c\}$ so $\mathcal{R} \cap \mathcal{P}_1 \neq \emptyset$ and hence the algorithm chooses any j_3 from $\mathcal{R} \cap \mathcal{P}_1 = \{a, c\}$. Say, we choose $j_3 = a$. Since $L_{1,a} = 0$ we remove a from $\mathcal{P}_1$.

- $t = 4, i_4 = 3$. Since $\ell_3 = 0$ and $\ell^* = 1$, user 3 is move into level 1, hence $j_4 = r_1 = b$, and now $\ell_3 = 1$.

- $t = 5, i_5 = 3$. Since $L_{3,r_1} = 0$ we have that $3 \notin \mathcal{U}_1$ and hence, since $\ell_3 = 1$ and $\ell^* = 1$, the algorithm attempts to create a new level by sampling j_5 uniformly at random from $\mathcal{R} = \{a, c\}$. Say we draw $j_5 = a$. Since $L_{3,a} = 1$, a new level is created. Now $\ell^* = \ell_3 = 2$ and $r_2 = j_5 = a$. We also initialise $\mathcal{P}_2 = \{a, b, c\}$. We note that if it had been the case that $L_{3,a} = 0$ then user 3 would have stayed at level 1 and no new level would have been created.
- $t = 6, i_6 = 2$. We have $\ell_2 = 1$, $\mathcal{R} = \{a, c\}$ and $\mathcal{P}_1 = \{b, c\}$. So $\mathcal{R} \cap \mathcal{P}_1 = \{c\}$ and hence $j_6 = c$.
- $t = 7, i_7 = 2$. We have $\ell_2 = 1$, $\mathcal{R} = \{a\}$ and $\mathcal{P}_1 = \{b, c\}$. So $\mathcal{R} \cap \mathcal{P}_1 = \emptyset$ and hence, since $\ell^* > \ell_2$, user 2 is moved up to level 2, so that $\ell_2 = 2$. Since $r_2 = a$ and hence $r_2 \in \mathcal{R}$ we choose $j_7 = r_2 = a$. We note that if it was the case that $r_2 \notin \mathcal{R}$ then j_t would have been chosen from $\mathcal{R}$.

3.2. Analysis

We analyze the properties of ORCA by giving an upper bound on its regret as well as a matching lower bound on the regret of *any* algorithm (hence showing the optimality of ORCA).

Theorem 2 *Let* ORCA *be run on a (C, D)-biclustered matrix $\boldsymbol{L}$ of size $M \times N$ with an arbitrary sequence of users $i_1, \ldots, i_T$ and an arbitrary monotonically increasing sequence of item sets $\mathcal{I}_1, \ldots, \mathcal{I}_T \subseteq [N]$. Then the expected regret of* ORCA *is upper bounded as*

$$\mathbb{E}[R] = \mathcal{O}(\min\{C, D\}(M + N)),$$

the expectation being over the internal randomization of the algorithm. ORCA *is parameter-free in that C, D, M, N need not be known.*

Proof Here we sketch the proof, deferring the full proof to Appendix B.1. Without loss of generality we assume that on every trial t there exists an item $j \in \mathcal{I}_t$ which i_t likes and has not been recommended to i_t before so that the regret equals the number of mistakes. Let Λ be the total number of levels created by Algorithm 1.

Given a level $\ell' \in [\Lambda]$ we count the following mistakes. There are at most N mistakes made on trials corresponding to Line 3 of Algorithm 1 with $k = \ell'$, at most M mistakes on trials corresponding to Line 4 with $\ell = \ell' - 1$ and, in expectation, at most N mistakes made on trials corresponding to Line 5 with $\ell = \ell' - 1$. This means that the total number of mistakes is $\mathcal{O}(\Lambda(M + N))$.

Next, we proceed to bounding Λ. A crucial property here is what we call the *separation property*, stating that given $\ell', \ell'' \in [\Lambda]$ with $\ell'' > \ell'$ and $u_{\ell''} \in \mathcal{U}_{\ell'}$ there exists some $i \in \mathcal{U}_{\ell'}$ with $L_{i,r_{\ell''}} = 0$. With this property in hand, we then analyze the algorithms UC and IC separately.

In IC the separation property implies that given distinct levels $\ell', \ell'' \in [\Lambda]$, their representative items $r_{\ell'}$ and $r_{\ell''}$ are not equivalent. This directly implies that $\Lambda \leq D$.

In UC the separation property leads to what we call the *tree property*, stating that given levels $\ell', \ell'' \in [\Lambda]$ with $\ell' < \ell''$ we have either $\mathcal{U}_{\ell''} \cap \mathcal{U}_{\ell'} = \emptyset$ or $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$. We build a forest (whose nodes are sets) as follows. First, for every level $\ell'' \in [\Lambda]$ we make $\mathcal{U}_{\ell''}$ a node. If there exists $\ell' < \ell''$ with $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$ then for the maximum such ℓ' we make $\mathcal{U}_{\ell'}$ the parent of $\mathcal{U}_{\ell''}$. Otherwise $\mathcal{U}_{\ell''}$ has no parent. Finally, if there exists $\ell' \in [\Lambda]$ such that $\mathcal{U}_{\ell'}$ has a single child $\mathcal{U}_{\ell''}$ then we add the set $\mathcal{U}_{\ell'} \setminus \mathcal{U}_{\ell''}$ to the forest and make it a child of $\mathcal{U}_{\ell'}$. Figure 2 gives an example of this forest. By the tree property all leaves are disjoint and non-empty, which implies that there are at most C leaves. Since each internal node of the forest has at least two children, and there are at least Λ nodes, this yields $\Lambda \leq 2C$.

We have hence shown that the mistakes of IC and UC are $\mathcal{O}(D(M + N))$ and $\mathcal{O}(C(M + N))$, respectively. When combined together into ORCA we hence get the desired bound. ∎
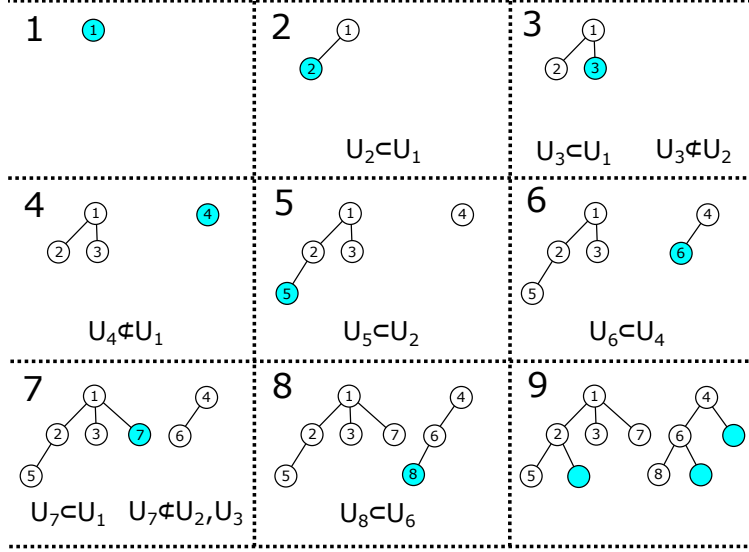

Figure 2: An example of the construction of the forest structure of the sets in ORCA-UC with $\Lambda = 8$. The set corresponding to a node is a strict subset of that corresponding to its parent (if it exists) whilst the sets corresponding to nodes on different root-to-leaf paths are disjoint. For all $\ell' \in [\Lambda]$ the node numbered ℓ' corresponds to the set $\mathcal{U}_{\ell'}$ and the ℓ'-th diagram depicts its construction (the blue node). The blue nodes in the 9-th diagram correspond to the sets $\mathcal{U}_2 \setminus \mathcal{U}_5$, $\mathcal{U}_6 \setminus \mathcal{U}_8$ and $\mathcal{U}_4 \setminus \mathcal{U}_6$ which are all non-empty.

As for the lower bound, we have the following result that proves the optimality of ORCA in the static inventory model. As the static model is a special case of the dynamic one, this also proves optimality in the dynamic inventory model.

Theorem 3 *For any algorithm and any $M, N, C, D \in \mathbb{N}$ with $\min(C, D) \leq \sqrt{\min(M, N)}$ there exists a (C, D)-biclustered matrix $\boldsymbol{L}$ of size $M \times N$, a time-horizon $T \in \mathbb{N}$, and a user sequence $i_1, \ldots, i_T$ such that the algorithm has, in the static inventory model, a regret of*

$$\Omega(\min\{C, D\}(M + N)) .$$

4. The Adversarial Perturbation Case

We now turn to the problem of incorporating arbitrary perturbations of the biclustered matrix. For simplicity we will only consider the static inventory model, but note that the dynamic-inventory methodology from the perturbation-free case can be applied to this case also. Our algorithm ORCA* only exploits similarities among items – we leave it as an open problem to adapt our methodology in order to exploit similarities among users.

We give two algorithms, ORCA*-UIE (ORCA* with User Item Exclusion) and ORCA*-UE (ORCA* with User Exclusion) which have expected regret bounds of

$$\mathbb{E}[R] = \mathcal{O}((D\psi + m + n)(M + N)) \quad \text{and} \quad \mathbb{E}[R] = \mathcal{O}((D + n + m/\psi)(M + N\psi)),$$

respectively. These algorithms can be fused together (into the algorithm ORCA*) in the same way as we did for ORCA-UC and ORCA-IC, in order to obtain a best-of-both guarantee. These two algorithms differ only by whether the instruction labelled "UIE" in the pseudo-code of Algorithm 2 (check Step 7 therein) is included or not.

Algorithm 2 One time recommendation algorithm for adversarial perturbation (ORCA*).

Input : $\psi \in \mathbb{N} \setminus \{1\}$ `// The dependence on $\psi$ can be avoided - see Section 4.2`
Initialization : $\ell^* \leftarrow 0; \quad \ell_i \leftarrow 0$ for all $i \in [M]; \quad \mathcal{E}, \mathcal{F} \leftarrow \emptyset;$
For $t = 1, \ldots, T :$

1. $\ell \leftarrow \ell_{i_t};$

2. $\mathcal{R} \leftarrow$ set of all items not recommended yet to $i_t;$

3. **If** $\mathcal{R} \cap \mathcal{F} \neq \emptyset$ **then** select any j_t from $\mathcal{R} \cap \mathcal{F};$

4. **Else if** $i_t \in \mathcal{E}$ **then** select any j_t from $\mathcal{R};$

5. **Else If** $\ell > 0$ and $L_{i_t, r_\ell} = 1$ and $\mathcal{R} \cap \mathcal{P}_\ell \neq \emptyset$ **then :**

    - Select any j_t from $\mathcal{R} \cap \mathcal{P}_\ell;$
    - **If** $L_{i_t, j_t} = 0$ **then :**
        - $c_{\ell, j_t} \leftarrow c_{\ell, j_t} + 1; \quad$ **If** $c_{\ell, j_t} > 2\psi$ **then** remove j_t from $\mathcal{P}_\ell;$

6. **Else if** $\ell \neq \ell^*$ **then :**

    - **If** $r_{\ell+1} \in \mathcal{R}$ **then** select $j_t \leftarrow r_{\ell+1},$ **Else** select any j_t from $\mathcal{R};$
    - $\ell_{i_t} \leftarrow \ell + 1;$

7. **Else :**
    - Select j_t uniformly at random from $\mathcal{R}; \quad$ Select $\gamma \sim \text{Bernoulli}(1/\psi);$
    - **If** $L_{i_t, j_t} = 1$ and $\gamma = 0$ **then :**
        - Add i_t to $\mathcal{E}; \quad$ Only for **UIE:** Add j_t to $\mathcal{F};$
    - **If** $L_{i_t, j_t} = 1$ and $\gamma = 1$ **then :**
        $\ell^* \leftarrow \ell^* + 1; \quad \ell_{i_t} \leftarrow \ell^*; \quad r_{\ell^*} \leftarrow j_t; \quad \mathcal{P}_{\ell^*} \leftarrow [N]; \quad c_{\ell^*, j} \leftarrow 0 \ \ \forall j \in [N];$

4.1. The One-Time Recommendation Algorithm ORCA*

ORCA* is a modification of ORCA-IC from Algorithm 1. Lines 5, 6, and 7 of the pseudo-code of ORCA* (Algorithm 2) correspond to Lines 3, 4, and 5 of the pseudo-code of ORCA. To arrive at ORCA* we make the following changes to ORCA-IC:

- Although this isn't actually a fundamental change, since ORCA* is in the static inventory model and we are only doing item clustering, the definition of a recommendable item simplifies: an item j is recommendable at trial t if and only if the current level ℓ of user i_t is greater than zero, j belongs to $\mathcal{P}_\ell$ and is not yet recommended to i_t, and $L_{i_t,r_\ell} = 1$. The concept of a recommendable item features in Step 5.

- In ORCA-IC we removed an item j from a set $\mathcal{P}_{\ell'}$ as soon as we determined that there exists some user $i \in \mathcal{U}_{\ell'}$ with $L_{i,j} = 0$. In ORCA* we only remove j from the set $\mathcal{P}_{\ell'}$ (in Step 5 of the pseudo-code in Algorithm 2) when we determine that there exist over 2ψ such users. The variable $c_{\ell',j}$ keeps track of how many of these users have been found so far. Note that the set $\mathcal{U}_{\ell'}$ is not explicitly defined in the pseudo-code of ORCA*.

- The next modification is rather counter-intuitive. In ORCA-IC (under the static inventory model) if i_t is at level ℓ^* and there are no items in $\mathcal{P}_{\ell*}$ that are not yet recommended to i_t, then we draw j_t uniformly at random from the unrecommended items, and if $L_{i_t,j_t} = 1$ we create a new level. In UIE however, we only create a new level (in Step 7) with probability $1/\psi$. Otherwise we *exclude* i_t and j_t. When a user i is excluded, future items are recommended to it arbitrarily (in Step 4) and trials t in which $i_t = i$ no longer have an effect on the rest of the algorithm. When an item j is excluded it will be recommended to all users as soon as possible (in Step 3). Excluded users and items are recorded in the sets $\mathcal{E}$ and $\mathcal{F}$, respectively. UE differs slightly in that only users are excluded, not items.

4.2. Analysis

Theorem 4 *Let ORCA* be run with parameter $\psi \geq 2$ on an $M \times N$ matrix $\boldsymbol{L}$ and an arbitrary sequence of users $i_1, ..., i_T$. Suppose that there exists a (C, D)-biclustered ground-truth matrix $\boldsymbol{L}^*$ that induces m bad users and $n = n(\psi)$ bad items (recall Definition 1) on $\boldsymbol{L}$. Then the expected regret of ORCA* is upper bounded as*

$$\mathbb{E}[R] = \mathcal{O}\Big(\min\Big\{(D\psi + m + n)(M + N),\ (D + n + m/\psi)(M + N\psi)\Big\}\Big),$$

the expectation being over the internal randomization of the algorithm. Note that C and D need not be known.

In Appendix B.4 we show that a standard doubling trick can remove the parameter ψ in ORCA*, allowing us to achieve a regret bound that is only an $\mathcal{O}(\ln(M))$ factor off the regret bound of ORCA* (Theorem 4) with ψ therein replaced by the optimal ψ in hindsight.

Proof Here we sketch the proof, deferring the full proof to Appendix B.2. Let us first consider UIE. As in the analysis of ORCA, we assume without loss of generality that for all trials t there exists an item that i_t likes and that has not been recommended to i_t so far. Following similar steps as in the analysis of ORCA, we first bound the contribution of each level to the regret, which is now $\mathcal{O}(M + \psi N)$.

Let $\mathcal{H}$ be the set of trials t in which Line 7 of Algorithm 2 is executed and $L_{i_t,j_t} = 1$. Let $\mathcal{H}^*$ be the set of trials $t \in \mathcal{H}$ in which $\gamma = 1$ on trial t. Note that on each trial in $\mathcal{H}^*$ a level is created and on each trial t in $\mathcal{H} \setminus \mathcal{H}^*$ user i_t and item j_t are excluded, which contributes $\mathcal{O}(M + N)$ to the total regret.

Trials $t \in \mathcal{H}^*$ in which $L^*_{i_t,j_t} = 0$ can hurt the algorithm by creating an unnecessary level. Given a non-bad user i, trials $t \in \mathcal{H}$ with $i_t = i$ and $L^*_{i_t,j_t} = 0$ happen infrequently enough that we can effectively ignore their contribution to the regret. However, given a bad user i, trials $t \in \mathcal{H}$ in which $L^*_{i_t,j_t} = 0$ can be more likely to happen. Yet, on such a trial, with probability $1 - 1/\psi$ user i_t becomes excluded which implies that bad users contribute, in expectation, only $\mathcal{O}(M + N)$ to the regret. Hence we need not be concerned with trials $t \in \mathcal{H}$ with $L^*_{i_t,j_t} = 0$. In a similar fashion, one can show that bad items contribute $\mathcal{O}(M + N)$ to the overall regret.

Let a *cluster* $\mathcal{K}$ be a set of items that are not bad and are all equivalent in the matrix $\boldsymbol{L}^*$. We will be interested in the contribution to the regret of trials $t \in \mathcal{K}' := \{t \in \mathcal{H} \mid j_t \in \mathcal{K}\}$ with $L^*_{i_t,j_t} = 1$. Define $\tau := \min \mathcal{K}' \cap \mathcal{H}^*$. Note that there are, in expectation, at most ψ trials in $\{t \in \mathcal{K}' \mid t < \tau\}$, and all such trials are in $\mathcal{H} \setminus \mathcal{H}^*$. Due to the fact that a level is created on trial τ (if τ is finite), and that its representative item j_τ is not bad, we can prove that for all trials $t \in \mathcal{K}'$ with $t > \tau$ we have $L_{i_t,j_\tau} = 0$. Since j_τ is not bad and j_τ and j_t are equivalent in $\boldsymbol{L}^*$, there can be at most ψ such t in which $L^*_{i_t,j_t} = 1$. This implies that in expectation at most one of them will be in $\mathcal{H}^*$. Summing up, this allows us to conclude that trials $t \in \mathcal{K}'$ with $L^*_{i_t,j_t} = 1$ have an expected contribution to ORCA*'s regret of $\mathcal{O}(\psi(M + N))$.

Taking a sum over all clusters, bad users and bad items gives us the required regret bound. In UE we can, without loss of generality, assume that there are no bad items. Noting the fact that each trial in $\mathcal{H} \setminus \mathcal{H}^*$ now only contributes $\mathcal{O}(N)$, the same analysis as UIE gives us the desired regret guarantee. ∎

5. Experiments

We now report the results of a preliminary set of experiments comparing (versions of) ORCA to common CF baselines.

Datasets. The MovieLens dataset, produced by the GroupLens research team (https://grouplens.org/datasets/movielens) (Harper and Konstan, 2015), is commonly used in recommender system studies. It consists of 1M ratings in the range [1, 5], and we follow the convention used in previous research (e.g., Lim et al. (2015)), where ratings above 3 are considered as positive feedback.

To test the algorithms under different conditions, we selected subsets of Movielens with increasing number of items. We selected N items uniformly at random, with $N \in \{50, 100, 200\}$. Then we retained all the "likes" where the item is in N. The resulting average number of users M and likes are 3376 ± 535 and 7765 ± 1870 for $N = 50$ items, 4488 ± 301 and 15886 ± 2375 for $N = 100$, 5290 ± 191 and 32123 ± 4117 for $N = 200$. In each experiment, to be sure we go through all likes, we ran $M \times N$ rounds of recommendations.

Algorithms and baselines. We use as baselines two naive methods and one based on Matrix Factorization (MF). The first method, referred to as RANDOM, just recommends an item at random (among the available ones for the current user). The second method, POP ("popularity"), leverages the fact that, in MovieLens, item popularity is highly predictive of a successful recommendation, and we recommend the most popular among the items that have been recommended in the past. Finally, as an MF approach, we used a well-known Matrix Factorization algorithm, specifically the Weighted Regularized Matrix Factorization (WRMF) proposed by Hu and Volinsky (2008); Rong Pan (2008). Since it was originally designed for batch recommendation, we adapted WRMF to an online setting as follows. Initially, random recommendations are generated for the first 1,000 rounds, as WRMF requires a minimum amount of "likes" to perform MF. Subsequently, we fit the WRMF
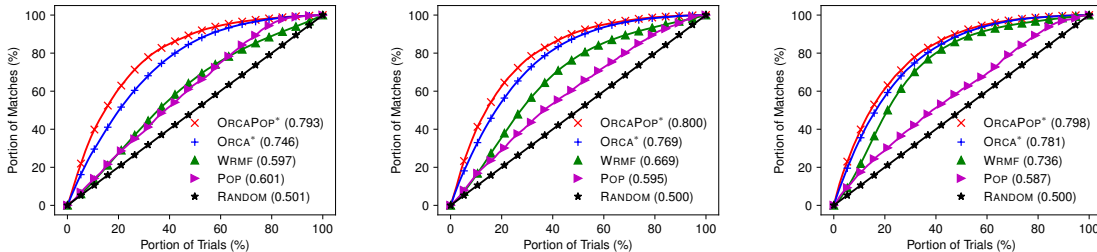
Figure 3: Recommendation curves for subsets of MovieLens with varying number of items (from left to right: $N = 50$, $N = 100$, $N = 200$ items). The curves are averages across 30 repetitions. The numbers in braces give the area under each curve (the higher the better). The displayed curves for WRMF are the best performers across the number of latent factors.

model to produce recommendations for the next batch of 1,000 rounds. As the number of rounds increases, the model needs to be trained less frequently, as more data is needed to obtain a significant performance increase. Hence, to make this approach feasible for millions of rounds, the batch size is incrementally increased by 10% at each re-train phase. This adjustment causes only a negligible difference in performance compared to full batch training at each round. We relied on the MREC recommender system library (https://mendeley.github.io/mrec/), using default values for confidence weight (1) and regularization constant (0.015), and 15 iterations of alternating least squares, since performance was less sensible to these parameters. Instead, the number of latent factors turned out to be very important for actual performance. We report the results for WRMF with 4, 8, 16, and 32 latent factors.

We compared these baselines to[4] ORCA*. Moreover, in order to show the versatility of ORCA*, we also leverage the fact that popular items have higher probability of success, and in steps 3, 4, 5, 6 we replace *select any j_t from $\square$* with *select the most popular j_t from $\square$*. This modification does not affect the validity of our theoretical results. We call the resulting algorithm ORCAPOP*. In all experiments, in order to avoid unfair comparisons with the baselines, the user sequence $i_1, \ldots, i_T$ is always generated uniformly at random over all available users.

Results. All our experiments have been run on an Intel Xeon Gold 6312U - 24c/48t - 2.4 GHz/3.6 GHz with 256 GB ECC 3200 MHz memory. Our results are contained in Figure 3 and Tables 1 and 2 (Appendix C). Figure 3 plots, for each of the three values of N, the fraction of uncovered user-item matches (i.e., the "likes" in the preference matrix $\boldsymbol{L}$) against the fraction of recommendations so far. The total number of recommendation (100% in the x-axis) always corresponds to $M \times N$, that is, the total number of entries in $\boldsymbol{L}$. These curves quantify the pace at which the matches are uncovered by the different algorithms, hence the higher the better. A more compact version of this metric is the area under these curves, reported in both Figure 3 and Table 1.

Though a bit preliminary in nature, some trends emerge: The performance of ORCA* and ORCAPOP* is consistently high even at small number of items, where MF algorithms suffer more due to the cold start problem. At higher number of items, when cold start is less evident, WRMF and ORCA* have closer performances. The POP heuristic suffers from the subsampling of the items we made during preprocessing.

---

4. We decided not run ORCA here, as the preference matrix is not (C, D)-biclustered (for some small C and D).

6. Conclusions and Limitations

We have considered a sequential content recommendation problem where items can only be recommended once to each user ("no-repetition constraint"). Unlike the abundant literature on MAB under low-rank and clustering assumptions, we have handled through a regret analysis the more general situation where users show up in the system in an arbitrary (possibly adversarial) order. We have proposed ORCA, that works under biclustering assumptions, and have shown that this algorithm exhibits an optimal (up to constant factor) regret guarantee against an omniscient oracle that knows the user-item preference matrix ahead of time. We have then extended ORCA to ORCA*, a more robust version which is able to handle arbitrary preference matrices. Finally, we have provided preliminary empirical evidence of the effectiveness of (versions of) our algorithm as compared to standard baselines.

Currently, ORCA* is not able to leverage similarities among users. This is likely to require a substantial redesign of our algorithm. Among the relevant extensions that would allow us to better address real-world scenarios are: i. The case where the learner has access to side information (i.e., features) about users and/or items, ii. The case where the user feedback is non-binary (e.g., relevance scores rather than clicks), and iii. Extending the biclustering assumption to the slightly more general low-rank assumption, ubiquitous in the CF literature.

References

ST Aditya, Onkar Dabeer, and Bikash Kumar Dey. A channel coding perspective of collaborative filtering. *IEEE Transactions on Information Theory*, 57(4):2327–2341, 2011.

Kaito Ariu, Narae Ryu, Se-Young Yun, and Alexandre Proutière. Regret in online recommendation systems. *Advances in Neural Information Processing Systems*, 33:21141–21150, 2020.

Alejandro Bellogín and Javier Parapar. Using graph partitioning techniques for neighbour selection in user-based collaborative filtering. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 213–216, 2012.

Gérard Biau, Benoît Cadre, and Laurent Rouviere. Statistical analysis of k-nearest neighbor collaborative recommendation. *The Annals of Statistics*, 38(3):1568–1592, 2010.

Guy Bresler and Mina Karzand. Regret bounds and regimes of optimality for user-user and item-item collaborative filtering. *IEEE Transactions on Information Theory*, 67(6):4197–4222, 2021.

Guy Bresler, George H Chen, and Devavrat Shah. A latent source model for online collaborative filtering. *Advances in neural information processing systems*, 27, 2014.

Guy Bresler, Devavrat Shah, and Luis Filipe Voloch. Collaborative filtering with low regret. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pages 207–220, 2016.

Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

Loc Bui, Ramesh Johari, and Shie Mannor. Clustered bandits. *arXiv preprint arXiv:1206.4169*, 2012.

Emmanuel Candes and Benjamin Recht. Exact matrix completion via convex optimization. *Communications of the ACM*, 55(6):111–119, 2012.

Onkar Dabeer. Adaptive collaborating filtering: The low noise regime. In *2013 IEEE International Symposium on Information Theory*, pages 1197–1201. IEEE, 2013.

Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280, 2007.

Claudio Gentile, Shuai Li, and Giovanni Zappella. Online clustering of bandits. In *International Conference on Machine Learning*, pages 757–765. PMLR, 2014.

F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

J. A. Hartigan. Direct Clustering of a Data Matrix. *Journal of the American Statistical Association*, 67(337):123–129, 1972. ISSN 01621459. doi: 10.2307/2284710. URL http://dx.doi.org/10.2307/2284710.

Elad Hazan, Satyen Kale, and Shai Shalev-Shwartz. Near-optimal algorithms for online matrix prediction. In *Conference on Learning Theory*, pages 38–1. JMLR Workshop and Conference Proceedings, 2012.

Mark Herbster, Stephen Pasteris, and Lisa Tse. Online matrix completion with side information. *Advances in Neural Information Processing Systems*, 33:20402–20414, 2020.

Joey Hong, Branislav Kveton, Manzil Zaheer, Yinlam Chow, Amr Ahmed, and Craig Boutilier. Latent bandits revisited. In *Advances in Neural Information Processing Systems*, volume 33, pages 13423–13433. Curran Associates, Inc., 2020.

Koren Yehuda Hu, Yifan and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *IEEE International Conference on Data Mining (ICDM 2008)*, pages 263–272, 2008.

Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674, 2013.

Matthieu Jedor, Vianney Perchet, and Jonathan Louedec. Categorized bandits. *Advances in Neural Information Processing Systems*, 32, 2019.

Kwang-Sung Jun, Rebecca Willett, Stephen Wright, and Robert Nowak. Bilinear bandits with low-rank structure. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3163–3172. PMLR, 2019.

Y. Kang, C.J. Hsieh, and T. Chun Man Lee. Efficient frameworks for generalized low-rank matrix bandit problems. In *Advances in Neural Information Processing Systems*. PMLR, 2022.

Sumeet Katariya, Branislav Kveton, Csaba Szepesvári, Claire Vernade, and Zheng Wen. Bernoulli rank-1 bandits for click feedback. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from a few entries. *IEEE transactions on information theory*, 56(6):2980–2998, 2010.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Joon Kwon, Vianney Perchet, and Claire Vernade. Sparse stochastic bandits. *arXiv preprint arXiv:1706.01383*, 2017.

Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

Shuai Li, Alexandros Karatzoglou, and Claudio Gentile. Collaborative filtering bandits. In *Association for Computing Machinery*, SIGIR '16, page 539–548, New York, NY, USA, 2016.

Daryl Lim, Julian McAuley, and Gert Lanckriet. Top-n recommendation with missing implicit feedback. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 309–312, 2015.

Xiuyuan Lu, Zheng Wen, and Branislav Kveton. Efficient online recommendation via low-rank ensemble sampling. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 460–464. Association for Computing Machinery, 2018. ISBN 9781450359016.

Yangyi Lu, Amirhossein Meisami, and Ambuj Tewari. Low-rank generalized linear bandit problems. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 460–468. PMLR, 2021.

Odalric-Ambrym Maillard and Shie Mannor. Latent bandits. In *International Conference on Machine Learning*, pages 136–144. PMLR, 2014.

Sahand Negahban and Martin J Wainwright. Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *The Journal of Machine Learning Research*, 13(1): 1665–1697, 2012.

Soumyabrata Pal, Arun Sai Suggala, Karthikeyan Shanmugam, and Prateek Jain. Optimal algorithms for latent bandits with cluster structure. *arXiv preprint arXiv:2301.07040*, 2023.

Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

Angelika Rohde and Alexandre B Tsybakov. Estimation of high-dimensional low-rank matrices. *The Annals of Statistics*, 39(2):887–930, 2011.

Bin Cao Nathan N. Liu Rajan Lukose Martin Scholz Qiang Yang Rong Pan, Yunhong Zhou. One-class collaborative filtering. In *Eighth IEEE International Conference on Data Mining*, 2008.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Ranked bandits in metric spaces: Learning diverse rankings over large document collections. *J. Mach. Learn. Res.*, 14(1):399–436, 2013.

Cindy Trinh, Emilie Kaufmann, Claire Vernade, and Richard Combes. Solving bernoulli rank-one bandits with unimodal thompson sampling. In *Proceedings of the 31st International Conference on Algorithmic Learning Theory*, volume 117 of *Proceedings of Machine Learning Research*, pages 862–889. PMLR, 2020.

Koen Verstrepen and Bart Goethals. Unifying nearest neighbors collaborative filtering. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 177–184, 2014.

Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, 2006.

Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1411–1420, 2013.

Appendix A. Further Related Work

Our problem also shares similarities with the classical problem of *Matrix Completion* (MC) see, e.g., the papers by Keshavan et al. (2010); Biau et al. (2010); Rohde and Tsybakov (2011); Aditya et al. (2011); Candes and Recht (2012); Negahban and Wainwright (2012); Jain et al. (2013). The objective in a MC problem is to estimate the remaining entries of a given matrix having at one's disposal a subset of the observed entries. It is often assumed that the matrix meets specific criteria (like low rank). Again, a closer inspection reveals that the typical conditions under which a MC algorithm work do not properly reflect the sequence of user events in a RS. For instance, whereas MC algorithms typically require the subset of the entries to be drawn at random according to some distribution, we are not bound to observe the matrix entries according to such a benign criterion, for users may visit a RS and revisit in an arbitrary order. Furthermore, we impose differing structural assumptions, such as the adversarial perturbation of the preference matrix. The closest MC problem to ours is when the components need to be predicted online (Hazan et al., 2012; Herbster et al., 2020). However, this problem is fundamentally different from ours in that on each trial a component of the preference matrix must be predicted instead of an item selected to be recommended to a given user.

Matrix factorization and structured bandit formulations have often been used to frame the design of RS algorithms – see, e.g., Koren et al. (2009); Dabeer (2013); Zhao et al. (2013); Wang et al. (2006); Verstrepen and Goethals (2014), and references therein. The relevant literature on RS is abundant, and we can hardly do it justice here. Yet, we observe that most of the classical investigations on RS are experimental in nature, while those on structural bandits (e.g., Bui et al. (2012); Maillard and Mannor (2014); Gentile et al. (2014); Li et al. (2016); Kwon et al. (2017); Jedor et al. (2019); Katariya et al. (2017); Lu et al. (2018); Hong et al. (2020); Jun et al. (2019); Trinh et al. (2020); Lu et al. (2021); Kang et al. (2022); Pal et al. (2023) or on MC Keshavan et al. (2010); Biau et al. (2010); Rohde and Tsybakov (2011); Aditya et al. (2011); Candes and Recht (2012); Hazan et al. (2012);

Negahban and Wainwright (2012); Jain et al. (2013); Herbster et al. (2020) do not readily apply to our adversarial scenarios.

Appendix B. Proofs

This appendix contains the complete proofs of all our claims.

B.1. Proof of Theorem 2

Proof We shall show that UC and IC have expected regret bounds of

$$\mathbb{E}[R] = \mathcal{O}(C(M+N)) \quad \text{and} \quad \mathbb{E}[R] = \mathcal{O}(D(M+N)) \,,$$

respectively.

Let us assume that on every trial t there exists an item $j \in \mathcal{I}_t$ which i_t likes and has not been recommended to i_t before. This is without loss of generality since on any trials in which this does not hold the omniscient oracle (when suggesting liked items before disliked items) is forced to make a mistake.

Let Λ be the value of ℓ^* on trial T. Given a level $\ell' \in [\Lambda]$ we will bound the number of mistakes made on trials of the following types:

- Trials corresponding to Line 3 (of the pseudocode in Algorithm 1) with $k = \ell'$. Given such a trial t, we have $j_t \in \mathcal{P}_{\ell'}$ and if a mistake is made j_t is removed from $\mathcal{P}_{\ell'}$ so only one mistake can be made per item. Hence, no more than N mistakes are made on such trials.

- Trials corresponding to Line 4 with $\ell = \ell' - 1$. There is at most one such trial per user and hence no more than M mistakes are made on such trials.

- Trials corresponding to Line 5 with $\ell = \ell' - 1$. On such a trial t, item j_t is selected uniformly at random from $\mathcal{R}$. Since (by the initial assumption) there exists an item $j \in \mathcal{R}$ with $L_{i_t,j} = 1$, there are in expectation at most N such trials t until $L_{i_t,j_t} = 1$. Once this happens, there are no more trials of this type, and hence there are at most N such trials in expectation.

Thus there are in expectation $\mathcal{O}(M+N)$ mistakes in trials of the above types for each $\ell' \in [\Lambda]$, implying

$$\mathbb{E}[R] = \mathcal{O}(\mathbb{E}[\Lambda](M+N)) \,.$$

Therefore, all we now need to prove is that $\Lambda = \mathcal{O}(C)$ and $\Lambda = \mathcal{O}(D)$ for UC and IC, respectively.

To this effect, recall that given a level $\ell' \in [\Lambda]$ we denote by $u_{\ell'}$ the user that created that level (i.e. $u_{\ell'} := i_t$ when t is the trial on which level ℓ' was created).

A crucial property needed to prove this is what we call the *separation property*:

Given $\ell', \ell'' \in [\Lambda]$ with $\ell'' > \ell'$ and $u_{\ell''} \in \mathcal{U}_{\ell'}$, there exists some $i \in \mathcal{U}_{\ell'}$ with $L_{i,r_{\ell''}} = 0$.

To see why this property holds, first let t be the trial on which $u_{\ell''}$ creates level ℓ'' (so $u_{\ell''} = i_t$). Since $\ell' < \ell''$, on such round t we have, direct from the algorithm, that either $u_{\ell''} \notin \mathcal{U}_{\ell'}$ or there is no item in $\mathcal{P}_{\ell'} \cap \mathcal{I}_t$ that has not yet been recommended to $u_{\ell''}$. So since $u_{\ell''} \in \mathcal{U}_{\ell'}$, and since $r_{\ell''}$ is recommended to $u_{\ell''}$ on trial t and hence not recommended to $u_{\ell''}$ before, we must have that $r_{\ell''} \notin \mathcal{P}_{\ell'}$ on trial t. But for this to happen there must exist a user $i \in \mathcal{U}_{\ell'}$ with $L_{i,r_{\ell''}} = 0$, as claimed.

Now let the symbol $\equiv$ denote that two users or items are equivalent with respect to the matrix $\boldsymbol{L}$.

Let us first focus on IC. Suppose, for contradiction, that we have $\ell', \ell'' \in [\Lambda]$ with $\ell'' > \ell'$ and $r_{\ell''} \equiv r_{\ell'}$. Since $L_{u_{\ell''}, r_{\ell''}} = 1$ we also have $L_{u_{\ell''}, r_{\ell'}} = 1$, and hence $u_{\ell''} \in \mathcal{U}_{\ell'}$. This implies, via the separation property, that there exists $i \in \mathcal{U}_{\ell'}$ with $L_{i, r_{\ell''}} = 0$, so choose such a i. Since $i \in \mathcal{U}_{\ell'}$ this gives $L_{i, r_{\ell'}} = 1$, which contradicts the fact that $r_{\ell'} \equiv r_{\ell''}$. So all the representatives of the different levels come from different clusters which gives us $\Lambda \leq D$, as required.

We now turn our attention to UC. We will show that for all $\ell', \ell'' \in [\Lambda]$ with $\ell'' > \ell'$ we have either $\mathcal{U}_{\ell''} \cap \mathcal{U}_{\ell'} = \emptyset$ or $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$, where the subset property is strict (i.e., $\mathcal{U}_{\ell''} \neq \mathcal{U}_{\ell'}$). To show this, suppose that $\mathcal{U}_{\ell''} \cap \mathcal{U}_{\ell'} \neq \emptyset$. Choose $i \in \mathcal{U}_{\ell''} \cap \mathcal{U}_{\ell'}$. Since $i \in \mathcal{U}_{\ell'}$ we have $L_{i, r_{\ell''}} = L_{u_{\ell'}, r_{\ell''}}$ for all $\ell'' \in [\ell']$, and since $i \in \mathcal{U}_{\ell''}$ we have $L_{i, r_{\ell''}} = L_{u_{\ell''}, r_{\ell''}}$ for all $\ell'' \in [\ell'']$. Thus, since $\ell'' > \ell'$ we have $L_{u_{\ell''}, r_{\ell''}} = L_{i, r_{\ell''}} = L_{u_{\ell'}, r_{\ell''}}$ for all $\ell'' \in [\ell']$, which in turn implies that $u_{\ell''} \in \mathcal{U}_{\ell'}$. Hence, for all $i \in \mathcal{U}_{\ell''}$ and $\ell'' \in [\ell']$ we have $L_{i, r_{\ell''}} = L_{u_{\ell''}, r_{\ell''}} = L_{u_{\ell'}, r_{\ell''}}$ so that $i \in \mathcal{U}_{\ell'}$. This implies that $\mathcal{U}_{\ell''} \subseteq \mathcal{U}_{\ell'}$. Hence, since $u_{\ell''}$ is trivially contained in $\mathcal{U}_{\ell''}$ it is also contained in $\mathcal{U}_{\ell'}$ which implies, by the separation property, that there exists a user $i \in \mathcal{U}_{\ell'}$ with $L_{i, r_{\ell''}} = 0$. Consider such an i. Since (directly from the algorithm) we have $L_{u_{\ell''}, r_{\ell''}} = 1$ we then have $L_{i, r_{\ell''}} \neq L_{u_{\ell''}, r_{\ell''}}$, so that $i \notin \mathcal{U}_{\ell''}$. Hence $\mathcal{U}_{\ell''} \subseteq \mathcal{U}_{\ell'}$, and there exists $i \in \mathcal{U}_{\ell'} \setminus \mathcal{U}_{\ell''}$ which implies that $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$, as required.

We have just shown that for all $\ell', \ell'' \in [\Lambda]$ with $\ell'' > \ell'$ we have either $\mathcal{U}_{\ell''} \cap \mathcal{U}_{\ell'} = \emptyset$ or $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$. We call this property the *tree property*. We will now construct a directed graph (see Figure 2 for an example), whose nodes are sets, as follows. For all $\ell'' \in [\Lambda]$ we have that $\mathcal{U}_{\ell''}$ is a node in the graph and that:

- If there exists $\ell' \in [\Lambda]$ with $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$ then the (unique) parent of $\mathcal{U}_{\ell''}$ is $\mathcal{U}_{\ell'}$ for the maximum such ℓ';

- If there does not exist such a level ℓ' then $\mathcal{U}_{\ell''}$ has no parent.

Note that, by the tree property, if $\mathcal{U}_{\ell'}$ is the parent of $\mathcal{U}_{\ell''}$ then (since $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$) we have $\ell' < \ell''$, thereby making the graph acyclic. Moreover, since each node has at most one parent the graph is a forest.

Suppose we have $\ell', \ell'' \in [\Lambda]$ such that $\ell'' > \ell'$ and that $\mathcal{U}_{\ell'}$ and $\mathcal{U}_{\ell''}$ are both roots. Since $\mathcal{U}_{\ell'}$ is a root we must have that $\mathcal{U}_{\ell'} \not\subset \mathcal{U}_{\ell''}$ so by the tree property $\mathcal{U}_{\ell'} \cap \mathcal{U}_{\ell''} = \emptyset$ holds. Now suppose we have $\ell', \ell'' \in [\Lambda]$ such that $\ell'' > \ell'$ and that $\mathcal{U}_{\ell'}$ and $\mathcal{U}_{\ell''}$ are siblings. Let ℓ'' be such that $\mathcal{U}_{\ell''}$ is the parent of these siblings. We must have $\mathcal{U}_{\ell'} \subset \mathcal{U}_{\ell''}$ so, again by the tree property, we have $\ell'' < \ell'$. We also must have that ℓ'' is the maximum element ℓ''' of $[\Lambda]$ such that $\mathcal{U}_{\ell'} \subset \mathcal{U}_{\ell'''}$. These two properties imply that $\mathcal{U}_{\ell''} \not\subset \mathcal{U}_{\ell'}$ and hence, by the tree property, $\mathcal{U}_{\ell'} \cap \mathcal{U}_{\ell''} = \emptyset$. This shows that any two roots, and any two siblings, correspond to disjoint sets.

For all $\ell', \ell'' \in [\Lambda]$ such that $\mathcal{U}_{\ell''}$ is the only child of $\mathcal{U}_{\ell'}$, create a new node $\mathcal{U}_{\ell'} \setminus \mathcal{U}_{\ell''}$ and make it a child of $\mathcal{U}_{\ell'}$. Since, here, $\mathcal{U}_{\ell''} \subset \mathcal{U}_{\ell'}$, such new nodes are non-empty, and hence, because for all $\ell' \in [\Lambda]$ we have $u_{\ell'} \in \mathcal{U}_{\ell'}$, all nodes are non-empty. Note that the property that any pair of siblings or roots are disjoint still holds so, since any node is a subset of each of its ancestors, all leaves are disjoint. Also, for all $\ell' \in [\Lambda]$ and $i \in \mathcal{U}_{\ell'}$ we have $i' \in \mathcal{U}_{\ell'}$ for all $i' \in [N]$ with $i' \equiv i$. This implies that each leaf of the forest contains a user cluster as a subset and hence that C is at least as large as the number of leaves. Since all the internal nodes of the forest have at least two children and the number of nodes in the forest is no less than Λ, we must have at least $\Lambda/2$ leaves. Hence $\Lambda \leq 2C$, as required. ∎

B.2. Proof of Theorem 3

Proof Let $E := \min\{C, D\}$. We will construct our $M \times N$ matrix $\boldsymbol{L}$ such that it is both E-user clustered and E-item clustered, which implies $\boldsymbol{L}$ is also (C, D)-biclustered.

First consider the case that $M \geq N$. Without loss of generality, assume that N is a multiple of E. For all $a \in [E]$ let $\boldsymbol{v}_a$ be the N-component vector such that for all $j \in [N]$ we have $v_{a,j} := 1$ if and only if

$$(a - 1)(N/E) < j \leq aN/E .$$

Define $T := ME$ and for all $i \in [M]$ and $t \in [T]$ with $(i - 1)E < t \leq iE$, let $i_t := i$. For all $i \in [M]$ we will choose some $a_i \in [E]$ in a way that is dependent on the algorithm and set the i-th row of $\boldsymbol{L}$ equal to $\boldsymbol{v}_{a_i}$. Note that we can always choose a_i such that in expectation $\Omega(E)$ mistakes are made in the E trials t for which $i_t = i$. Since $N/E \geq E$, an omniscient oracle would make no mistakes, and hence the expected regret of the learner is equal to its expected number of mistakes, which is $\Omega(ME)$ as required.

We now turn to the case that $N \geq M$. Without loss of generality, assume that N is a multiple of E and assume $M = E^2$ (since for any $i \in [M]$ with $i > E^2$ we will be able to choose the i-th row of $\boldsymbol{L}$ arbitrarily). For all $a \in [E]$ define $\mathcal{X}_a$ to be the set of all $i \in [M]$ such that

$$(a - 1)E < i \leq aE .$$

We will construct our matrix $\boldsymbol{L}$ so that for all $a \in [E]$ there exists an N-component vector $\boldsymbol{w}_a$ such that for all $i \in \mathcal{X}_a$ the i-th row of $\boldsymbol{L}$ is equal to $\boldsymbol{w}_a$. Note that $\boldsymbol{L}$ will then be E-user clustered as required. We set our time horizon $T := NE$. Our user sequence is defined as follows. For all $i \in [M]$ we have that $i_t := i$ for all $t \in [T]$ with

$$(i - 1)N/E < t \leq iN/E .$$

For all $a \in [E]$, let $\mathcal{Z}_a$ be the set of trials $t \in [T]$ with $i_t \in \mathcal{X}_a$, noting that $|\mathcal{Z}_a| = N$ and the trials in $\mathcal{Z}_a$ come directly before those of $\mathcal{Z}_{a+1}$.

We now turn to the construction of the vectors $\{\boldsymbol{w}_a \mid a \in [E]\}$. To do so, we will construct, in order, a sequence of sets $\{\mathcal{W}_a \mid a \in [E]\}$ where, for all $a, a' \in [E]$ with $a' \neq a$, we have $\mathcal{W}_a \subseteq [N]$ and $|\mathcal{W}_a| = N/E$ and $\mathcal{W}_{a'} \cap \mathcal{W}_a = \emptyset$.

For all $a \in [E]$ the vector $\boldsymbol{w}_a$ is defined so that for all $j \in [N]$ we have

$$w_{a,j} := [\![j \in \mathcal{W}_a]\!] .$$

Suppose we have constructed $\mathcal{W}_{a'}$ for all a' less than some $a \in [E]$. Take an arbitrary set $\mathcal{W}' \subseteq [N]$ with $|\mathcal{W}'| = N/E$ and $\mathcal{W}' \cap \mathcal{W}_{a'} = \emptyset$ for all $a' \in [a - 1]$. Suppose that $\mathcal{W}_a$ is set equal to $\mathcal{W}'$ and the learning algorithm is run. Given some trial $t \in \mathcal{Z}_a$, let $\mathcal{Y}_t$ be the set of all items $j \in \mathcal{W}'$ such that there exists a trial $t' \in \mathcal{Z}_a$ with $t' < t$ and $j_{t'} = j$. Let s be the first trial in $\mathcal{Z}_a$ in which

$$|\mathcal{Y}_s| = N/(4E)$$

(or $\max \mathcal{Z}_a$ if no such s exists), and let $\mathcal{V}$ be the set of all $t \in \mathcal{Z}_a$ with $t < s$. The only trials $t \in \mathcal{V}$ in which the algorithm (with knowledge of $\mathcal{W}_{a'}$ for all $a' \in [a - 1]$) can be assured of not making a mistake are contained in the set of trials $t' \in \mathcal{V}$ such that there exists an item $j \in \mathcal{Y}_s$ that has not been recommended to $i_{t'}$ before trial t'. Since $|\mathcal{Y}_s| \leq N/(4E)$ and there are only E users i in which

$i_t = i$ for some $t \in \mathcal{Z}_a$, there are at most $N/4$ trials in $\mathcal{V}$ in which the algorithm is assured of not making a mistake. Similarly there are at most $N/4$ trials in $\mathcal{V}$ in which no mistakes are made. As we shall see, $N/4$ is small enough that we can ignore such trials. On all other trials in $\mathcal{V}$ the algorithm must search for an item in $\mathcal{W}'$. Since $\mathcal{W}'$ is an arbitrary subset of $N - (a-1)N/E$ elements with cardinality N/E we can choose $\mathcal{W}'$ in such a way that there are, in expectation,

$$\Omega\Big(|\mathcal{Y}_s|(N - (a-1)N/E)/(N/E)\Big) = \Omega(N(E-a)/E)$$

such trials in which mistakes are made. This is because (from above) there are at most $N/4$ trials in $\mathcal{V}$ in which mistakes are not made, and there exists a constant θ such that

$$\theta N(E-a)/E + N/4 \le N \ ,$$

N being the number of trials in $\mathcal{Z}_a$.

Summing the above mistake lower-bounds over all $a \in [E]$ gives us a total mistake bound of $\Omega(NE)$. Since, for all $a \in [E]$, we have $|\mathcal{W}_a| = N/E$, and each user is queried N/E times, an omniscient oracle would make no mistakes so the expected regret is equal to the expected number of mistakes which is $\Omega(NE)$ as claimed. Moreover, for any item $j \in [N]$, any $a \in [E]$ and any user $i \in \mathcal{X}_a$ we have $L_{i,j} = [\![j \in \mathcal{W}_a]\!]$ so since the sets $\{\mathcal{W}_a \mid a \in [E]\}$ partition $[N]$ we have that $\boldsymbol{L}$ is E-item clustered as required. ∎

B.3. Proof of Theorem 4

Proof We will first analyze UIE and then show how to modify the analysis for UE.

Recall that for all users $i \in [M]$ the values ω_i and ξ_i are the number of times that user i is queried and the number of items that user i likes, respectively. We can assume without loss of generality that $\omega_i \le \xi_i$ for all users $i \in [M]$, so that the regret is the number of mistakes. This is because if, on some trial t, there is no item that i_t likes and has not been recommended to i_t so far, then on such a trial the omniscient oracle (when suggesting liked items before disliked items) would incur a mistake. Note that this assumption means that on every trial t there exists an item j that i_t likes and has not been recommended to i_t so far. This assumption also entails that the regret of ORCA* is equal to its number of mistakes.

Let $\mathcal{H}$ be the set of trials t in which Line 7 of the pseudocode in Algorithm 2 is invoked and $L_{i_t,j_t} = 1$. Let $\mathcal{H}^*$ be the set of trials $t \in \mathcal{H}$ in which $\gamma = 1$ on trial t. Note that on each trial in $\mathcal{H}^*$ a level is created, and hence $|\mathcal{H}^*| = \Lambda$ where Λ the value of ℓ^* on the final trial T.

We will now bound the expected number of mistakes in terms the cardinality of the above sets. To do this, we consider a trial t in which a mistake is made. We have the following possibilities on trial t:

- The condition in Line 3 of Algorithm 2 is true. In this case $j_t \in \mathcal{F}$. For all $j \in \mathcal{F}$ we have that j was added to $\mathcal{F}$ on some trial in $\mathcal{H}$, and we know that the number of rounds t in which $j_t = j$ is bounded from above by M. Hence there can be at most $M|\mathcal{H}|$ such trials t.

- The condition in Line 4 holds. In this case $i_t \in \mathcal{E}$. For all $i \in \mathcal{E}$ we have that i was added to $\mathcal{E}$ on some trial in $\mathcal{H}$, and we know that the number of trials t in which $i_t = i$ is bounded from above above by N. Hence there can be at most $N|\mathcal{H}|$ such trials t.

- The condition in Line 5 holds. Let $j := j_t$ and let ℓ be the value of ℓ_{i_t} on trial t. We must have that $j_t \in \mathcal{P}_\ell$, and hence that

$$c_{\ell, j_t} \leq 2\psi$$

at the start of trial t. But c_{ℓ, j_t} is increased by one on such a trial, which means there can be no more than 2ψ trials t with $j_t = j$ and $\ell_{i_t} = \ell$. Note that each level $\ell \in [\Lambda]$ is created on a trial in $\mathcal{H}^*$ so there are at most

$$2\psi N |\mathcal{H}^*|$$

mistakes made on trials in which Line 5 applies.

- The condition in Line 6 is true. For every level $\ell \in [\Lambda]$ and every user $i \in [M]$ there is at most one such trial t with $i_t = i$ and $\ell_i = \ell$ (on trial t). Hence there are no more than $M\Lambda$ trials in total. Note that each level $\ell \in [\Lambda]$ is created on a trial in $\mathcal{H}^*$ so there are at most

$$M |\mathcal{H}^*|$$

mistakes made on trials in which Line 6 applies.

- The condition in Line 7 holds. Suppose t' is a trial in which the condition in Line 7 is true but $L_{i_{t'}, j_{t'}} = 1$. This means that $t' \in \mathcal{H}$, so there cannot be more than $|\mathcal{H}|$ such trials t'. But given an arbitrary trial t' in which that condition holds, the probability that $L_{i_{t'}, j_{t'}} = 1$ is at least $1/N$ (since $\omega_{i_{t'}} \leq \xi_{i_{t'}}$). This implies that there are, in expectation, at most

$$N \mathbb{E}[|\mathcal{H}|]$$

trials in which Line 7 applies and a mistake is made.

Putting together, we have so far shown that:

$$\mathbb{E}[R] = \mathcal{O}((M + \psi N)\mathbb{E}[|\mathcal{H}^*|] + (M + N)\mathbb{E}[|\mathcal{H}|]) .$$

Recall that given a user $i \in [M]$, its perturbation level δ_i is the number of items $j \in [N]$ in which $L_{i,j} \neq L_{i,j}^*$. Given a trial $t \in [T]$, let ρ_t be the number of items that user i_t likes and have not been recommended to them so far. Let $\mathcal{H}^\bullet$ be the set of trials $t \in \mathcal{H}$ with $L_{i_t, j_t}^* = 0$ and $\rho_t > 2\delta_{i_t}$, and $\mathcal{H}^\circ$ be the set of trials $t \in \mathcal{H}$ with $L_{i_t, j_t}^* = 0$ and $\rho_t \leq 2\delta_{i_t}$.

Let $\mathcal{G}$ be the set of items which are *good* (i.e., not bad). We call a non-empty set $\mathcal{K} \subseteq \mathcal{G}$ a *cluster* if and only if for all pairs of items $j, j' \in \mathcal{K}$ we have that the j-th and j'-th columns of $\boldsymbol{L}^*$ are identical and, in addition, for all items $j'' \in \mathcal{G} \setminus \mathcal{K}$, the j-th and j''-th columns of $\boldsymbol{L}^*$ differ. Note that there are no more than D clusters. Given a cluster $\mathcal{K}$, we define $\mathcal{K}'$ to be the set of all $t \in \mathcal{H}$ with $j_t \in \mathcal{K}$ and such that i_t is *good* (that is, not bad).

Let us now focus on a specific cluster $\mathcal{K}$ and define

$$\tau := \min(\mathcal{K}' \cap \mathcal{H}^*) ,$$

with the convention that the minimizer of the empty set is ∞. Let s be the level created on trial τ. We then partition $\mathcal{K}'$ into the following sets:

- $\mathcal{K}_1$ is the set of all $t \in \mathcal{K}'$ with $t < \tau$;

- $\mathcal{K}_2$ is the set of all $t \in \mathcal{K}'$ with $t \notin \mathcal{H}^\bullet \cup \mathcal{H}^\circ \cup \mathcal{H}^*$ and $t \geq \tau$;

- $\mathcal{K}_3$ is the set of all $t \in \mathcal{K}' \cap \mathcal{H}^*$ with $t \notin \mathcal{H}^\bullet \cup \mathcal{H}^\circ$ (noting this implies $t \geq \tau$);

- $\mathcal{K}_4$ is the set of all $t \in \mathcal{K}' \cap \mathcal{H}^\bullet$ with $t \geq \tau$;

- $\mathcal{K}_5$ is the set of all $t \in \mathcal{K}' \cap \mathcal{H}^\circ$ with $t \geq \tau$.

We will next analyze how much each of these sets contributes to the above regret bound.

Every $t \in \mathcal{H}$ has a $1/\psi$ probability of being in $\mathcal{H}^*$, which implies that the expected cardinality of $\mathcal{K}_1$ is at most ψ. Since each element of $\mathcal{K}_1$ is not in $\mathcal{H}^*$, it contributes $\mathcal{O}(M + N)$ to the regret bound. Hence, the overall contribution of $\mathcal{K}_1$ to the regret bound is in expectation equal to

$$\mathcal{O}(\psi(M + N)) \, .$$

We will now show that for all $j \in \mathcal{K}$ we always have

$$c_{s,j} \leq 2\psi \, .$$

To see this, take such a j and suppose we have a round $t \in [T]$ in which $c_{s,j}$ is incremented. Note that on such a t we necessarily have $L_{i_t,r_s} = 1$ and $L_{i_t,j} = L_{i_t,j_t} = 0$. We have the following two possibilities:

- If $L^*_{i_t,r_s} = 0$ then since $L_{i_t,r_s} = 1$ we have $L^*_{i_t,r_s} \neq L_{i_t,r_s}$. Since $r_s = j_\tau$ and $\tau \in \mathcal{K}'$ we have that $r_s \in \mathcal{K}$ so r_s is good, and hence there can be no more than ψ such trials.

- If $L^*_{i_t,r_s} = 1$ then since $r_s = j_\tau \in \mathcal{K}$ and $j \in \mathcal{K}$ we have $L^*_{i_t,j} = L^*_{i_t,r_s} = 1$. Since $L_{i_t,j} = 0$ we then have $L_{i_t,j} \neq L^*_{i_t,j}$. So, since j is good, there can be no more that ψ such trials.

This has proven our claim that for all $j \in \mathcal{K}$ the inequality $c_{s,j} \leq 2\psi$ holds deterministically.

We now analyze the cardinality of $\mathcal{K}_2$. To do this consider some arbitrary $t \in \mathcal{K}_2$. Since $t \in \mathcal{K}'$ we have $j_t \in \mathcal{K}$. Since $t > \tau$ and j_t was not recommended to i_t before trial t we must have that either $L_{i_t,r_s} = 0$ or $c_{s,j_t} > 2\psi$ (at some point). But $j_t \in \mathcal{K}$ so, by above, $c_{s,j_t} \leq 2\psi$ always holds so we must have $L_{i_t,r_s} = 0$. As $t \notin \mathcal{H}^\bullet \cup \mathcal{H}^\circ$ we have $L^*_{i_t,j_t} = 1$ so since $j_t, r_s \in \mathcal{K}$ we have $L^*_{i_t,r_s} = 1$. This implies $L_{i_t,r_s} \neq L^*_{i_t,r_s}$ and hence there can be at most ψ possible values of i_t.

We have just shown that the cardinality of $\{i_t \mid t \in \mathcal{K}_2\}$ is at most ψ. Now note that on any $t \in \mathcal{K}_2$ we have $t \notin \mathcal{H}^*$, so i_t is added to $\mathcal{E}$ and hence cannot be equal to $i_{t'}$ for any future trial $t' \in \mathcal{K}_2$ with $t' > t$. Hence, for each $t \in \mathcal{K}_2$ we have that i_t is unique, so the cardinality of $\mathcal{K}_2$ is equal to that of $\{i_t \mid t \in \mathcal{K}_2\}$, which is at most ψ. Since each $t \in \mathcal{K}_2$ is in $\mathcal{H}$ but not $\mathcal{H}^*$ it contributes $\mathcal{O}(M + N)$ to the regret, so that $\mathcal{K}_2$ contributes

$$\mathcal{O}(\psi(M + N))$$

to ORCA*'s the regret bound.

Suppose we have a trial $t \in \mathcal{K}_2 \cup \mathcal{K}_3 \setminus \{\tau\}$. If $\gamma = 0$ on trial t then $t \in \mathcal{K}_2$, while if $\gamma = 1$ we have $t \in \mathcal{K}_3$. Since the probability that $\gamma = 1$ is $1/\psi$ and $|\mathcal{K}_2| \leq \psi$ we have

$$|\mathcal{K}_3| = \mathcal{O}(1 + |\mathcal{K}_2|/\psi) = \mathcal{O}(1)$$

in expectation. Since each trial $t \in \mathcal{K}_3$ contributes $\mathcal{O}(M + \psi N)$ to the regret bound, this allows us to conclude that in expectation $\mathcal{K}_3$ contributes

$$\mathcal{O}(M + \psi N)$$

to ORCA*'s regret bound.

We now argue that we can exclude the contributions of $\mathcal{H}^\bullet$ (and hence also $\mathcal{K}_4$) to the regret bound. To this effect, suppose we have some $t \in \mathcal{H}$ with $\rho_t > 2\delta_{i_t}$. Note that j_t is drawn uniformly at random from the items not yet recommended to i_t so far, and $L_{i_t,j_t} = 1$. This implies that each of the ρ_t items j that user i_t likes and have not been recommended to i_t so far have a $1/\rho_t$ probability of being j_t. Since at most δ_{i_t} of these items j satisfy $L^*_{i_t,j_t} = 0$ we have that there is at most a

$$\delta_{i_t}/\rho_t < 1/2$$

probability that $L^*_{i_t,j_t} = 0$ (so that $t \in \mathcal{H}^\bullet$). Hence, $|\mathcal{H}^\bullet|$ affects the regret bound by a constant factor only, so that we can exclude the contribution of $\mathcal{K}_4$.

Finally, suppose we have some $t \in \mathcal{K}_5$. Note that $t \in \mathcal{H}^\circ$ which implies that i_t is bad. This means that $t \notin \mathcal{K}'$ which leads to a contradiction. The set $\mathcal{K}_5$ is therefore empty, hence it does not contribute to the regret.

Hence, the set $\mathcal{K}'$ contributes $\mathcal{O}(\psi(M + N))$ to the regret. Since the union of $\mathcal{K}'$ over all clusters $\mathcal{K}$ is equal to the set of all $t \in \mathcal{H}$ such that i_t and j_t are both good, we have shown that the total expected regret is bounded by $\mathcal{O}(D\psi(M + N))$ plus the contribution (to the regret) of all $t \in \mathcal{H}$ in which either i_t or j_t is bad. Hence, we now analyze the contribution to the regret of the latter kinds of rounds.

Consider some bad user i or bad item j, and let $\mathcal{B}$ be the set of trials $t \in \mathcal{H}$ with $i_t = i$ or $j_t = j$, respectively. Note that given $t \in \mathcal{B}$ with $t \notin \mathcal{H}^*$, in trial t it must happen that both i_t is added to $\mathcal{E}$ and j_t is added to $\mathcal{F}$. This means that there can be no further trials in $\mathcal{B}$, hence there is at most one trial in $\mathcal{B} \setminus \mathcal{H}^*$. This also implies that whenever we encounter a round $t \in \mathcal{B}$, there is a $1 - 1/\psi$ probability that there are no further trials in $\mathcal{B}$ and a $1/\psi$ probability that $t \in \mathcal{H}^*$. This implies that the expected number of trials in $\mathcal{B} \cap \mathcal{H}^*$ is bounded from above by

$$\sum_{a \in \mathbb{N}} 1/\psi^a = \frac{\psi}{\psi - 1} - 1 \leq 2/\psi \,,$$

where the inequality uses the condition $\psi \geq 2$. Since trials in $\mathcal{B} \setminus \mathcal{H}^*$ contribute $\mathcal{O}(M + N)$ to the regret, and trials in $\mathcal{B} \cap \mathcal{H}^*$ contribute $\mathcal{O}(M + \psi N)$, this shows that in expectation $\mathcal{B}$ contributes overall

$$\mathcal{O}(M + N)$$

to the regret.

Since there are m bad users and n bad items, the above shows that the set of trials $t \in \mathcal{H}$ such that i_t or j_t is bad contributes $\mathcal{O}((m + n)(M + N))$ to the regret. Hence,

$$\mathbb{E}[R] = \mathcal{O}((D\psi + m + n)(M + N)) \,,$$

as claimed.

We now single out the analysis changes for UE. First note that we can, without loss of generality, assume there are no bad items. This is because for any bad item j we can modify $\boldsymbol{L}^*$ so that its j-th

column is equal to that of $\boldsymbol{L}$ noting that $\boldsymbol{L}^*$ becomes $(D + n)$-item clustered and there are now no bad items.

Now observe that, since no items are ever added to $\mathcal{F}$, the condition in Line 3 of Algorithm 2 is never true, so our regret is now

$$\mathbb{E}[R] = \mathcal{O}((M + \psi N)|\mathcal{H}^*| + N|\mathcal{H}|)$$

so trials in $\mathcal{H} \setminus \mathcal{H}^*$ now only contribute $\mathcal{O}(N)$ instead of $\mathcal{O}(M + N)$. Since there are no bad items (so $n = 0$ and we can ignore in the analysis the fact that items are never added to $\mathcal{F}$) this change leads to a regret bound of the form

$$\mathbb{E}[R] = \mathcal{O}((D + m/\psi)(M + N\psi)) ,$$

as claimed. ∎

B.4. Doubling trick

We briefly detail the doubling trick needed to get rid of parameter ψ.

For each value of ψ in $\{2^a \mid a \in \mathbb{N},\, a \leq \log_2(M) + 1\}$ take an instance of ORCA* with that parameter value. On any trial we predict with and update only one instance a. We stay with instance a until a mistake is made. Once a mistake is made we set $a \leftarrow a + 1$ modulo $\lfloor \log_2(M) + 1 \rfloor$. This method allows us to achieve a regret bound that is only an $\mathcal{O}(\ln(M))$ factor off the regret bound of ORCA* (Theorem 4) with ψ therein replaced by the optimal ψ in hindsight.

Appendix C. Further Empirical Results

Table 1 contains the area under the learning curve for all algorithms we tested. Table 2 contains running times.

Table 1: Area under the curve (multiplied by 100) for all the methods we tested on the three versions of the MovieLens dataset. Standard errors over 30 repetitions are shown. Each column is tagged by the number N of randomly selected items, along with the resulting (average) number M of users. In bold is the best performance on each dataset, which turned out to be ORCAPOP*'s in all experiments we ran.

Method	$N = 50$ $M \approx 3376$	$N = 100$ $M \approx 4488$	$N = 200$ $M \approx 5290$
RANDOM	50.08±0.07	50.01±0.04	50.03±0.03
POP	60.11±0.96	59.49±0.88	58.69±0.55
ORCA-IC	77.04±0.56	77.61±0.46	77.82±0.30
ORCA*	74.58±0.67	76.87±0.39	78.06±0.26
ORCAPOP*	**79.34±0.56**	**80.02±0.42**	**79.77±0.30**
WRMF-4	59.71±0.27	66.94±0.26	73.64±0.22
WRMF-8	57.02±0.24	64.76±0.23	72.99±0.20
WRMF-16	53.10±0.18	61.11±0.23	70.52±0.20
WRMF-32	48.73±0.38	55.11±0.26	65.98±0.19

Table 2: Average execution time per round (in milliseconds) for all methods on three versions of the MovieLens dataset. Standard errors over 30 repetitions are shown. Each column is tagged by the number N of randomly selected items, along with the resulting (average) number M of users. WRMF methods turn out to be 50 to 100 times slower.

Method	$N = 50$ $M \approx 3376$	$N = 100$ $M \approx 4488$	$N = 200$ $M \approx 5290$
RANDOM	0.0038 ± 0.0007	0.0070 ± 0.0005	0.0102 ± 0.0005
POP	0.0072 ± 0.0016	0.0166 ± 0.0016	0.0295 ± 0.0019
ORCA-IC	0.0043 ± 0.0009	0.0083 ± 0.0008	0.0124 ± 0.0007
ORCA*	0.0037 ± 0.0008	0.0070 ± 0.0006	0.0093 ± 0.0007
ORCAPOP*	0.0074 ± 0.0014	0.0177 ± 0.0014	0.0310 ± 0.0018
WRMF-4	1.0552 ± 0.0938	1.0339 ± 0.0271	0.6993 ± 0.0150
WRMF-8	1.0880 ± 0.0746	0.9921 ± 0.1143	0.6934 ± 0.0153
WRMF-16	1.0449 ± 0.1052	1.0757 ± 0.0450	0.7061 ± 0.0175
WRMF-32	1.0050 ± 0.0690	1.0559 ± 0.1020	0.7555 ± 0.0169