On the Negative Impact of Social Influence in Recommender Systems: a Study of Bribery in Collaborative Hybrid Algorithms

Guilherme Ramos[a,c], Ludovico Boratto[b], Carlos Caleiro[c]

[a]*Instituto de Sistemas e Robótica, Dept. of Electrical and Computer Engineering, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal*
[b]*Data Science and Big Data Analytics, EURECAT (Centre Tecnológic de Catalunya) Carrer de Bilbao, 72 (Edifici A), 08005 Barcelona (Spain)*
[c]*SQIG - Instituto de Telecomunicações, Dept. of Mathematics, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal*

Abstract

Recommender systems are based on inherent forms of social influence. Indeed, suggestions are provided to the users based on the opinions of peers. Given the relevance that ratings have nowadays to push the sales of an item, sellers might decide to *bribe* users so that they rate or change the ratings given to items, thus increasing the sellers' reputation. Hence, by exploiting the fact that influential users can lead an item to get recommended, bribing can become an effective way to negatively exploit social influence and introduce a bias in the recommendations. Given that bribing is forbidden but still employed by sellers, we propose a novel matrix completion algorithm that performs hybrid memory-based collaborative filtering using an approximation of Kolmogorov complexity. We also propose a framework to study the bribery effect and the bribery resistance of our approach. Our theoretical analysis, validated through experiments on real-world datasets, shows that our approach is an effective way to counter bribing while, with state-of-the-art algorithms, sellers can bribe a large part of the users.

Keywords: Bribing, Algorithmic bias, Social influence

1. Introduction

Recommender systems have become essential tools to deal with the overwhelming amount of information available on the Web. Thanks to these infor-

Email addresses: guilherme.ramos@tecnico.ulisboa.pt (Guilherme Ramos), ludovico.boratto@acm.org (Ludovico Boratto), carlos.caleiro@tecnico.ulisboa.pt (Carlos Caleiro)

mation filtering tools, users are able to get in touch only with items which might be interesting for them [1].

Social influence can affect the behavior of users, either intentionally or unintentionally [2, 3]. Collaborative filtering algorithms, which are by far the most effective and employed class of recommender systems [4, 5], are inherently driven by social influence, since they automate the "follow by example" pattern [6]. Indeed, by being based on the opinions of similar people, recommendations usually reflect the governing behavior of the community [7]. This happens because of a phenomenon, known in the literature as *minority influence* [8].

Recommender systems are known to be very effective not only for the users but also for sellers since they can push the items' sales [1]. Therefore, sellers of individual items[1] might *bribe* users, by offering them money or goods to increase the ratings given to items and/or to write positive reviews. Indeed, companies have used Amazon Mechanical Turk to pay users to write good reviews about their items[2], and Amazon sellers have been caught paying their customers in order to get 5-star ratings[3]. While Amazon forbids to "pay or offer any incentive to a buyer for either providing or removing feedback"[4], the phenomenon is so relevant that, in 2017, Google Play updated their Developer Policies, by forbidding "inflating product ratings, reviews, or install counts by illegitimate means, such as fraudulent or incentivized installs, reviews, and ratings"[5], identifying incentivization as offering "money, goods, or the equivalent in exchange for ratings, reviews, or installs"[6].

Recent studies have characterized the impact of bribing in ranking systems and showed how to improve their robustness to this type of attacks [9, 10]. Given the previously mentioned key role of recommender systems in our decision making, if a seller bribes the users to change the ratings of its items, to make sure that they get recommended, this would allow him/her to increase his/her profit. However, this type of activity has strong ethical implications. It is known that users put greater trust in highly ranked results [11] (by definition, a recommender system presents the top-N items) and that they believe that rankings produced by algorithms are powerful, infallible, and thus unbiased [12, 13]. This is clearly not the case since, by bribing users, sellers can introduce bias in the main input considered by a recommender system (i.e., the user ratings). Hence, bribery can affect the list of items a user gets recommended, while he/she believes that he/she is getting suggested items based on unbiased

[1] In this work, we consider as *platform* the e-commerce store where the users can buy items (e.g., Amazon), and as *sellers* those who sell the individual items. Indeed, anyone can sell items on Amazon (https://www.amazon.com/sell-products-online/b?node=12766669011).

[2] https://techcrunch.com/2009/01/17/belkin-paying-65-cents-for-good-reviews-on-newegg-and-amazon/

[3] https://www.nytimes.com/2012/01/27/technology/for-2-a-star-a-retailer-gets-5-star-reviews.html

[4] https://sellercentral.amazon.com/gp/help/external/200386250

[5] https://play.google.com/about/storelisting-promotional/ratings-reviews-installs/

[6] http://android-developers.googleblog.com/2017/06/google-plays-policy-on-incentivized.html

opinions of their peers. Moreover, sellers bribe "powerful" users (i.e., those who can lead to changes in the recommended items, influencers); this phenomenon is known in the social influence literature as *abusive control* [3]. Since those users have already rated the items, the ones who receive biased recommendations are those who did not rate these items yet; hence, they are totally *unaware* of these dynamics in the rating process. In addition to the ethical issues, Chau, Ho, Ho, and Yao [14] highlight that irrelevant or biased recommendations lead to a significant decrease in terms of trust of the users in the recommender systems or in the provider.

Our contributions. Even though the literature has analyzed the robustness of recommender systems under several types of attacks [15], to the best of our knowledge, no study ever analyzed the impact of bribing users to change their ratings on recommender systems and how to counter this phenomenon. Considering the centrality of recommender systems and the ethical implications of this problem, we believe that characterizing the impact of bribing on recommender systems and providing actionable knowledge is of central importance both for the research community and for the industry.

The goal of an item's seller is to bribe users in a way that his/her item gains visibility in the recommendation lists that are presented to users. Either by making the item arise in a top-N recommendation list, or, if the item is already shown in the top-N, to give the item more relevance, making it appear in a higher position. Intuitively, to counter the effects of bribing in a recommender system, we need an approach that limits the impact of a change in terms of ratings, so that when a user is bribed, the biased rating has as little impact as possible on the recommendations produced by the algorithm. In order to achieve this goal, we also propose a novel hybrid Collaborative Filtering that explores the idea of Kolmogorov complexity to propose a new similarity measure. While a user/item profile is usually represented as a vector of all the ratings the user gave or the item received, thanks to our measure, the rating pattern of a user/item is represented as a compression of the string built with those ratings, and we use the size of this compression to compute the similarity between users. Hence, the changes in the ratings caused by bribing will have a lower impact in the size of the compressed string with respect to the actual user profile, thus countering its effects.

Moreover, we model and study the impact of bribery in our algorithm in a theoretical experimental framework. By identifying the bribable users and the revenue of bribing them, an item's seller can design optimal bribing strategies to increase his/her wealth and, consequently, make his/her product more visible by being more recommended to users.

After presenting our algorithm and modeling the bribing problem through our theoretical framework, we validate our proposal through experiments on real-world datasets, aimed at showing that our proposed algorithm is as effective as the state-of-the-art solutions, but more resistant to bribing.

In summary, the contributions of our paper are the following:

- we study, for the first time in the literature, the problem of bribing users

in the recommender systems context, identifying under which conditions sellers can make effective attacks;

- we propose a novel hybrid Collaborative Filtering algorithm to counter bribing;
- we identify, from the point of view of a seller of an item i, which users are profitable to bribe in the following cases: (i) bribe a user that already rated i to increase his/her rating; (ii) bribe a user that did not rate i to rate it; (iii) bribe a user that rated a competitor product to decrease his/her rating; (iv) bribe a user to rate a related item;
- we show that our algorithm is as effective as the state-of-the-art approaches while being more efficient;
- we illustrate our framework, by studying the impact of bribing in our algorithm and a real-world system (the SVD recommender system [16]), considering the MovieLens 100k and 1M datasets, showing that our approach is more robust against bribery.

Paper structure. We organize the rest of the paper as follows: in Section 2, we present related work; Section 3 introduces the notation used in this work; in Section 4, we present our hybrid Collaborative Filtering algorithm; in Section 5, we introduce the possible bribing strategies that can be adopted in recommender systems; Section 6 provides the detail of the experimental framework we built to validate our theoretical results; finally, in Section 7, we give some concluding remarks, and we draw avenues for future research.

2. Related Work

As we mentioned in the Introduction, bribery is a problem that has been widely-studied in ranking systems. In [9], the authors analyzed the resistance to bribery of two ranking systems, the first does the arithmetic average (AA) of the ratings of all users, and the second takes into account the influence network of each user, using the AA to compute the ranking for each network. They showed that the first system is bribable and that in the second system the bribery effect is attenuated. Their work assumes a fixed set of users with only one item to rate and that an item seller may bribe users to rate or increase the given rating to its item. In [10], the authors extend the previous work for the scenario where each user has a reputation and, hence, the AA is replaced by a weighted average (WA) of ratings by users' reputations. The authors also explored two scenarios. In the first scenario, the ranking is the WA of the ratings of all users, weighed by their reputations, and in the second case, the users are clustered into groups of users with similar tastes, being the ranking computed also as the WA of ratings, but for each cluster. Moreover, in this work, the authors do not assume that there is a fixed set of users, and they allow the coexistence of diverse items to be rated. Here, we explore similar ideas to [10], but in the context of Recommender

Systems. In this work, an item seller may perform a larger set of bribe actions: bribe users to rate/increase the given rating to that seller's item, and bribe users to rate or to increase/decrease a given rating to a competitors' item.

Previous work has studied shilling attacks on recommender systems and defined new metrics to quantify the impacts of this type of attack on known recommender algorithms, applying a detect+filtering approach in order to mitigate the spammers' effects on recommendations. For a detailed survey on this topic, we refer the reader to [15].

Recommender systems were originally conceived to deal with information overload and present us the items that are likely to be interesting [1]. While it is widely known that good recommender systems can be profitable for the platforms that employ them, it is also known that it is important to put users in the loop, by generating transparency (e.g., through explanations) and by trying to increase their trust in the system [7]. Nevertheless, part of the literature has started considering the profit that a recommender system can generate not only as a positive side effect (remember that the original main goal of a recommender is to support users in their decision making) but as an objective of the system itself. This led to a class of systems known as *profit-aware* (or *profitable*) *recommender systems*. Differently from the scenario that we are considering in this paper, in which the platform forbids bribing and the profit is generated by the seller of the items, in profitable recommender systems the platform generates a ranking that maximizes the profit that can be made. Jannach and Adomavicius [17] re-rank the items based on their profitability, showing that a low loss in accuracy (given by the fact that users are not necessarily recommended the items they like the most) leads to a great increase in terms of profit. Azaria et al. [18] present a model that recommends movies based on the revenue that they can generate. Das et al. [19] propose an approach that allows the platform to control the amount of deviation between profitable recommendation and a traditional one. Hosanagar et al. [20] and Panniello et al. [21] measure the current trust of the users in the system, thus pushing profitable recommendations when the trust is high and the users would buy anyway, and classic recommendations when the trust is low. Chen et al. [22] generate the recommendation list combining the probability for a user to purchase an item with the revenue it generates.

3. Notation

We denote a set of n users by $U = \{u_1, \ldots, u_n\}$, a set of m items by $I = \{i_1, \ldots, i_m\}$, a set of ratings given by users in U to items in I by $\mathcal{R}$, where $\mathcal{R} \subseteq U \times I \times \mathcal{D}$ and $\mathcal{D} =]0, 1]$ is the domain of the allowed ratings. We assume the domain to be $]0, 1]$ without loss of generality, since any domain of rating that does not contain 0 can be normalized by dividing by the maximum allowed rating, $R_\top$, and the domain is transformed in $[\frac{R_\perp}{R_\top}, 1]$, where $R_\perp$ denotes the smallest allowed rating. We denote the $n \times m$ matrix of ratings by M, where $M_{ui} = R_{ui}$ if $(u, i, R_{ui}) \in \mathcal{R}$ and $M_{ui} = 0$ whenever user u did not rate item i.

The entries take values on $[0, 1]$, that is $\mathcal{D} \cup \{0\}$. We denote by U_i the set of users that rated item $i \in I$, $U_i = \{(v, j, R_{vj}) \in \mathcal{R} : j = i\}$, and, similarly, we denote by I_u the set of items that user $u \in U$ rated, $I_u = \{(v, j, R_{vj}) \in \mathcal{R} : v = u\}$. Furthermore, we denote the set of users that rated items $i \in I$ and $j \in I$ by $U_{i,j} = U_i \cap U_j$. Analogously, we denote the set of items that users $u \in U$ and $v \in U$ rated in common by $I_{u,v} = I_u \cap I_v$. Given a recommendation list of items for a user u, sorted by increasing order of relevance, L_u, we denote by top-$N(u)$ the first N elements of the list (the N most relevant items to recommend to user u). Moreover, if item $i \in I$ appears in the first N elements of L_u, we denote by top-$N(u, i)$ its position on the top-$N(u)$ list. For instance if top-$3(u) = \{a, b, c\}$ then top-$3(u, b) = 2$, because item b is the second item to be recommended to user u. Finally, we use the semi-norm $\| \cdot \|_0$. Given a vector x, $\|x\|_0$ is the number of non zero entries of x.

4. A Novel Hybrid Algorithm to Counter Bribing

As we mentioned in the Introduction, the intuition behind this work is to develop an algorithm that can counter bribing, by allowing small changes in the rating to have as little impact as possible in the recommendation process. To achieve this goal, we propose a recommender system that can perform matrix completion as in hybrid memory-based CF approaches. Our approach uses two similarities, one between users, S_U, and another between items, S_I. Then, we predict the entry corresponding to user u and item i by assigning it a convex combination, by a parameter α, of two quantities:

1. a weighted average of the ratings of item i given by other users similar to user u;
2. a weighted average of the ratings that user u gave to other items by the similarities of the other items with item i.

Therefore, to generate a recommendation list for a user, we do not need to complete the entire matrix of ratings, but only the row respective to that user. This makes our recommender system fast and scalable for large datasets, because it may use essentially the amount of memory of the order of loading the dataset.

In what follows, we present the details of our prediction algorithm.

4.1. Similarity measure

As previously mentioned, the two components of our rating prediction algorithm consider two similarities, one between users (S_U) and the other between items (S_I). In this section, we present the measure that we propose to implement these similarities.

Given the description of a string, $\tilde{x}$, its *Kolmogorov complexity* [23], $K(\tilde{x})$, is the length of the smallest computer program that outputs x (i.e., $K(\tilde{x})$ is the length of the smallest compressor for $\tilde{x}$). Although this value is known to be non-computable, there are efficient and computable approximations based on compressors. Let C be a compressor, then $C(\tilde{x})$ denotes the length of the output string resulting from the compression of $\tilde{x}$ using C.

The measure we propose is inspired in the *normalized compression distance (NCD)*, see [24], but with a difference that leads to a better computational complexity, thus providing us the desired efficiency of the algorithm. Several works propose compression based similarity measures, we refer the reader for instance to [25, 26, 27] and references therin. We implement the description of a user as the string composed by the index of items he/she rated, followed by the respective rating. For instance, if user u rated the items $i_1, i_2, \ldots, i_l$, $l \leq M$ (sorted in lexicographical order), with ratings $R_{u1}, \ldots, R_{ul}$, respectively, then we write the description of user u as the string $\tilde{u} =$"$i_1 R_{u1} i_2 R_{u2} \ldots i_l R_{ul}$". Similarly, we implement the description of an item as the string composed by the *id* of the users who rated that item, followed by the rating they gave to it. Hence, if item i is rated by users $u_1, u_2, \ldots, u_s$, $s \leq N$ (sorted in lexicographical order), with ratings $R_{1i}, \ldots, R_{si}$, respectively, then we write the description of item i as the string $\tilde{i} =$"$u_1 R_{1i} u_2 R_{2i} \ldots u_s R_{si}$". Observe that from the description of one user we may not uniquely obtain a set of items and a set of ratings, i.e., the description of a user is not a bijection. In future work, we will explore the use of a bijective function, as it is done in [28]. While the classic NCD would compress every pair of users (and items) descriptions, in order to reduce the computational complexity we use the assumption that users do not behave randomly but follow rating patterns, as it is assumed in several approaches, such as in [16, 29][7]. Therefore, the aim of our metric is to capture the mentioned rating pattern in the size of the descriptions' compressions. It is important to stress out that this pattern would not be captured if we considered only the size of the description, without considering the compression (indeed, we would lose the concept of rating pattern). The similarity measure we propose is the following.

Compression similarity. *We define the* Compression similarity *(CS) as:*

$$CS(x, y) = \left(1 + |C(\tilde{x}) - C(\tilde{y})|\right)^{-1}.$$

Observe that the length of the compressions of two users ratings' descriptions, $\tilde{x}$ and $\tilde{y}$, is the same when they rate the same items with the same ratings, and $CS(x, y) = 1$. If two users rate the same set of items with different ratings or rate different items but with close ratings and with the same sequence, $CS(x, y)$ is close to 1 (although we want to stress that this last scenario is very unlikely to happen). It is also important to notice that by not having to compress pairs of users ratings' descriptions, we achieve a noticeable speedup in terms of computational complexity, as we analyze in Section 4.3.

To compress the strings (i.e., to implement the C compressor), we use the standard *zlib* library[8], which provides a lossless compression of the data it re-

[7] When two ratings patterns are random, our approach might identify them as similar patterns, but this is not assumed to happen (indeed, recommender systems base their effectiveness on the capability of characterizing the user preferences and his/her rating behavior), and the approaches we previously mentioned would behave in the same way in case of random patterns.

[8] `https://tools.ietf.org/html/rfc1950`

ceives as input. It is worth considering, in the future, the use of different compressors, or another string complexity approximations, in the calculation of this similarity measure.

According to the descriptions that we give as input, the compression similarity measure, CS, is used to compute two similarities, S_U and S_I, the similarity between users and between items, respectively.

4.2. Prediction algorithm

To predict the values of the non-rated items in the rating matrix R, we set each non-filled entry R_{ui} in the completed one, $\hat{R}$, as a convex combination of two quantities by a parameter α.

The first quantity is based on users that are similar to the user for which we are estimating the rating. A known phenomenon, as pointed in [30], is that users may vary the scale of the ratings that they use. To compensate for this effect, we do the following rating prediction, based on the similarities between users ($S_{U_{uv}}$), which also accounts for the number of items that have been rated by both users ($|I_{u,v}|$) and for each user's rating average (denoted as $\bar{R}_u$),

$$\text{pred}_U(u,i) = \bar{R}_u + \frac{\sum_{v \in U_i} |I_{u,v}|^2 S_{U_{uv}} (R_{vi} - \bar{R}_v)}{\sum_{v \in U_i} |I_{u,v}|^2 S_{U_{uv}}}.$$

The second summand is the sum of the ratings of each item $j \neq i$, weighed by the square of the number of users that rated both the items ($|U_{i,j}|$), together with the similarity between the two items i and j ($S_{I_{ij}}$)[9]:

$$\text{pred}_I(u,i) = \frac{\sum_{j \in I_u} |U_{i,j}|^2 S_{I_{ij}} R_{uj}}{\sum_{j \in I_u} |U_{i,j}|^2 S_{I_{ij}}}.$$

Lastly, for a fixed value of the parameter $0 \leq \alpha \leq 1$, we estimate each non filled matrix entry as

$$\hat{R}_{ui} = \alpha \ \text{pred}_U(u,i) + (1-\alpha) \ \text{pred}_I(u,i).$$

The previous steps are summarized in Algorithm 1. Observe that if $\alpha = 1$, it corresponds to user-based CF, and if $\alpha = 0$, it corresponds to item-based CF. Also, notice that we incorporate the quantities of common rated items and common rating users, respectively $I_{u,v}$ and $U_{i,j}$, to compensate the effect that the compression of the descriptions of the two entities may have the same value, being the number of ratings of each one different.

Our approach allows to decouple the problem into a set of independent user-by-user subproblems. Hence, to generate recommendations for a user, we do not need to complete the entire rating matrix, but only the corresponding matrix row.

[9]Note that the following formula does not subtract the average of the user's rating, as we did in the user-based component. This is a standard practice in item-based collaborative filtering [30]; moreover, if we subtract, then we get less effective results (these results are omitted due to space constraints and for readability purposes).

Algorithm 1 Rating prediction: KolMaC CS

input: α, *set of given ratings* R, user u, item i
output $\hat{R}_{ui} = \alpha \text{ pred}_U(u,i) + (1-\alpha) \text{ pred}_I(u,i)$
compute S_U from the training set
compute S_I from the training set
set $\hat{R} = R$
apply threshold α
for each user u **do**
  for each item i such that $\hat{R}_{ui} = \perp$ **do**
    if M_{uo} is not filled **then**
      set $\hat{R}_{ui} = \alpha \text{ pred}_U(u,i) + (1-\alpha) \text{ pred}_I(u,i)$
    end if
  end for
end for
output: $\hat{R}$

4.3. Complexity analysis

To compute the similarities between users, S_U, we pre-compute the quantity $C(\tilde{u})$ for each user $u \in U$. After, we do not need to build an $N \times N$ matrix where each entry $S_{U_{uv}} = \text{CS}(u,v)$ for each $u, v \in U$, because we can compute each of these entries in $\mathcal{O}(1)$ time by accessing the pre-computed values. At the end, the pre-computed values consist in compressing strings that are a partition of the string with all the ratings, which takes $\mathcal{O}(|\mathcal{R}|)$. *Mutatis mutandis*, for the time and space complexities of the items' similarities. Notice that this differs from the KNN methods [31, 32], where computing the similarity between all pairs of users, and items, takes $\mathcal{O}(MN^2)$, and $\mathcal{O}(M^2N)$. These similarities may be computed in $\mathcal{O}(N^2)$ and $\mathcal{O}(M^2)$, respectively, in the method we propose.

In order to compute the quantities $U_{i,j}$ and $I_{u,v}$ we may opt for different strategies, depending whether we want to optimize the time complexity, the memory complexity, or to achieve a good compromise between both. If we want to optimize the time complexity, we pre-compute the value $U_{i,j} = U_{j,i}$ for all $i, j \in I$ and store them in a sparse matrix which has, in the worst case, M^2 non-zero entries. Similarly, we may perform the same and build, in the worst case, a matrix with the values of $I_{u,v} = I_{v,u}$ with N^2 non-zero entries. The time complexity is the one of computing the product of two sparse matrices which indicate the items each user rated. In the opposite case, when we want to optimize memory, we compute $U_{i,j}$, and $I_{u,v}$ every time we need. If we want to predict k ratings, this leads to $\mathcal{O}(k \max\{N^2, M^2\})$. In the compromise scenario, the one we adopt, we may store the values of $U_{i,j}$ and $I_{u,v}$ in a dictionary, or a hash table, up to a certain amount of memory, and when we need to use such a value, either we access it in $O(1)$ if it is pre-computed, or if the value is not stored, we need to compute it (and store it if we did not exceed the maximum memory we want to spend). Nevertheless, Algorithm 1 is *embarrassingly parallel* [33]. In other words, we may have a process to compute $\text{pred}_U(u,i)$ and a

process to compute $\text{pred}_I(u, i)$ for each $u \in U$ and for each $i \in I$.

Next, we define a framework to study the bribing resistance of recommender system. Further, we use a simplified version of the proposed KolMaC CS, that allows us to determine under which conditions a seller can bribe a user to rate/change his/her rating with positive return.

5. Bribery in Recommender Systems

Here, we define a bribing strategy and the respective profit of playing it, in the scenario of a top-N recommender system, considering the algorithm we previously presented.

5.1. Initial Wealth

We define the *wealth* of item's $i \in I$ seller as the sum, for each user, of a value inversely proportional to the position where i appears in the top-N recommendation list of each user. This means that if item i does not appear in any top-N recommendation list, then the wealth of item's i seller is zero. Moreover, the wealth increases with the number of times i is recommended, by a value inversely proportional to the position in the recommendation list. Given that there does not exist a dataset that contains both the profit that items generate and the ratings that users gave to those items, assigning a wealth based on the position of the items is an accepted practice in the literature [17]. Hence, to simplify the calculations, we model the wealth generated by a user $u \in U$ as:

$$\mathcal{P}_N^i(u) = \begin{cases} 0 & \text{if } i \notin \text{top-}N(u) \\ 1/p & \text{if } i \in \text{top-}N(u) \text{ and top-}N(u, i) = p \end{cases}. \tag{1}$$

We recall the definition of top-$N(u, i)$ of Section 3. If item $i \in I$ appears in the first N elements of the recommendation list for user u, then the top-$N(u, i)$ is the position of item i on the top-$N(u)$ list.

Another possibility would be to consider that the previous quantity exponentially decreases with the position of the item in the top-N recommendation list, but the choice does not affect the reasoning of the theoretical results.

The *initial wealth* of seller of item i is given as the sum of (1) for each user, i.e.,

$$\mathcal{J}(i) = \sum_{u \in U} \mathcal{P}_N^i(u). \tag{2}$$

5.2. Bribing Strategies

We define a bribing strategy, played by item's $i \in I$ seller, as a matrix $\mathfrak{S}^i \in \mathbb{R}^{n \times m}$, where $\mathfrak{S}^i_{uj}$, with $u \in U$ and $j \in I$, is the change that item's i seller wants user u to do to his/her rating to item j. Therefore, $\mathfrak{S}^i_{uj} \in [R_{uj} + \frac{R_\perp}{R_\top}, 1 - R_{uj}]$. Furthermore, we assume that the invested value of item's i seller to bribe a user

$u \in U$ to change his/her rating by $\mathfrak{S}^i_{uj}$ is $|\mathfrak{S}^i_{uj}|$. This simplification captures the idea that the invested value should be proportional to the increase/decrease of the rating of the bribed user. In other words, item's i seller can bribe user u to downgrade his/her rating until the minimum allowed rating, $R_\perp$, or to upgrade it to the maximum allowed rating, $R_\top$. The number of bribed users in a strategy $\mathfrak{S}^i$ is $\|\mathfrak{S}^i\|_0$. We denote the value of $\mathcal{P}^i_N(u)$, after strategy $\mathfrak{S}^i$ being played, by $\mathcal{P}^{\mathfrak{S}^i}_N(u)$.

5.3. Cost of Playing a Strategy

The cost that seller of item i incurs in by playing bribing strategy $\mathfrak{S}^i$ is

$$\mathcal{C}(\mathfrak{S}^i) = \sum_{u \in U} \sum_{j \in I} |\mathfrak{S}^i_{uj}|. \tag{3}$$

5.4. Final Wealth

After seller of item i plays bribing strategy $\mathfrak{S}^i$, his/her *final wealth* is the new wealth minus the cost of the bribing strategy computed as in (3), i.e.,

$$\mathcal{J}(\mathfrak{S}^i) = \sum_{u \in U} \mathcal{P}^{\mathfrak{S}^i}_N(u) - \mathcal{C}(\mathfrak{S}^i). \tag{4}$$

In other words, the final wealth is the new wealth of seller of item i, after playing bribing strategy $\mathfrak{S}^i$, minus what he/she spent playing strategy $\mathfrak{S}^i$.

5.5. Profit

Now, we can define the *profit* of seller of item i after playing strategy $\mathfrak{S}^i$, which is the difference between the final and initial wealths, respectively (4) and (2), i.e.,

$$\pi(\mathfrak{S}^i) = \mathcal{J}(\mathfrak{S}^i) - \mathcal{J}(i). \tag{5}$$

We say that a bribing strategy $\mathfrak{S}^i$ is *profitable if* $\pi(\mathfrak{S}^i) > 0$ and it is *not profitable* otherwise.

In general, we may not be able to have a theoretical study about the bribing strategies that are profitable, $\pi(\mathfrak{S}^i) > 0$, not only due to the complicated computation that a recommender system performs to estimate ratings, but also due to the discrete and combinatorial nature of the definition of $\mathcal{P}^i_N$. However, we can find theoretical results if we consider a simple recommender system together with an approximation of $\mathcal{P}^i_N$.

Notice that, although the definition of wealth may accurately capture the real value of this quantity, it is quite difficult to use it for symbolic computations, due to its combinatorial nature. To overcome this issue and to derive the theoretical results, we consider approximating the value of $\mathcal{P}^i_N(u)$ by the estimated rating of user u to item i (denoted as $\hat{R}_{ui}$), i.e., $\mathcal{P}^i_N(u) = \hat{R}_{ui}$ if user u did not rate item i and $\mathcal{P}^i_N(u) = 0$, otherwise. This approximation approaches the scenario when N is ∞ in the considered top-N. This approximation also allows us to derive theoretical conclusions about bribing strategies, because the original $\mathcal{P}^i_N$ based

	i_1	i_2	i_3	i_4
u_1	5	–	4	3
u_2	4	–	–	4
u_3	5	2	4	–

Table 1: Synthetic dataset with $U = \{u_1, u_2, u_3\}$, $I = \{i_1, i_2, i_3, i_4\}$ and ratings from 1 to 5. The items that users did not rate are represented by "–".

on the top-N, although more realistic, implies sorting the rating estimations (to compute the top-N). Hence, its combinatorial nature makes the theoretical study of profitable bribing strategies untractable. Again, observe that we may also consider it as decreasing exponentially with this value, and the results we present can also be adapted to such scenario.

Another important observation is the fact that usually the real-world recommender systems are very intricate, which in turn makes the symbolic expression of a rating estimation unintelligible and impossible to use. Thereupon, we consider a simple recommender system which is a coarse approximation of the real-world used systems. This system estimates the rating of user u to item i as the combinations (by $\alpha \in [0, 1]$) of two quantities, the average of given ratings by user given by u, and the average of given ratings to item i, i.e.,

$$\hat{R}_{ui} = \frac{\alpha}{|U_i|} \sum_{v \in U_i} R_{vi} + \frac{1-\alpha}{|I_u|} \sum_{j \in I_u} R_{uj}. \tag{6}$$

Even though the recommender system considered in this framework is relatively simple, in Section 6 we will show that our theoretical results reflect those obtained by a more advanced recommender system.

To illustrate the use of $\mathcal{P}_N^i$ and its approximated version by the estimated ratings, consider the following scenario described in Table 1. Now, let us compute the initial wealth of item i_2 seller. Using Equation (6) and setting $\alpha = 0.5$ and $N = 2$. We have that $\hat{R}_{u_1 i_2} = \frac{0.5}{1}2 + \frac{1-0.5}{3}(5 + 4 + 3) = 3$, $\hat{R}_{u_2 i_2} = \frac{0.5}{1}2 + \frac{1-0.5}{2}(4 + 4) = 3$ and $\hat{R}_{u_2 i_3} = \frac{0.5}{2}(4 + 4) + \frac{1-0.5}{2}(4 + 4) = 4$. Hence, if we use the normalized rating estimations to compute the initial wealth of item i_2 seller, we obtain $\mathcal{J}(i_2) = \sum_{u \in U} \mathcal{P}_N^i(u)) = \frac{3}{5} + \frac{3}{5} = 1.2$. In the case that we use Equation (1) to compute the initial wealth, we get $\mathcal{J}(i_2) = \frac{1}{1} + \frac{1}{2} = 1.5$, because item i_2 is the first and only recommended item to user u_1, and it is recommended in second place to user u_2.

Considering a scenario like Amazon, in which the recommendation algorithm is known and fairly simple [34, 35] and the ratings of the competitors are publicly available (indeed, for each item we know how many users rated it and with which rating), it is possible for sellers to design profitable strategies to bribe users. Hence, we can study which are the profitable *elementary* bribing strategies, i.e., that consist in bribing one user. We now present the results we got from our theoretical analysis (proofs are reported in Appendix B, to improve the readability of the paper).

The first aspect we consider is if and when it is profitable for the seller of an item to bribe a user to increase/decrease the rating that he/she gave to that

item.

Proposition 1. *It is profitable for item's i seller to bribe any user $w \in U_i$ to increase the given rating, $\mathfrak{S}^i_{wi} > 0$* if and only if *$|U_i| < \frac{\alpha}{1+\alpha}|U|$, and it is not profitable to bribe w to decrease the given rating to item i, $\mathfrak{S}^i_{wi} < 0$.*

Now that we know when it is profitable for the seller of an item to bribe the users who already rated it, we analyze when it is profitable to bribe those who did not rate the seller's item.

Proposition 2. *If $w \notin U_i$, then it is profitable for item's i seller to bribe w if*

$$\mathfrak{S}^i_{wi}\frac{|U|-|U_i|}{|U_i|}\alpha > \left\lceil \frac{\alpha}{|U_i|(1+|U_i|)} \sum_{\substack{u \neq w \\ u \notin U_i}} R_{vi} + |\mathfrak{S}^i_{wi}| + \hat{R}_{wi} \right\rceil$$

However, a seller might also bribe the users who rated both his/her item and an item of a different seller, to make them upgrade/downgrade the rating given to another item. Hence, we further analyze under which conditions it is profitable for that seller to do so.

Proposition 3. *If $w \in U_i$, $k \in I_w$, and $k \neq i$, then it is not profitable for item's i seller to bribe w to downgrade the rating of item k, while it is profitable for item's i seller to bribe w to upgrade the rating of item k whenever $(|U| > 2) \wedge \left(\alpha > \frac{1}{|U|-1}\right) \wedge \left(|U_i| < \frac{\alpha}{\alpha+1}|U|\right)$.*

Although it may not seem intuitive, a seller may get more visibility for a item he/she sells by making products that the recommender system relates to that item more visible.

After computing when it is profitable for a seller to bribe the users who rated his/her item, so that they change the rating given to another item, we explore when it is profitable for the seller to bribe users who did not rate his/her item.

Proposition 4. *If $w \notin U_i$, $k \in I_w$, and $k \neq i$, then it is profitable for item i seller to bribe w to change the rating of item k* if and only if *$\alpha = 0$ and $|I_w| = 1$.*

The previous result agrees with the intuition that a seller bribing users to downgrade the rating they gave to other products can effectively give more visibility to the seller's product.

The last aspect we consider is how profitable it is for a seller to bribe a user to rate an item that he/she is not selling.

Proposition 5. *If $w \notin U_i$, $k \in I \setminus I_w$, and $k \neq i$, then it is profitable for item's i seller to bribe w to rate item k if $\alpha \neq 1$ and $\mathfrak{S}^i_{wk} > \frac{|I_w|}{|I_w|^2+1} \sum_{j \in I_w} R_{wj}$.*

An important observation to make is the following. Although the initial wealth decreases with the value of N (when considering the top-N version of the function $\mathcal{P}^i_N$), after bribing a user, the final wealth may be bigger for the

top-N_1 versus top-N_2 with $N_1 < N_2$. This may seem to be a paradox. However, if item i does not appear in top-N_1 of user u, but it appears in top-N_2 of user u at position $p > N_1$ then, if after the bribing the item appears at position $p' \leq N_1$, $\mathcal{P}^{\mathfrak{S}^i}_{N_1}(u) - \mathcal{P}^i_{N_1}(u) = 1/p' - 0$ and $\mathcal{P}^{\mathfrak{S}^i}_{N_1}(u) - \mathcal{P}^i_{N_1}(u) = 1/p' - 1/p$. We will recall this remark in the experimental results of Section 6.

5.6. Optimal bribing strategies

An important property that is worth noticing is the following.

Corollary 1. *Let item's $i \in I$ seller bribe users $u, v \in U$ and let $\mathfrak{S}^i_{ui;vi}$ denote the adopted strategy. We have the following:*

1. *if $u, v \in U_i$, then $\pi\left(\mathfrak{S}^i_{ui;vi}\right) = \pi\left(\mathfrak{S}^i_{ui}\right) + \pi\left(\mathfrak{S}^i_{vi}\right)$;*
2. *if $u, v \in U_i$, $j \in I_u$, and $k \in I_v$, then $\pi\left(\mathfrak{S}^i_{uj;vk}\right) = \pi\left(\mathfrak{S}^i_{uj}\right) + \pi\left(\mathfrak{S}^i_{vk}\right)$.*

Corollary 1 shows two cases where the profit of bribing two users is equal to the summation of the profits of bribing each user. In the case where the users are bribed to change a given rating, we can do such a decomposition. Otherwise, we have the case where an item seller wants to bribe users to rate a product they did not rate before. Here and in general, we cannot do the decomposition of the profit into the profits of each elementary strategy. The aforementioned occurs because we need to increment the number of raters of the item that the seller bribes the user to rate.

In summary, a seller can generate the optimal bribing strategy, in the scenario when we can decompose the profit given in Corollary 1, by selecting all the profitable users by decreasing value of profit, and bribe the users until either exhausting the profitable users or his/her wealth. In the case that a seller wants to bribe users to rate his/her product and the users did not rate it before, we have a more intricate scenario, and it must be solved case by case.

6. Experimental Framework

Next, we describe our framework, developed to setup the experiments and analyze the results. All experiments were done in a virtual server with 20 CPUs (Intel Core Processor, 64 bits), with 128Gb RAM, running Linux CentOS 7.4.1708, using Matlab 2016 and Python 3. The implementation of our algorithm is in Matlab 2016, and the benchmark recommender systems that we compare with are implemented in the Python scikit Surprise[10], in Python 3. We decided to use the Surprise framework because of our familiarity with the Python language and since it implements all the rating-prediction-based Collaborative Filtering algorithms (hence, we could compare our proposal against the same class of algorithms).

[10] `http://surpriselib.com/`

	ML–100K	ML–1M	ML–10M	ML–20M	JESTER
$\lvert U\rvert$	983	6040	71,567	138,493	24,983
$\lvert I\rvert$	1682	3952	65,133	131,262	100
$\lvert\mathcal{R}\rvert$	100,000	1,000,000	10,000,000	20,000,000	1,810,455
Rating scale	$[0, 5]$	$[0, 5]$	$[0, 5]$	$[0, 5]$	$]-10, 10[$

Table 2: Datasets details.

6.1. Datasets

We evaluated and validated our proposal on real-world datasets. We chose to consider datasets of different sizes, to show the effectiveness of our approach both in small- and large- scale data. We use MovieLens 100k (ML–100k), MovieLens 1M (ML–1M), MovieLens 10M (ML–10M), and MovieLens 20M (ML–20M)[11], all having ratings in $[1, 5]$, with $\perp = 0$. Further, we use the Jester's datasets (three datasets)[12]. The datasets consist in ratings to a set of 100 jokes, with continuous ratings in $]-10, 10[$, with $\perp = 99$. JESTER-1 has 24,983 users who rated 36 or more jokes. JESTER-2 consists of 23,500 users that rated 36 or more jokes. JESTER-3 has 24,938 users who rated between 15 and 35 jokes. Due to space constraints, since the results are very similar for all Jester datasets, we only report the ones for JESTER-1, which we denote by JESTER from now on. Table 2 contains a compact description of the datasets for which we report the results.

6.2. Evaluation metrics

To evaluate and compare the performance of the proposed algorithm, Algorithm 1, we use the 5-fold-cross-validation method on real and synthetic data. For the ML–100k, the dataset already provides a set of 5 train and test files. For the ML–1M, ML–10M, ML–20M and the three Jester's datasets we randomly split each of the original datasets in a set of 5 train/test files (with 80%/20% proportions). We use the *root-mean-square error (RMSE)* [29] and the *normalized discounted cumulative gain (nDCG)* [36] to evaluate the accuracy of our algorithm. Let R be the original matrix, R^* equal to R except on the missing entries of the test set $\mathcal{T}$, where it has the value $\perp$, and let $\hat{R}$ be the estimation of M by a matrix completion method when applied to R^*. The RMSE is given by

$$\mathrm{RMSE}(R, \hat{R}) = \sqrt{\frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (R_{ui} - \hat{R}_{ui})^2}.$$

Let $REL(u)$ denote the set of the first N more relevant items in $\mathcal{T}$ for user $u \in U$. Further, let

$$rel_i(u) = \begin{cases} 1, & \text{if the } i\text{-th element of top-}N(u) \text{ is in } REL(u) \\ 0, & \text{otherwise} \end{cases}.$$

[11] `http://movielens.umn.edu`
[12] `http://eigentaste.berkeley.edu/dataset/`

The nDCG@N for user u is computed as:

$$\text{nDCG@N}(u) = \frac{\text{DCG@N}(u)}{\text{IDCG@N}(u)},$$

where

$$\text{DCG@N}(u) = \sum_{i=1}^{N} \frac{rel_i(u)}{\log_2(i+1)},\ \text{IDCG@N}(u) = \sum_{i=1}^{|REL(u)|} \frac{1}{log_2(i+1)}.$$

Finally, let the set of users in the test set $\mathcal{T}$ be denoted by $\mathcal{T}_U = \{u : \exists_{i \in I}(u, i) \in \mathcal{T}\}$. We define the nDCG@N as

$$\text{nDCG@N} = \sum_{u \in \mathcal{T}_U} \frac{\text{nDCG@N}(u)}{|\mathcal{T}_U|}.$$

6.3. Experimental Strategy

We evaluated our proposal through three sets of experiments:

1. **Effectiveness evaluation.** We compare our proposal against the baselines, by measuring the RMSE and nDCG@25 obtained by each approach. We also evaluated the time and space complexity of each approach but, to improve the readability of the paper, these results are reported in Appendix A. The value of α chosen to run our algorithm is 0.3.
2. **Impact of bribing.** First, for the ML–100k and ML–1M datasets, we compute the percentage of users that are profitable for the seller of item $i = 1$ to bribe using as $\mathcal{P}_N^1(u)$ the top-5, the top-25, and the ratings estimation $\hat{R}$. We choose top-5 and top-25 to show how our system behaves with small and large recommendation lists, respectively. We compute this percentage by bribing each user:
   - that rated item 1, to increase his/her rating to the maximum rating, as in Proposition 1;
   - that did not rate item 1, to rate item 1 with the maximum rating, as in Proposition 2;
   - that rated the competitor seller's item 2, to downgrade item 2 rating to the minimum rating.

   We explore the last strategy to check if the result in Proposition 3 is a good approximation to the recommender system used in real-world scenarios. Because this proposition states that this strategy in never profitable in the simplified recommender system that we use to explore bribing properties. We do not test Propositions 4 and 5, because they represent very particular instances of users which are profitable to bribe and very connected with the approximated rating estimation we use. Hence, they do not occur often and a seller should be more interested in bribing according to Propositions 1-3.

Moreover, for each dataset, we compute the sum of the profits of strategy $\mathfrak{S}^1$, consisting in bribing the most profitable user at each time. Again, we explored the described bribing strategy using as $\mathcal{P}_N^1(u)$ the top-5, the top-25, and the ratings estimation $\hat{R}$. Notice that after the current most profitable user is bribed, the previous second most profitable user to bribe may no longer be the new user most profitable to bribe. This can occur because the ratings estimations are recomputed and, therefore, another user might be the new one which is more profitable to bribe. It is also worth noticing that such a bribing strategy, although optimal, is very time consuming, because each time we select a user to bribe we need to recompute all the rating predictions and recommendation lists for each user.

Hence, considering larger datasets in this set of experiments not only would not add any scientific value to this work (in both the datasets that we analyzed the trends in the results are clear), but it would also not be trivial, due to the challenges in the experimental setup we mentioned in the previous paragraph (i.e., the predictions' re-computation and the identification of the new most profitable user to bribe).

6.4. Experimental Results

Here, we present the results obtained in each set of experiments.

6.4.1. Effectiveness evaluation

Table 3 summarizes the RMSE values obtained for the real-world datasets, and Table 4 the nDCG@25 results obtained for the same datasets. We obtained the best RMSE result for one of the datasets, the Jester dataset. Considering the other datasets, two model-based algorithms (i.e., SVD and SVD++) achieve the best results in terms of RMSE. While our algorithm is not that far from the results obtained by these two algorithms. In terms of nDCG@25, results depicted in Table 4, we obtained results that compete with the benchmark algorithms. In addition to the intrinsic advantages that our algorithms has (such as the possibility to generate explainable recommendations), we also obtain other strong improvements in terms of efficiency, which will be shown in Appendix A. A sign of this lack of efficiency can be already noticed for the SVD++ algorithm (one of the best performing ones) and the largest datasets; more specifically, as the size of the data increases, the algorithm cannot return recommendations in a reasonable time, which is key in a rapidly-evolving environment such as the Web.

We note that we may have obtained these RMSE results because the majority of the compared methods assume that the matrix they complete is low rank, which might be the case in these datasets, but might not be true in general. Therefore, the RMSE results are comparable and of the same order as the best-reported ones.

Method	ML–100k	ML–1M	ML–10M	ML–20M	Jester
SVD	0.9396	0.8936	0.7972	**0.7859**	4.4957
SVD++	**0.9200**	**0.862**	**0.7965**	0.7887	4.5192
NMF	0.9634	0.9155	0.8708	0.8660	6.2372
Slope One	0.9454	0.9065	0.8607	0.8545	4.5187
Co-clustering	0.9678	0.9155	0.8819	0.8778	4.6634
KNN Basic	0.9789	0.9207	0.8194	0.8051	4.4786
KNN With Means	0.9514	0.9292	0.8211	0.8079	4.4800
KNN Baseline	0.9306	0.8949	0.8067	0.7990	4.4733
KolMaC CS	0.9582	0.9330	0.9711	0.9686	**4.4719**

Table 3: RMSE for the datasets detailed in Table 2.

Method	ML–100k	ML–1M	ML–10M	ML–20M	Jester
SVD	0.0968	0.0953	0.0747	0.0600	0.6091
SVD++	0.1099	0.1080	0.0806	0.0633	0.6144
NMF	0.0033	0.0025	5e−5	3e−6	0.5595
Slope One	0.0015	0.0009	7e−6	2e−7	0.6737
Co-clustering	0.0563	0.0622	0.0813	0.0998	0.7858
KNN Basic	0.2985	0.2515	0.2081	0.1840	0.7833
KNN With Means	**0.2972**	**0.2616**	**0.2102**	**0.1899**	0.7852
KNN Baseline	0.2689	0.2477	0.2043	0.1812	0.7841
KolMaC CS	0.0911	0.0908	0.0721	0.0569	**0.7863**

Table 4: NDCG@25 for the datasets detailed in Table 2.

6.4.2. Impact of Bribing

Next, we explore the impact of bribing when using Equation (1) as $\mathcal{P}_N^i$, with top-5 and top-25, and when we approximate the value of $\mathcal{P}_N^i(u)$ with the value $\hat{R}_{ui}$.

ML-100k dataset. In Tables 5 and 6, we can see that when item's 1 seller chooses to bribe users that already rated his/her item, in order to increase their ratings, more than half of the users are profitable to bribe when using the SVD recommender system, but much less than half of the users are when using the KolMaC CS system. When the seller bribes users that did not rate his/her product before, in the more realistic scenarios (where $\mathcal{P}_N^1(u)$ is computed using either the top-5 or top-25 as in Equation (1)), most of the users (more than 80%) are not profitable to bribe (row $\pi(\mathfrak{S}_{u1}^1) \leq 0$ and last two columns of the block of non-raters, in Tables 5 and 6). This contrasts with the result obtained using $\hat{R}_{u1}$ to compute $\mathcal{P}_N^1(u)$ (first column of the block of non-raters, in Tables 5 and 6). In

	Raters			Non Raters		
$\mathcal{P}_N^i$	$\hat{R}$	top-5	top-25	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u1}^1) > 0$	81.94	66.67	62.50	85.47	4.65	8.26
$\pi(\mathfrak{S}_{u1}^1) \leq 0$	18.06	33.33	37.50	14.53	95.35	91.74

Table 5: Percentage of users that are profitable and non profitable for item 1 seller to bribe, using different functions as $\mathcal{P}_N^1(u)$, with the SVD recommender system.

	Raters			Non Raters		
$\mathcal{P}_N^i$	$\hat{R}$	top-5	top-25	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u1}^1) > 0$	93.53	17.71	32.56	82.82	6.04	20.78
$\pi(\mathfrak{S}_{u1}^1) \leq 0$	6.47	82.29	67.44	17.18	93.96	79.22

Table 6: Percentage of users that are profitable and non profitable for item 1 seller to bribe, using different functions as $\mathcal{P}_N^1(u)$, with the KolMaC CS recommender system.

	Decrease a Competitor $i = 2$		
$\mathcal{P}_N^i$	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u2}^1) > 0$	19.39	53.06	66.37
$\pi(\mathfrak{S}_{u2}^1) \leq 0$	80.61	46.94	33.67

Table 7: Percentage of users that are profitable and non profitable for item 1 seller to bribe to decrease their ratings to the competitor item 2, using different functions as $\mathcal{P}_N^1(u)$, with the SVD recommender system.

the latter case, using the estimated rating, it follows that more than 80% of the users are profitable to bribe. This discrepancy of results for the different choices of the function $\mathcal{P}_N^1(u)$ has to do with the small increase of estimated ratings for item 1 which are accounted as profit in the $\hat{R}$ scenario, and in the top-N scenarios a small increase is not accounted unless the item starts to appear in the recommendation list or, if the item is already recommended, it changes its order of appearance to a higher one. Another important thing to recall is the observation we made in Section 5.5, which justifies why the percentage of profitable users may be larger in the top-5 than in the top-25 results, as it happens in Table 5.

In Tables 7 and 8, we study the percentage of users that are profitable for a seller to bribe, so that they decrease the rating given to a competitor product, using the SVD and the KolMaC CS recommender systems, respectively. In fact, with a real-world recommender system, the result of Proposition 3 does not hold, and there are users that are profitable to bribe to downgrade a competitors item rating, at least when utilizing the SVD recommender system. However, we notice that by using the $\hat{R}$ to estimate the profit, the percentage of users that are profitable to bribe in the described setting is low, $< 20\%$. Further, the proposed KolMaC CS avoids this case (Table 8), yielding no profitable users to bribe. This experiment goes according to Proposition 3, i.e., it is not profitable for a seller of an item to bribe users to downgrade their ratings for a competitor's item.

In Figure 1, we observe the sum of the profits of strategy $\mathfrak{S}^1$, consisting in bribing the most profitable user at each time. In all the depicted cases, bribing produces a linear increase of wealth. Notice that since one user changing his/her rating may change the estimations of other users, the wealth increase could grow faster than linear and the effect of bribing would be more nefarious to the users' recommendation lists. Further, we can observe that the KolMaC

$\mathcal{P}_N^i$	Decrease a Competitor $i=2$		
	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u2}^1) > 0$	0.00	0.00	0.00
$\pi(\mathfrak{S}_{u2}^1) \leq 0$	100.00	100.00	100.00

Table 8: Percentage of users that are profitable and non profitable for item 1 seller to bribe to decrease their ratings to the competitor item 2, using different functions as $\mathcal{P}_N^1(u)$, with the KolMaC CS recommender system.

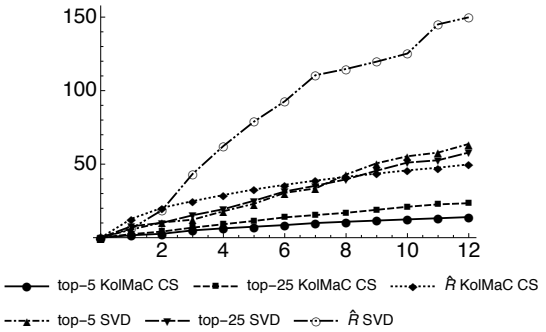

Figure 1: Increase of wealth (sum of the profit) of playing strategy $\mathfrak{S}^1$ using as $\mathcal{P}_N^1(u)$ the top-5, the top-25 and the ratings estimation $\hat{R}$, in the ML-100k dataset, for both SVD and KolMaC CS.

CS recommender system is able to diminish the bribing impact in the ratings estimations. Therefore, it is more resistant to bribery.

ML-1M dataset. Now, we repeat the experiments in the ML-1M dataset. In Tables 9 and 10, and Tables 11 and 12, we show, respectively, the impact of bribing the users to change the ratings of his/her item, and that of a competitor.

For the SVD recommender system, in the case the seller bribes users to rate/increase his/her ratings (Table 9), the estimation with $\hat{R}$ is closer to the results of top-5 and top-25 than in the scenario where the seller bribed to decrease a competitors item ratings (Table 11). This effect happens because a small increase in a rating estimation and decrease in a competitor item may drastically change the top-N recommendation for one user, by removing the

$\mathcal{P}_N^i$	Raters			Non Raters		
	$\hat{R}$	top-5	top-25	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u1}^1) > 0$	54.74	76.84	65.26	12.20	18.13	22.53
$\pi(\mathfrak{S}_{u1}^1) \leq 0$	45.26	23.16	34.74	87.80	81.87	77.47

Table 9: Percentage of users that are profitable and non profitable for item 1 seller to bribe, using different functions as $\mathcal{P}_N^1(u)$, dataset ML-1M, with the SVD recommender system.

	Raters			Non Raters		
$\mathcal{P}_N^i$	$\hat{R}$	top-5	top-25	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u1}^1) > 0$	0.91	7.88	7.72	87.43	0.07	12.84
$\pi(\mathfrak{S}_{u1}^1) \leq 0$	99.09	92.12	92.28	12.57	99.93	87.16

Table 10: Percentage of users that are profitable and non profitable for item 1 seller to bribe, using different functions as $\mathcal{P}_N^1(u)$, dataset ML-1M, with the KolMaC CS recommender system.

competitor's item from the recommendation and pushing up the position of the bribing-seller's item.

Again, we observe that the proposed KolMaC CS system is better to deter the bribing effect, being more difficult to obtain users that are profitable to bribe.

	Decrease a Competitor $i = 2$		
$\mathcal{P}_N^i$	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u2}^1) > 0$	18.00	62.00	62.00
$\pi(\mathfrak{S}_{u2}^1) \leq 0$	82.00	38.00	38.00

Table 11: Percentage of users that are profitable and non profitable for item 1 seller to bribe to decrease their ratings to the competitor item 2, using different functions as $\mathcal{P}_N^1(u)$, with the SVD recommender system.

	Decrease a Competitor $i = 2$		
$\mathcal{P}_N^i$	$\hat{R}$	top-5	top-25
$\pi(\mathfrak{S}_{u2}^1) > 0$	0.00	0.00	0.00
$\pi(\mathfrak{S}_{u2}^1) \leq 0$	100.00	100.00	100.00

Table 12: Percentage of users that are profitable and non profitable for item 1 seller to bribe to decrease their ratings to the competitor item 2, using different functions as $\mathcal{P}_N^1(u)$, with the KolMaC CS recommender system.

Again, we notice that with a real-world recommender system, the result of Proposition 3 does not hold, and there are users which are profitable to bribe to downgrade a competitors item rating. However, we notice once again that by using the $\hat{R}$ to estimate the profit, the percentage of users that are profitable to bribe in the described setting is low ($< 20\%$).

Also, as we observed in Figure 1, in Figure 2 we can notice that strategy $\mathfrak{S}^1$, in the three depicted cases, produces a linear increase of wealth. This corroborates the observation we make that the bribing effect exists and can change users' recommendation lists, but not in a very drastic way.

Further, the KolMaC CS recommender system is, again, more robust to bribing than the SVD system. In fact, as we showed in Proposition 3, it is not

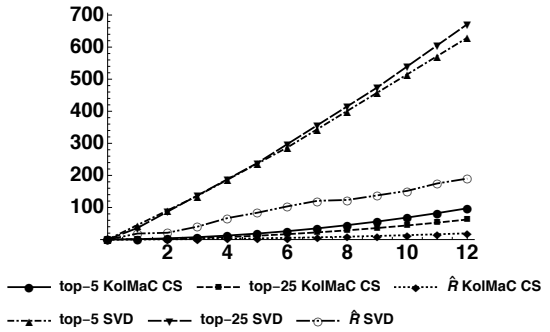

Figure 2: Increase of wealth (sum of the profit) of playing strategy $\mathfrak{S}^1$ using as $\mathcal{P}_N^1(u)$ the top-5, the top-25 and the ratings estimation $\hat{R}$, in the ML-1M dataset, for both SVD and KolMaC CS.

profitable for a seller of an item to bribe users to downgrade their ratings for a competitor's item.

6.5. Discussion

In this section, we discuss our results, to try to infer actionable knowledge on the impact of bribing in recommender systems.

It is important to highlight that, even though the theoretical results were presented on a simplified recommender algorithm to enable us to derive theoretical results about bribery, we can see that the results obtained by both the SVD algorithm and the proposed KolMaC algorithm actually reflect the theoretical ones. This shows that the framework we presented in Section 5 can effectively model the impact of bribing in recommender systems.

The experimental results show that even in the scenario of Proposition 3, which states that it is not profitable for the seller of item i to bribe a user to downgrade the given rating to an item $k \neq i$, this scenario can also be very profitable with more complex recommender systems (such as SVD). In fact, we can notice from the experimental results with SVD algorithm that around half of the users that we tested to bribe generate profit, which points that the effect of bribing can be a very important issue in the area of recommender systems, causing ethical problems and making users question the veracity of recommendations they are presented with.

Moreover, the experimental results point that the proposed recommender system, KolMaC CS, is more robust to bribery than the SVD algorithm. This conclusion is expected, because a small change in a rating pattern of a user (and of an item) does not lead to considerable changes in the compressed versions, whose size is utilized to compute the similarities between users (and between items).

Finally, by showing that bribing is a phenomenon that can effectively affect recommender systems and that it is possible to quantify the impact of bribing strategies, we believe that e-commerce platforms can take concrete action.

Indeed, since bribing is forbidden and platforms are obviously aware of the algorithms that implement their recommender systems, we believe that studying this phenomenon in a similar way as we proposed, would provide a simple and effective way to counter bribing in their platforms, thus ensuring more trustable and unbiased services to their customers.

7. Conclusions and Future Work

In this paper, we considered the impact on recommender systems of bribing users, by persuading them to change the ratings they gave to the items.

We proposed a novel hybrid Collaborative Filtering algorithm that uses a new similarity measure, based on Kolmogorov complexity, aimed at reducing the impact of ratings' change, to counter the effects of bribing. Our algorithm can produce effective recommendations, while coping better with the effect of bribing.

Our theoretical framework allowed us to identify when it is profitable to bribe users to increase/decrease their ratings under several conditions. Both our theoretical results and validation reflect the implications of bribery in recommender systems. Given the strong impact that bribing has on recommender systems and the ethical implications that this type of activity has, a final discussion allowed us to identify actionable points to avoid this phenomenon. Our framework may also be used by e-commerce websites helping them with the choice of the recommender system they will deploy.

As future work, we would like to explore the use of different compressors and evaluate their performance in the proposed compression similarity, and also to explore better approximations of the Kolmogorov complexity, such as the ones available in `http://complexitycalculator.com/`. Furthermore, future research includes modeling the dynamics that emerge when several item's sellers bribe users, for instance as an auction process or by modeling the process with game theory, as a multi-player game.

Acknowledgments

Guilherme Ramos acknowledges the support of Instituto de Telecomunicações (Portugal), through scholarship UID/EEA/50008/2017. This research was done under the scope of Project UID/EEA/50008/2019 of Instituto de Telecomunicações (IT), financed by the applicable framework (FCT/MEC through national funds and cofunded by FEDER-PT2020), and partially supported by the Agència per a la Competivitat de l'Empresa, ACCIÓ, under "AlgoFair" Project. Further, this work was partially funded through the Portuguese Fundao para a Ciência e a Tecnologia (FCT) through Institute for Systems and Robotics (ISR), under Laboratory for Robotics and Engineering Systems (LARSyS) project UID/EEA/50009/2019 and through scholarship BL229/2018_IST-ID.

References

[1] F. Ricci, L. Rokach, B. Shapira, Recommender systems: Introduction and challenges, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 1–34.

[2] J. C. Turner, Social influence, Open University Press Milton Keynes, 1991.

[3] R. B. Cialdini, Influence: Science and practice (5th ed.), Boston: Allyn & Bacon, 2008.

[4] X. Ning, C. Desrosiers, G. Karypis, A comprehensive survey of neighborhood-based recommendation methods, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 37–76.

[5] Y. Koren, R. M. Bell, Advances in collaborative filtering, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 77–118.

[6] A. Jameson, B. Berendt, S. Gabrielli, F. Cena, C. Gena, F. Vernero, K. Reinecke, Choice architecture for human-computer interaction, Found. Trends Hum.-Comput. Interact. 7 (2014) 1–235.

[7] A. Jameson, M. C. Willemsen, A. Felfernig, M. de Gemmis, P. Lops, G. Semeraro, L. Chen, Human decision making and recommender systems, in: F. Ricci, L. Rokach, B. Shapira (Eds.), Recommender Systems Handbook, Springer, 2015, pp. 611–648.

[8] S. Moscovici, G. Mugny, Minority Influence, Springer New York, New York, NY, 1983, pp. 41–64. doi:10.1007/978-1-4612-5578-9_3.
URL https://doi.org/10.1007/978-1-4612-5578-9_3

[9] S. Kambhampati (Ed.), Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016, IJCAI/AAAI Press, 2016.
URL http://www.ijcai.org/Proceedings/2016

[10] J. Saúde, G. Ramos, C. Caleiro, S. Kar, Reputation-based ranking systems and their resistance to bribery, in: V. Raghavan, S. Aluru, G. Karypis, L. Miele, X. Wu (Eds.), 2017 IEEE International Conference on Data Mining, ICDM 2017, New Orleans, LA, USA, November 18-21, 2017, IEEE Computer Society, 2017, pp. 1063–1068.

[11] B. Pan, H. Hembrooke, T. Joachims, L. Lorigo, G. Gay, L. A. Granka, In google we trust: Users' decisions on rank, position, and relevance, J. Computer-Mediated Communication 12 (3) (2007) 801–823.

[12] M. Eslami, K. Karahalios, C. Sandvig, K. Vaccaro, A. Rickman, K. Hamilton, A. Kirlik, First i "like" it, then i hide it: Folk theories of social feeds, in: Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI '16, ACM, New York, NY, USA, 2016, pp. 2371–2382.

[13] A. Springer, V. Hollis, S. Whittaker, Dice in the black box: User experiences with an inscrutable algorithm, 2017.

[14] P. Y. K. Chau, S. Y. Ho, K. K. W. Ho, Y. Yao, Examining the effects of malfunctioning personalized services on online users' distrust and behaviors, Decision Support Systems 56 (2013) 180–191.

[15] R. Burke, M. P. OMahony, N. J. Hurley, Robust collaborative recommendation, in: Recommender systems handbook, Springer, 2015, pp. 961–995.

[16] R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization., in: Nips, Vol. 1, 2007, pp. 2–1.

[17] D. Jannach, G. Adomavicius, Price and profit awareness in recommender systems, CoRR abs/1707.08029. arXiv:1707.08029.
URL http://arxiv.org/abs/1707.08029

[18] A. Azaria, A. Hassidim, S. Kraus, A. Eshkol, O. Weintraub, I. Netanely, Movie recommender system for profit maximization, in: Proceedings of the 7th ACM Conference on Recommender Systems, RecSys '13, ACM, New York, NY, USA, 2013, pp. 121–128.

[19] A. Das, C. Mathieu, D. Ricketts, Maximizing profit using recommender systems, CoRR abs/0908.3633. arXiv:0908.3633.
URL http://arxiv.org/abs/0908.3633

[20] K. Hosanagar, R. Krishnan, L. Ma, Recomended for you: The impact of profit incentives on the relevance of online recommendations, in: Proceedings of the International Conference on Information Systems, ICIS 2008, Paris, France, December 14-17, 2008, Association for Information Systems, 2008, p. 31.

[21] U. Panniello, M. Gorgoglione, S. Hill, K. Hosanagar, Incorporating profit margins into recommender systems: A randomized field experiment of purchasing behavior and consumer trust.

[22] L.-S. Chen, F.-H. Hsu, M.-C. Chen, Y.-C. Hsu, Developing recommender systems with the consideration of product profitability for sellers, Inf. Sci. 178 (4) (2008) 1032–1048.

[23] T. M. Cover, J. A. Thomas, Elements of information theory, John Wiley & Sons, 2012.

[24] M. Li, X. Chen, X. Li, B. Ma, P. M. Vitányi, The similarity metric, IEEE transactions on Information Theory 50 (12) (2004) 3250–3264.

[25] A. R. Cohen, P. M. B. Vitnyi, Normalized compression distance of multisets with applications, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (8) (2015) 1602–1614.

[26] P. M. Vit, et al., Compression-based similarity, in: Data Compression, Communications and Processing (CCP), 2011 First International Conference on, IEEE, 2011, pp. 111–118.

[27] J. S. Resende, P. R. Sousa, R. Martins, L. Antunes, Breaking mpc implementations through compression, International Journal of Information Security.

[28] A. Bellogn, P. Snchez, Collaborative filtering based on subsequence matching: A new approach, Information Sciences 418-419 (2017) 432 – 446.

[29] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2008, pp. 426–434.

[30] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: The adaptive web, Springer, 2007, pp. 291–324.

[31] Y. Koren, Factor in the neighbors: Scalable and accurate collaborative filtering, ACM Transactions on Knowledge Discovery from Data (TKDD) 4 (1) (2010) 1.

[32] N. S. Altman, An introduction to kernel and nearest-neighbor nonparametric regression, The American Statistician 46 (3) (1992) 175–185.

[33] M. Herlihy, N. Shavit, The art of multiprocessor programming, Morgan Kaufmann, 2011.

[34] G. Linden, B. Smith, J. York, Amazon.com recommendations: Item-to-item collaborative filtering, IEEE Internet Computing 7 (1) (2003) 76–80.

[35] B. Smith, G. Linden, Two decades of recommender systems at amazon.com, IEEE Internet Computing 21 (3) (2017) 12–18.

[36] Y. Wang, L. Wang, Y. Li, D. He, W. Chen, T.-Y. Liu, A theoretical analysis of ndcg ranking measures, in: Proceedings of the 26th Annual Conference on Learning Theory (COLT 2013), Vol. 8, 2013.

Appendix A. Space and Time Evaluation

In this appendix, we will report the amount of RAM memory and the number of seconds needed to run each approach.

Method	ML–100k	ML–1M	ML–10M	ML–20M	Jester
Normal Predictor	0.12Gb	0.77Gb	17.18Gb	35.4Gb	1.07GB
Baseline Only	0.12Gb	0.80Gb	12.19Gb	24.3Gb	1.08Gb
SVD	0.13Gb	0.79Gb	13.06Gb	25.15Gb	1.09Gb
SVD++	0.13Gb	0.88Gb	13.76Gb	26.02Gb	1.09Gb
NMF	0.13Gb	0.80Gb	10.08Gb	21.21Gb	1.09Gb
Slope One	0.19Gb	1.07Gb	11.71Gb	40.32Gb	2.13Gb
Co-clustering	0.13Gb	0.80Gb	10.08Gb	21.21Gb	0.87GB
KNN Basic	0.15Gb	1.46Gb	37.64Gb	88.21Gb	53.75Gb
KNN With Means	0.15Gb	1.46Gb	37.66Gb	88.34Gb	53.76Gb
KNN Baseline	0.15Gb	1.47Gb	37.91Gb	88.90Gb	53.87Gb
KolMaC CS	**0.11Gb**	**0.72Gb**	**2.43Gb**	**4.17Gb**	**0.32Gb**

Table A.13: Memory (RAM) used for the datasets detailed in Table 2.

Method	ML–100k	ML–1M	ML–10M	ML–20M	Jester
Normal Predictor	1s	5s	**50s**	**120s**	**9s**
Baseline Only	1s	6s	94s	228s	10s
SVD	10s	95s	689s	1395s	169s
SVD++	375s	7074s	51042	103254	2645s
NMF	9s	78s	698s	1418s	136s
Slope One	2s	33s	389s	936s	14s
Co-clustering	3s	22s	302s	682s	37s
KNN Basic	2s	92s	2578s	6011s	1678s
KNN With Means	2s	90s	2531s	6585s	1691s
KNN Baseline	2s	92s	2608s	6020s	1717s
KolMaC CS	**1s**	**4s**	87s	215s	**9s**

Table A.14: Execution time to generate the predictions to one user for the datasets detailed in Table 2.

Space evaluation. Table A.13 reports the amount of RAM memory needed to run each of the approaches. As highlighted in the previous set of experiments, as the size of the data increases, the classic memory-based algorithms take incredible amounts of RAM to run, confirming their lack of scalability. Our approach, despite being a memory-based one, requires a very low amount of memory. As the results show, our algorithm outperforms all the baselines by a large margin.

Time evaluation. Although the programming languages of the proposed algorithm and the baselines are not the same, which implies that a direct comparison of the computation times may suffer from a constant penalization factor, our method is one of the fastest to estimate rating prediction (see Table A.14). Indeed, results show that our algorithm is either the one that takes the lowest time to compute the results or, when the size of data increases, our results are slightly above the Normal Predictor algorithm that, despite being very fast, is much less accurate than ours (as Table 3 shows).

Appendix B.

Proof of Proposition 1. The initial wealth, using (2) and (6), is

$$\mathcal{J}(i) = \sum_{u \notin U_i} \left(\frac{\alpha}{|U_i|} \sum_{v \in U_i} R_{vi} + \frac{1-\alpha}{|I_u|} \sum_{j \in I_u} R_{uj} \right),$$

and, after playing strategy $\mathfrak{S}^i$, the final wealth is

$$\begin{aligned} \mathcal{J}(\mathfrak{S}^i) = \sum_{u \notin U_i} \left[\frac{\alpha}{|U_i|} \left(\sum_{v \in U_i} R_{vi} + \mathfrak{S}^i_{wi} \right) + \frac{1-\alpha}{|I_u|} \sum_{j \in I_u} R_{uj} \right] \\ - |\mathfrak{S}^i_{wi}|. \end{aligned}$$

Therefore, the profit that the seller of item i obtains by playing strategy $\mathfrak{S}^i$ is

$$\begin{aligned} \pi(\mathfrak{S}^i) &= \mathcal{J}(\mathfrak{S}^i) - \mathcal{J}(i) \\ &= \sum_{u \notin U_i} \frac{\alpha}{|U_i|} \mathfrak{S}^i_{wi} - |\mathfrak{S}^i_{wi}| \\ &= \mathfrak{S}^i_{wi} \left(\alpha \frac{|U| - |U_i|}{|U_i|} - sgn(\mathfrak{S}^i_{wi}) \right) \end{aligned}$$

Thus, if $sgn(\mathfrak{S}^i_{wi}) = 1$, the strategy is profitable *if and only if* $\pi(\mathfrak{S}^i) > 0$, which is equivalent to $|U_i| < \frac{\alpha}{1+\alpha}|U|$. If $sgn(\mathfrak{S}^i_{wi}) = -1$, then we need to have that $\alpha \frac{|U \setminus U_i|}{|U_i|} < -1$ to have that $\pi(\mathfrak{S}^i) > 0$, which is impossible. □

Proof of Proposition 2. The profit that the seller of item i obtains by playing strategy $\mathfrak{S}^i$ is

$$\begin{aligned} \pi(\mathfrak{S}^i) &= \mathcal{J}(\mathfrak{S}^i) - \mathcal{J}(i) \\ &= \sum_{\substack{u \neq w \\ u \notin U_i}} \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi} + \mathfrak{S}^i_{wi}}{|U_i| + 1} + (1-\alpha) \frac{\sum_{j \in I_u} R_{uj}}{|I_u|} \right] \\ &- |\mathfrak{S}^i_{wi}| - \sum_{\substack{u \neq w \\ u \notin U_i}} \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi}}{|U_i|} + (1-\alpha) \frac{\sum_{j \in I_u} R_{uj}}{|I_u|} \right] \\ &- \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi}}{|U_i|} + (1-\alpha) \frac{\sum_{j \in I_w} R_{wj}}{|I_w|} \right] \\ &= \mathfrak{S}^i_{wi} \frac{|U| - |U_i|}{|U_i|} \alpha - \left[\alpha \frac{\sum_{\substack{u \neq w \\ u \notin U_i}} R_{vi}}{|U_i|(1 + |U_i|)} + |\mathfrak{S}^i_{wi}| + \hat{R}_{wi} \right]. \end{aligned}$$

When $\mathfrak{S}^i_{wi} > 0$, we have that $\pi(\mathfrak{S}^i) > 0$ if

$$\mathfrak{S}^i_{wi}\frac{|U|-|U_i|}{|U_i|}\alpha > \left[\frac{\alpha}{|U_i|(1+|U_i|)}\sum_{\substack{u\neq w\\ u\notin U_i}} R_{vi} + |\mathfrak{S}^i_{wi}| + \hat{R}_{wi}\right].$$

□

Proof of Proposition 3. The profit that the seller of item i obtain playing strategy $\mathfrak{S}^i$ is

$$\begin{aligned}
\pi(\mathfrak{S}^i) &= \mathcal{J}(\mathfrak{S}^i) - \mathcal{J}(i)\\
&= \sum_{\substack{u\neq w\\ u\notin U_i}}\left[\alpha\frac{\sum_{\substack{v\neq w\\ v\in U_i}} R_{vi} + \mathfrak{S}^i_{wk}}{|U_i|} + (1-\alpha)\frac{\sum_{j\in I_u} R_{uj}}{|I_u|}\right]\\
&- |\mathfrak{S}^i_{wk}| - \sum_{\substack{u\neq w\\ u\notin U_i}}\left[\alpha\frac{\sum_{\substack{v\neq w\\ v\in U_i}} R_{vi}}{|U_i|} + (1-\alpha)\frac{\sum_{j\in I_u} R_{uj}}{|I_u|}\right]\\
&= \mathfrak{S}^i_{wk}\left(\alpha\frac{|U|-|U_i|}{|U_i|} - sgn(\mathfrak{S}^i_{wk})\right).
\end{aligned}$$

Hence, the bribing strategy is not profitable if $\mathfrak{S}^i_{wk} < 0$ and, otherwise, it is profitable if $(|U|>2) \wedge \left(\alpha > \frac{1}{|U|-1}\right) \wedge \left(|U_i| < \frac{\alpha}{\alpha+1}|U|\right)$. □

Proof of Proposition 4. The profit that the seller of item i obtain playing strategy $\mathfrak{S}^i$ is given by

$$
\begin{aligned}
\pi(\mathfrak{S}^i) &= \mathcal{J}(\mathfrak{S}^i) - \mathcal{J}(i) \\
&= \sum_{\substack{u \neq w \\ u \notin U_i}} \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi}}{|U_i|} + (1-\alpha) \frac{\sum_{j \in I_u} R_{uj}}{|I_u|} \right] - |\mathfrak{S}^i_{wi}| \\
&+ \left[\alpha \frac{\sum_{\substack{u \neq w \\ u \notin U_i}} R_{vi}}{|U_i|} + (1-\alpha) \frac{\left(\sum_{w \in I_u} R_{uj} + \mathfrak{S}^i_{wk}\right)}{|I_w|} \right] \\
&- \sum_{u \notin U_i} \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi} + \mathfrak{S}^i_{wi}}{|U_i|} + (1-\alpha) \frac{\sum_{j \in I_u} R_{uj}}{|I_u|} \right] \\
&= \frac{1-\alpha}{|I_w|} \mathfrak{S}^i_{wk} - |\mathfrak{S}^i_{wk}| \\
&= \mathfrak{S}^i_{wk} \left(\frac{1-\alpha}{|I_w|} - sgn(\mathfrak{S}^i_{wk}) \right).
\end{aligned}
$$

Therefore, the strategy is profitable *if and only if* $1-\alpha > sgn(\mathfrak{S}^i_{wk})|I_w|$. Since $|I_w| \geq 1$, this is equivalent to $\mathfrak{S}^i_{wk} < 0$. □

Proof of Proposition 5. The profit that the seller of item i obtains by playing strategy $\mathfrak{S}^i$ is

$$
\begin{aligned}
\pi(\mathfrak{S}^i) &= \mathcal{J}(\mathfrak{S}^i) - \mathcal{J}(i) \\
&= \sum_{\substack{u \neq w \\ u \notin U_i}} \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi}}{|U_i|} + (1-\alpha) \frac{\sum_{j \in I_u} R_{uj}}{|I_u|} \right] - \mathfrak{S}^i_{wi} \\
&+ \left[\alpha \frac{\sum_{\substack{u \neq w \\ u \notin U_i}} R_{vi}}{|U_i|} + (1-\alpha) \frac{\left(\sum_{w \in I_u} R_{uj} + \mathfrak{S}^i_{wk}\right)}{|I_w|+1} \right] \\
&- \sum_{u \notin U_i} \left[\alpha \frac{\sum_{\substack{v \neq w \\ v \in U_i}} R_{vi} + \mathfrak{S}^i_{wi}}{|U_i|} + (1-\alpha) \frac{\sum_{j \in I_u} R_{uj}}{|I_u|} \right] \\
&= -(1-\alpha) \frac{\sum_{j \in I_w} R_{wj}}{|I_w|(1+|I_w|)} + \mathfrak{S}^i_{wk} \left(\frac{1-\alpha}{|I_w|} - 1 \right).
\end{aligned}
$$

If $\alpha = 1$ then it is not profitable, otherwise, it is profitable if

$$
\mathfrak{S}^i_{wk} > \frac{|I_w|}{|I_w|^2+1} \sum_{j \in I_w} R_{wj}.
$$

□