Neurocomputing 246 (2017) 12–24

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Multi-Objective Ranked Bandits for Recommender Systems

Anisio Lacerda

Centro Federal de Educação Tecnológica de Minas Gerais – CEFET-MG, Av. Amazonas, Belo Horizonte 7675, Minas Gerais, Brazil

ARTICLE INFO

Article history:
Received 16 March 2016
Revised 24 September 2016
Accepted 3 December 2016
Available online 10 February 2017

Keywords:
Recommender systems
Online recommendation
Multi-armed bandits

ABSTRACT

This paper is interested in recommender systems that work with implicit feedback in *dynamic* scenarios providing *online* recommendations, such as news articles and ads recommendation in Web portals. In these dynamic scenarios, user feedback to the system is given through clicks, and feedback needs to be quickly exploited to improve subsequent recommendations. In this scenario, we propose an algorithm named multi-objective ranked bandits, which in contrast with current methods in the literature, is able to recommend lists of items that are accurate, diverse and novel. The algorithm relies on four main components: a scalarization function, a set of recommendation quality metrics, a dynamic prioritization scheme for weighting these metrics and a base multi-armed bandit algorithm. Results show that our algorithm provides improvements of 7.8 and 10.4% in click-through rate in two real-world large-scale datasets when compared to the single-objective state-of-the-art algorithm.

1. Introduction

In a world of information overload, having systems capable of filtering out content according to users preferences is paramount. In this scenario, recommender systems play an important role in recommending relevant items to users. The recommendations generated by these systems may account for explicit feedback information, such as the rating a user gives to a movie or a book, or implicit feedback, e. g., whether a user clicks an ad in a portal or listens to a song in a music streaming service.

This paper is interested in recommender systems that work with implicit feedback in *dynamic* scenarios providing *online* recommendations. One example within this scenario is to recommend news articles in a Web portal, illustrated in Fig. 1, where content changes dynamically, with frequent insertions and deletions of news.

In dynamic scenarios such as the aforementioned, user feedback to the system (given through clicks) needs to be quickly exploited to improve subsequent recommendations. The most popular methods to deal with the problem of online interactive recommender systems follow a multi-armed bandit (MAB) approach, where an algorithm makes a sequence of choices (each choice represented by an arm) according to the algorithm policy. In each trial the algorithm chooses from a set of alternatives and receives a reward associated with its choice. When using MABs for recommendation, each arm is modeled as an item to be recommended to the user. At each trial, one item is recommended (one arm is chosen) and the algorithm receives as feedback information of whether the user clicked to see the recommended item.

Traditionally, MAB algorithms are limited to recommending one item per trial to the user. However, there are scenarios where a list of items is expected instead of a single item. In this direction, *Ranked Bandits* [26,31] were proposed as an alternative approach to recommend item lists within the context of Web search engines. In contrast with the traditional MAB approach, this method associates a full MAB algorithm to each position of the ranking of retrieved items. Hence, the user receives a list of items, where the item in each position of the ranking was recommended by a different MAB.

Ranked Bandits, as well as other classical MAB approaches, focus mostly on improving recommendation accuracy, which is usually measured by click-through rate. However, it has become a consensus in the literature that successful recommender systems need to consider other dimensions of information utility besides accuracy [8]. Notably, diversity and novelty of the suggestions performed by the system are key objectives to measure recommendation quality [27,28]. In other words, even being accurate, obvious and monotonous recommendations are generally of little use, since they do not expose users to unprecedented experiences.

In this direction, in [17] we introduced the idea of building upon Ranked bandits to account for the optimization of multiple recommendation quality metrics, namely accuracy, diversity and, novelty of the ranking. We transformed this multi-objective optimization problem into a single-objective optimization by using scalarization functions to weight different objectives to predict the probability of a user clicking in a given recommended item.

E-mail addresses: anisiom@gmail.com, anisio@dcc.ufmg.br

http://dx.doi.org/10.1016/j.neucom.2016.12.076
0925-2312/© 2017 Elsevier B.V. All rights reserved.

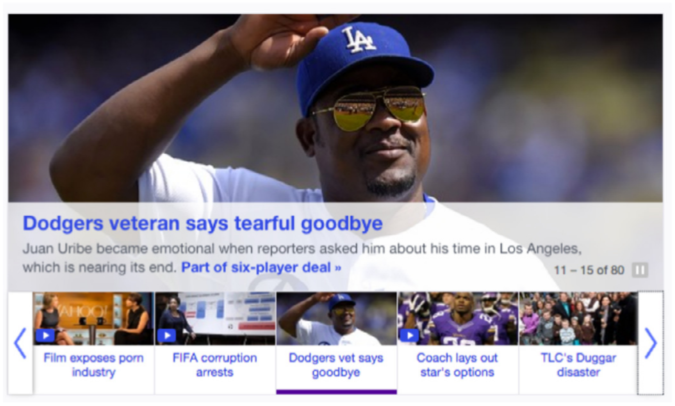

Dodgers veteran says tearful goodbye
Juan Uribe became emotional when reporters asked him about his time in Los Angeles, which is nearing its end. Part of six-player deal »
11 – 15 of 80
Film exposes porn industry
FIFA corruption arrests
Dodgers vet says goodbye
Coach lays out star's options
TLC's Duggar disaster

Fig. 1. Example of news articles recommendation in the "Main" tab of the Yahoo! Front Page. The highlighted recommended item refers to a sports article [17].

This paper goes further, and besides testing non-linear scalarization schemes, it proposes a new method that dynamically prioritizes different recommendation quality metrics during the life cycle of the user in the system. This is done by optimizing weights for different metrics online, as the user interacts with the system, while accounting for his feedback. This is important because the method adapts the systems responses to the user needs (e.g., new system users may benefit from more accurate suggestions, whereas older users may require more diversified and novel recommendations).

We conducted an extensive evaluation of the proposed method in two large datasets from real-world recommendation tasks. The first task, namely news article recommendation, uses data from the Yahoo! Today News dataset and contains over 45,811,883 user visits. The second task deals with online advertisement (ad) recommendation, and uses data from the KDD Cup 2012 Online Ads (track 2) dataset. The dataset contains 149,639,105 views of advertisements. Our algorithm provides improvements of 7.8 and 10.4% in click-through rate for Yahoo! Today News and KDD Cup 2012 Online Ads data, respectively, when compared to the state-of-the-art single-objective algorithm.

The main contributions of this paper are:

- We propose and test two scalarization methods for weighting accuracy, diversity and novelty of the produced rankings in the context of ranked bandits.
- We propose a method that dynamically changes the weight of multiple recommendation quality metrics under three different perspectives: the user, the system and the ranking position.
- We conducted experiments in two real-world datasets to evaluate the performance of our method and a representative set of bandit algorithms.

The remainder of this paper is organized as follows. In Section 2, we review related approaches in the literature. In Section 3, we present the formulation of the online interactive recommendation problem. We then detail our method to predict arm reward considering multiple objectives in Section 4. In Section 5, we analyze the performance of our method comparing with traditional bandit algorithms proposed in the literature. Finally, Section 6 draws conclusions and discusses extensions of this work.

2. Related work

There is an extensive literature of methods conceived to perform personalized offline and online recommendation. Currently, the state-of-the-art methods for offline recommendation follow a Collaborative Filtering (CF) approach [29]. CF approaches recommend an item i to a user u in two situations: (i) if the item is liked by other users whose preferences are similar to those of u (user-based CF), or (ii) if the item is similar to other items user u liked in the past (item-based CF). Since they rely on the historical ratings or preferences provided by the users, their performance is poor in online recommendation tasks.

Instead of going over the literature of offline recommendation methods exhaustively, we refer the interested reader to a recent comprehensive survey of recommender systems [5]. In this section, we focus on the studies that are most relevant to our approach, i.e., multi-objective recommender systems.

In the early days, the main drive of the recommender systems field was to accurately predict users' preferences. Nowadays, however, a wider perspective towards recommendation utility has been taking place, and the importance of diversity and novelty, among other properties, is becoming an important part of evaluation practice [11,36].

Multi-objective recommender systems have been successfully applied in several scenarios, such as, movies [27], tags [15], and advertisements [18]. For example, in [41] the authors concluded that user satisfaction does not always correlate with high recommendation accuracy. Hence, different multi-objective algorithms have been proposed to enhance user experience considering more than the accuracy of suggestions. In [40], the authors present a method to improve recommendation diversity, which is based on an optimization procedure that considers two objective functions reflecting the preference for similarity and diversity of recommended items.

In this same direction, in [28], the authors propose hybrid approaches to multi-objective recommender systems based on the concept of Pareto-efficiency. Their method aggregates the output lists of basic recommender systems into a single list. In [43], the authors propose a genetic programming algorithm to simultaneously consider recommendation accuracy and diversity of suggested items.

We highlight that all the aforementioned methods run over an offline setting, i.e., the recommendation models are produced before user interactions with the system. On the other hand, our approach focuses in a online setting by considering users' feedback to improve recommendation quality. As far as we know, this is the first work that presents an online multi-objective recommendation algorithm that dynamically prioritize accuracy, diversity, and novelty when suggesting items to users.

When dealing with online scenarios, approaches based on multi-objective multi-armed bandit algorithms (MO-MAB) deserve our attention. In MO-MAB, several arms (items) are candidates to optimal depending on the set of objectives [9,38,39]. The standard approach to MO-MABs consists of modifying classical multi-armed bandit algorithms to consider more than one objective. For instance, in [9] the authors extend the classical UCB1 algorithm to propose the Pareto-UCB1, its multi-objective counterpart.

The current research line in MO-MAB algorithms investigates multi-objective versions of classical algorithms, which keep selecting a single arm (item) per interaction. Differently, we are interested in suggesting multiple items to users considering several objectives, which is equivalent to selecting many arms simultaneously, one from each MAB associated with a ranking position. Given the different nature of the results of the recommendation (a single versus a list of items), we consider unfair to compare MO-MAB algorithms and the methods proposed here. Note that, even if we consider the use of MO-MAB algorithms for recommending a single item per iteration to users, we restrict the possible objectives that can be considered. For instance, the diversity of a recommendation list is a metric that makes no sense if a single item is returned for each recommendation.

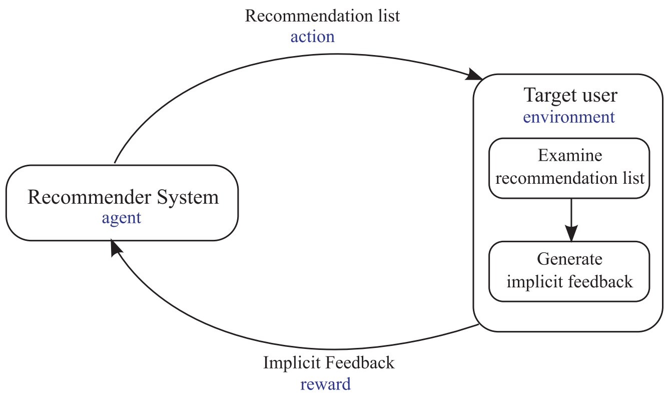


Fig. 2. The online recommendation task modeled as a multi-objective ranked bandit problem.

As we are interested in suggesting multiple items to users considering several objectives, in this work we follow the approach presented in [26,31], named *Ranked Bandits*. *Ranked Bandits* is an online learning algorithm that learns a ranking of items based on users' behavior [26]. The authors in [31] build upon [26] and present variants of ranked bandits to consider side information on similarity between items returned by search engines. Both works have as their main goal to minimize query abandonment, which happens when the user does not click in any of the presented query results.

There are two crucial differences between the works aforementioned and the one proposed here. First, they are interested only on minimizing the query abandonment of search engine rankings. Hence, they ignore other important metrics of these rankings, such as diversity and novelty. Second, since they ignore the user features, they cannot provide personalized recommendations, which is a crucial characteristic of our problem.

3. Problem formulation

Online recommendation is usually modeled as a multi-armed bandit (MAB) problem, which originates from the reinforcement learning paradigm. The MAB problem is defined as a sequential Markov decision process (MDP) of an agent that tries to optimize its actions while improving its knowledge on the arms. The fundamental challenge in bandit problems is the need for balancing exploration and exploitation. Specifically, the algorithm $\mathcal{A}$ *exploits* past experience to select the arm that seems best. On the other hand, this seemingly optimal arm may be suboptimal, due to imprecisions in $\mathcal{A}$'s knowledge. To avoid this undesired situation, $\mathcal{A}$ has to *explore* by choosing seemingly suboptimal arms so as to gather more information about them.

In the context of online recommender systems, each arm represents an item to be recommended to a user, and the reward corresponds to whether the user clicked the suggested item or not. In Fig. 2, we show the interaction cycle between a target user and an online recommender system. The user arrives and the recommender system produces an ordered list of items, which is presented to the user. The user interacts with the list by clicking on items that are relevant to him. The interaction is interpreted as a feedback about the quality of the suggested item list. This feedback is then used to update the recommendation model aiming to provide better recommendations in the future. The cycle finishes and the next user is considered. This formulation translates to the Reinforcement Learning problem (see the terminology in blue color in Fig. 2).

Formally, let U be an arbitrary input space and $I = \{1, \ldots, M\}$ be a set of M arms, where each arm corresponds to an item. Let E be a distribution over tuples $(u, \vec{r})$, where $u \in U$ corresponds to the target user, i.e., the user we are interested in recommending items to, and $\vec{r} \in \{0, 1\}^M$ is a vector of rewards. For each pair $(u, \vec{r})$, the j^{th} component of $\vec{r}$ is the reward associated to arm i_j. The target user u can be described using features such as demographic information and gender.

Algorithm 1 Multi-armed bandits.

```
1: for $t = 1, \ldots, T$ do
2:     $i(t) \leftarrow$ select-arm$(u_t, \mathcal{A})$
3:     show $i(t)$ to user; register click
4:     if user clicked $i(t)$ then
5:         $r_t = 1$
6:     else
7:         $r_t = 0$
8:     end if
9:     update-mab$(u_t, i(t), r_t, \mathcal{A})$
10: end for
```

In Algorithm 1, we detail a MAB algorithm. A MAB algorithm iterates in discrete trials $t = 1, 2, \ldots,$ (Line 1). In each trial t, the algorithm chooses an item $i(t) \in I$ based on user u_t and on the knowledge it collected from the previous trials (Line 2), and shows the chosen item to the target user (Line 3). After that, the reward r_t is revealed (Line 4–8).[1] The algorithm then improves its arm-selection strategy with the new observation (u_t, $i(t)$, r_t) (Line 9). Note that the algorithm selects a single item per trial, i.e., it is unable to return a recommendation list of items.

Also observe that the reward is revealed only for the chosen item. When the suggested item is clicked, the reward is 1; otherwise, the reward is 0. The expected reward of an item is the percentage of clicks it receives considering all trials, a.k.a the *Click-Through Rate* (CTR). Hence, the algorithm needs to choose the item with maximum CTR, which maximizes the expected number of clicks from users.

Recently, works in the multi-armed bandit literature have investigated methods for learning a ranking of arms instead of a single arm per trial [26], generating the *Ranked Bandits*. In Algorithm 2, we present the ranked bandits algorithm, which focuses on optimizing the choice of multiple arms. It runs an MAB instance $\mathcal{A}_i$ for *each ranking position i*. We consider that users scroll the ranking from top to bottom. Hence, algorithm $\mathcal{A}_1$ chooses which item is suggested at rank 1. Next, algorithm $\mathcal{A}_2$ selects which item is shown at ranking position 2, unless the item was already selected at a higher ranking position. If that is the case, the second item is chosen arbitrarily. This procedure is repeated sequentially, for all top k ranking positions. If a user clicks on a item in ranking position i, then the algorithm $\mathcal{A}_i$ receives a reward of 1, and all higher (i.e., skipped) algorithms $\mathcal{A}_j$, where $j < i$, receive a reward of 0. For algorithms $\mathcal{A}_j$ with $j > i$, the state is rolled back as if this trial had never happened (as if the user never considered these items). If no item is clicked, then all algorithms $\mathcal{A}_*$ receive a reward of 0.

Different from the previously mentioned ranked bandits algorithm, we address the problem of learning an optimal *multi-objective* ranking of items $\mathcal{D} = \{d_1, d_2, \ldots, d_k\}$ for a target user. Specifically, we consider the multi-objective ranked bandit problem, where we have k ranking positions, $k \geq 1$; M items, $M \geq 2$; and O objectives, $O \geq 1$. The goal is to simultaneously consider all objectives when recommending items to users. Note that each user may consider a subset of items as relevant according to their preferences and past experience, and the remainder of the items as non-relevant. Hence, users with different preferences will have

[1] We use r_t as the reward for item $i(t)$ in trial t.

Algorithm 2 Single-objective ranked bandits.

```
Require: bandits $\mathcal{A}$
 1: Initialize $\mathcal{A}_1(M), \ldots, \mathcal{A}_k(M)$                ▷ Initialize MABs
 2: for $t = 1, \ldots, T$ do
 3:     for $j = 1, \ldots, k$ do                                ▷ Sequentially choose items
 4:         $\hat{\mathrm{i}}_j(t) \leftarrow$ select-arm$(u_t, \mathcal{A}_j)$
 5:         if $j > 1$ then
 6:             if $\hat{\mathrm{i}}_j(t) \in \{\mathrm{i}_1(t), \ldots, \mathrm{i}_{j-1}(t)\}$ then
 7:                 $\mathrm{i}_j(t) \leftarrow$ arbitrary unselected item
 8:             else
 9:                 $\mathrm{i}_j(t) \leftarrow \hat{\mathrm{i}}_j(t)$
10:             end if
11:         end if
12:     end for
13:     show $\{\mathrm{i}_1(t), \ldots, \mathrm{i}_k(t)\}$ to user; register clicks
14:     for $j = 1, \ldots, k$ do                                ▷ Find feedback for $\mathcal{A}_j$
15:         if user clicked $\mathrm{i}_j(t)$ and $\hat{\mathrm{i}}_j(t) = \mathrm{i}_j(t)$ then
16:             $r_{jt} = 1$
17:         else
18:             $r_{jt} = 0$
19:         end if
20:         update-mab$(u_t, \hat{\mathrm{i}}_j(t), r_{jt}, \mathcal{A}_j)$
21:     end for
22: end for
```

different relevant rankings, while users with similar preferences will have similar relevant rankings.

4. Multi-objective ranked bandits

In this section, we detail the multi-objective algorithm proposed to iteratively suggest items to users considering multiple quality recommendation metrics. We focus on learning an optimally relevant ranking of items for a target user, and model the recommendation task as a *ranked bandit* problem, as described in Section 3.

Before going into detail on the proposed algorithm, we illustrate the rationale behind it using the example in Table 1. In this example, we want to recommend 3 items ($k = 3$) to a user u considering three quality recommendation metrics: accuracy, diversity and novelty (the definition of these metrics is given in Eqs. (5) and (7). The multi-objective ranked bandit works as follows. For each position of the ranking of recommended items, a MAB $\mathcal{A}_k$ is instantiated. Each arm of $\mathcal{A}_k$ represents an item to be recommended. For each item, a vector $\vec{s}$ with its accuracy, diversity and novelty is stored. The accuracy corresponds to the predicted reward obtained by $\mathcal{A}_k$. Note that each independent $\mathcal{A}_k$ is associated with a weight vector $\vec{w}$, which will assign different weights to different objectives (quality metrics) at different positions of the ranking. In this way, more diverse and novel items may be privileged into higher positions, while accuracy may be considered more important into lower ranks. Although the example illustrated in Table 1 associates the weights to ranking positions, other weighting strategies will be discussed later in this section.

At each trial, $\mathcal{A}_1$ returns an item to u by combining the weighted objectives using a function (which in this example is linear). $\mathcal{A}_1$ returns i_3, while a single-objective version baseline (column SO) accounting only for accuracy would return i_2. Then, $\mathcal{A}_2$ updates the diversity metric for all items in relation to the current ranking $\mathcal{D}$. Next, $\mathcal{A}_2$ recommends an item to u by combining the weighted objectives, returning item i_1. Since the novelty metric depends only on the selected item and not on the entire list, it is updated in the next trial. These same steps are followed to recommend the remaining items. At the end of this process, we have a ranking that was generated using a set of objectives simultaneously. If we compare the final rankings $\mathcal{D}_{MO-RANK} = \{i_3, i_1, i_2\}$ and $\mathcal{D}_{SO} = \{i_2, i_3, i_1\}$ in Table 1, we can notice that the positions of items change according to the values and weights given to different quality metrics.

The process we just described is composed by four main components: (i) a scalarization function, (ii) a set of recommendation quality metrics, (iii) a weighting scheme, and (iv) a base MAB algorithm $\mathcal{A}$.

The scalarization function defines how we weight each objective to compute a relevance score for each item. In Section 5.4.2, we detail two scalarization functions considered in this paper. The recommendation quality metrics define the objectives that we are interested in maximizing on recommendation rankings, as described in Section 4.2. As we assume that different scenarios require different objectives' prioritization, we propose different weighting schemes to set these weights, as discussed in Section 4.3. Finally, in Section 4.4, we detail the set of base MAB algorithms used in this work.

4.1. Scalarization functions

A traditional way to deal with the multi-objective scenario is to transform it into a single-objective scenario using *scalarization functions* [9]. Here we consider both linear and non-linear scalarization functions.

As previously explained, in the multi-objective scenario, for each arm (item) i, there is a score vector $\vec{s_i} = (s_i^1, \ldots, s_i^O)$, where O is a fixed number of objectives. We use a scalarization function to compute the estimation of rewards for all items, In other words, we compute a combined score using the scalarization function to predict the reward vector $\vec{r} \in \{0, 1\}^M$, for all M items. The

Table 1
In this toy example, we compare the final rankings of a single objective (SO) and a multi-objective (MO-RANK) recommender.

Rank	MAB instance	Weights	Items	Accuracy	Diversity	Novelty	Final ranking	
							SO	MO-RANK
1	$\mathcal{A}_1$	(0.3, 0.2, 0.5)	i_1	0.2	1.0	0.2	$\mathbf{i_2}$	$\mathbf{i_3}$
			i_2	0.5	1.0	0.4		
			i_3	0.3	1.0	0.6		
2	$\mathcal{A}_2$	(0.4, 0.5, 0.1)	i_1	0.7	0.5	0.2	$\mathbf{i_3}$	$\mathbf{i_1}$
			i_2	0.1	0.4	0.4		
			i_3	0.9	0.1	0.2		
3	$\mathcal{A}_3$	(0.6, 0.1, 0.3)	i_1	0.8	0.2	0.1	$\mathbf{i_1}$	$\mathbf{i_2}$
			i_2	0.1	0.7	0.4		
			i_3	0.1	0.1	0.2		

two scalarization functions used here receive the weighted sum of the components of the score vector $\vec{s}$ and return a scalar value [10].

We consider that the score vectors of all arms are ordered using the partial order on multi-objective spaces [42]. A score vector s is considered better than, or *dominates*, another score vector v, $s \prec v$, if and only if there exists at least one objective j for which $v^j < s^j$, and for all other objectives l, we have $v^l \leq s^l$. In contrast, s is *non-dominated* by v, $v \nprec s$, if and only if there exists at least one dimension j for which $v^j < s^j$. The *Pareto optimal score set* (or *Pareto set* for short) is the set of score vectors that are non-dominated by any other score vectors [9]. Following, we detail the scalarization functions used in this work.

Linear scalarization This function takes as input the vector of weights $\vec{w} = (w^1, \ldots, w^O)$, where $\sum_{o=1}^{O} w^o = 1$, assigns to each component s_i^o of the score vector $\vec{s_i}$ of an arm i a weight w^o, and returns a scalar equals to the weighted sum of the means:

$$f_{\vec{w}}(\vec{s_i}) = w^1 s_i^1 + \cdots + w^O s_i^O. \tag{1}$$

A known drawback with linear scalarization is its incapacity to potentially find all the points in a non-convex Pareto set [9].

Chebyshev scalarization: this function takes the weight vector as the above function and also a O-dimensional reference point $z = (z^1, \ldots, z^O)^T$, which has to be dominated by all elements in the Pareto set, i.e., all optimal reward vectors s_i^*. The Chebyshev scalarization function is defined as:

$$f_{\vec{w}}(\vec{s_i}) = \min_{1 \leq j \leq O} w^i \times (s_j^i - z^i), \ \forall j. \tag{2}$$

The vector z corresponds to the minimum of the current optimal rewards minus a small positive value,[2] $\tau > 0$. Hence, for each objective l, we have:

$$z^l = \min_{1 \leq j \leq O} s_j^l - \tau^l, \ \forall j. \tag{3}$$

A key advantage of the Chebyshev scalarization function is that under certain conditions it encounters all points in a non-convex Pareto set [39].

Now, given that we are able to convert a multi-objective MAB problem into a single-objective problem, we can select the best arm $i^*_{f_{\mathbf{w}}}$ that maximizes the function $f_{\mathbf{w}}$:

$$i^*_{f_{\vec{w}}} = \operatorname*{argmax}_{1 \leq i \leq M} f_{\vec{w}}(\vec{s_i}). \tag{4}$$

4.2. Recommendation quality metrics

In this section, we detail the recommendation quality metrics used in this work, namely, (i) accuracy, (ii) diversity, and (iii) novelty. Besides accuracy, which has been the typical goal of recommender systems, suggesting items that are not easily discovered by the users is essential, and it can be measured by the diversity and novelty of the recommendations [28].

We interpret the estimated reward computed by base MAB algorithms for all set of arms as an estimation of accuracy (see details in Section 5.3). Next, we detail how we compute diversity and novelty scores for each candidate item from our item set. Note that we consider normalized scores for all metrics (i.e., accuracy, diversity, and novelty).

In order to measure the diversity of an item i_j, we adapt the distance-based model from [36] to consider the set of items above ranking position j. Specifically, we consider all items with ranking position l, such that $1 \leq l < j$, and compare them to the current item i_j. The diversity of item i_j is given by:

$$\operatorname{div}(i_j|S) = \frac{1}{|S|} \sum_{1 \leq l < j} d(i_j, i_l), \tag{5}$$

where S is the set of items in ranking position l smaller than j; and $d(a, b)$ is the cosine similarity between items a and b. When S is empty, i.e., we are considering the first position in the ranking, we initialize all item diversities with equal values, e.g., 1.

In order to measure the novelty of item i_j, in turn we compute a popularity-based item novelty metric, as presented in [36]. Let the probability of an item i_j being seen as:

$$p(seen|i_j) = \frac{plays_l(i_j)}{\sum_{m=1}^{M} plays_l(i_m)}, \tag{6}$$

where $plays_l(x)$ is the number of times $\mathcal{A}_l$ instance chooses item x.

The novelty of item i_j chosen by $\mathcal{A}_l$ is given by:

$$\operatorname{nov}(i_j) = 1 - p(seen|i_j). \tag{7}$$

Summarizing, diversity is related to the internal differences within a ranking, whereas novelty can be understood as the difference between present and past experiences [36].

4.3. Weighting schemes

In Section 4.2, we have discussed the importance of considering a wider perspective towards the added value of recommendation, going beyond prediction accuracy. Here, we focus on different weighting schemes that can dynamically adapt to the needs of the user, the system, or even to minimize the inherent biases of recommendation.

Next, we detail the proposed methods for weighting each recommendation quality metric, i.e., accuracy, diversity and novelty. Since the motivations for enhancing each metric are themselves diverse, we propose three different weighting schemes by considering user, system, and ranking position perspectives. In Algorithm 3,

Algorithm 3 Multi-objective arm selection.

```
Require: weight vector $\vec{w}$, score of objectives matrix $S_{M \times O}$, scalarization function $f$
1: function SELECT-ARM
2:     for $i = 1, \ldots, M$ do
3:         final-scores$_i = f_{\vec{w}}(\vec{s_i})$    ▷ Note: $\vec{s_i}$ is the objectives scores for item $i$
4:     end for
5:     best-arm = argmax(final-$\vec{s}$cores)
6:     return best-arm
7: end function
```

we detail the algorithm used to select the best arm (i.e., best item) to recommend to users. Note that the only difference among the proposed weighting schemes (i.e., **per User, per System**, and **per Rank**) is that we need to instantiate a different weight vector $\vec{w}$ depending on the chosen scheme.

Dynamic multi-objective weighting per user (DMO-USER): from the user perspective, both novelty and diversity are desirable and act as a direct source of user satisfaction. They help users to expand their horizon and, as a consequence, enrich their experience. Besides that, according to [24], new users have different needs from experienced users in a recommender system. Since new users need to establish trust and rapport with a recommender, they may benefit from more accurate suggestions, whereas older users may require more novel and diversified suggestions. However, how accurate, diverse and novel the recommendations are should depend on the user experience with the system

[2] We used $\tau = 0.001$ in our experiments.

and his tastes. We propose to have a *distinct weight vector for each user*, which will be adapted according to the feedback he provides. Looking at the example in Table 1, instead of associating weights with ranking positions, we would have the same weights for all positions, but they would vary from one user to another. We emphasize that, in this weighting scheme, we have a different weight vector $\vec{w}$ for each user.

Dynamic multi-Objective weighting per system (DMO-SYSTEM): from the system perspective, there is a great extent of uncertainty on what the actual user preferences really are, given the limitation of the user actions. For instance, in this paper we focus on implicit feedback, such as clicks on items. Although clicks are driven by user interests, it is still difficult to identify what motivates a user to click on an item. As the recommender has access to a limited number of user activities, it has an incomplete knowledge about the users. In order to alleviate this limitation, a healthy level of diversity and novelty may help the system to reach a better understanding of the users and, as a consequence, improve the quality of recommendations. Another motivation behind providing more diverse and novel suggestions is related to business' motivations. For instance, suggesting items in the long tail is a strategy to draw profit from market niches by selling less of more [2]. Moreover, product diversification is a way to expand business and mitigate risk [22,35]. In this case, *a global weighting vector* is used, and the weights are the same for all users and all positions of the ranking. In this weighting scheme, we have a global weight vector $\vec{w}$ common for all users and ranking positions.

Dynamic multi-objective weighting per rank (DMO-RANK): we also consider an inherent bias involved when users choose items from ordered lists, i.e., the positional bias. This bias states that items in the top positions of the ranking are more likely to receive clicks than items at lower positions [14]. This bias has been investigated in the Information Retrieval [12,13] and Recommender Systems [14,19] literature. Here, we propose a weighting scheme that dynamically assigns different weights for accuracy, diversity and novelty to *different ranking positions* (see Table 1). Given that higher ranking positions receive more attention from users, a valid strategy is to diversify recommendations on these positions for new items, to understand how attractive they are to users. Meanwhile, if users reach lower positions of the ranking, the recommender may prioritize accuracy on these positions to benefit users' effort. In this weighting scheme, we have a different weight vector $\vec{w}$ for each ranking position.

For all of aforementioned weighting schemes, we want to dynamically learn the values of weights using the users' feedback. Hence, we model the online inference of the weights as a state-space model [25]. A generic form of the model is:

$$\vec{z}_t = g(\vec{c}_t, \vec{z}_{t-1}, \nu_t), \tag{8}$$

$$\vec{f}_t = h(\vec{z}_t, \vec{c}_t, \delta_t), \tag{9}$$

where $\vec{z}_t$ is the hidden state, $\vec{c}_t$ is an optional input, ν_t is the system noise at time *t*, δ_t is the observation noise at time *t*, $\vec{f}_t$ is the observation, *g* is the transition model and *h* is the observation model.

We define the hidden state to represent the weights, i.e., $\vec{z}_t \equiv \vec{w}_t$, and let the time-varying observation model represent the current score vector $\vec{s}$. Hence, we apply the Kalman filter to the observation model to update our posterior beliefs about the weights $\vec{w}$ as the recommender interacts with the users. Finally, the update of the model parameters is given by:

$$\vec{w}_t = \vec{w}_{t-1} + \frac{1}{\sigma^2}\,\Sigma_{t|t}(r_t - \vec{s}_t^{\,T}\vec{w}_{t-1})\vec{s}_t, \tag{10}$$

where r_t is the reward associated to the suggested item at time *t* and σ^2 is the standard deviation. We compute the weighting values using a weighting regressor $\mathcal{W}$, which uses Eq. (10) to update weights according to users' feedback. We detailed this procedure in terms of input and output in Algorithm 4.

Algorithm 4 Update regressor method.

```
Require: weight vector $\vec{w}$, reward $r$, and score of objectives vector $\vec{s}$
 1: function UPDATE-REGRESSOR
 2:     ▷ Compute new weights for each objective (as a computation over input vectors)
 3:     $\vec{w}_{new} = \vec{w} + \frac{1}{\sigma^2}\,\Sigma(r - \vec{s}^{\,T}\vec{w})\vec{s}$
 4:     return $\vec{w}_{new}$
 5: end function
```

Algorithm 5 Multi-objective ranked bandits.

```
Require: andits $\mathcal{A}$, scalarization function $f$, andweighting regressor $\mathcal{W}$
 1: Initialize $\mathcal{A}_1(M), \ldots, \mathcal{A}_k(M)$                ▷ Initialize MABs
 2: for $t = 1, \ldots, T$ do
 3:     for $j = 1, \ldots, k$ do                          ▷ Sequentially choose items
 4:         $\vec{w}_t$ = get-weights($\mathcal{W}$)
 5:         $\hat{\imath}_j(t) \leftarrow$ select-arm($u_t, \vec{w}_t, f, \mathcal{A}_j$)          ▷ Eq.~4
 6:         if $j > 1$ then
 7:             if $\hat{\imath}_j(t) \in \{\mathrm{i}_1(t), \ldots, \mathrm{i}_{j-1}(t)\}$ then
 8:                 $\mathrm{i}_j(t) \leftarrow$ arbitrary unselected item
 9:             else
10:                 $\mathrm{i}_j(t) \leftarrow \hat{\imath}_j(t)$
11:             end if
12:         end if
13:     end for
14:     show $\{\mathrm{i}_1(t), \ldots, \mathrm{i}_k(t)\}$ to user; register clicks
15:     for $j = 1, \ldots, k$ do                          ▷ Find feedback for $\mathcal{A}_j$
16:         if user clicked $\mathrm{i}_j(t)$ and $\hat{\imath}_j(t) = \mathrm{i}_j(t)$ then
17:             $r_{jt} = 1$
18:         else
19:             $r_{jt} = 0$
20:         end if
21:         update-mab($u_t, \hat{\imath}_j(t), r_{jt}, \mathcal{A}_j$)
22:         $\vec{w}_t$ = update-regressor($\mathcal{W}, \vec{w}_t, r_{jt}$)          ▷ Alg.~4
23:     end for
24: end for
```

4.4. Algorithm formalization

Algorithm 5 presents the proposed multi-objective ranked bandit algorithm, which receives as input a set of bandit algorithms $\mathcal{A}$, a scalarization function $\mathcal{S}$, and a weighting regressor $\mathcal{W}$. The scalarization function refers to how we weight each objective to compute a relevance score for each item. We also assume that different scenarios require different objectives' prioritization. A weighting regressor is used to dynamically update the weights for each recommendation quality metric.

The individual bandit algorithms are initialized to properly consider the set of items *I*, where $|I| = M$ (Line 1). At each trial *t*, the algorithm interacts with target user u_t (Line 2). For each ranking position *j*, the base bandit algorithm instance chooses an item from a fixed item set according to the current weight vector $\vec{w}_t$ (Lines 3–5). Recall that for each rank *j* there is a separate instance bandit algorithm $\mathcal{A}_j$ which is responsible for recommending an item for the *j*th position.

If the selected item has already been chosen on a higher rank, we select a new item arbitrarily (Lines 6–10). The chosen items are displayed to the user and a click (or non-click) is registered

(Line 12). After the target user scrolls the recommended ranking, the algorithm processes his feedback for each jth ranking position (Lines 13–20).

We consider only binary feedback, i.e., the reward is 1 if the user clicked on the suggested item, and 0 otherwise (Lines 14–18). Hence, we update each multi-armed bandit instance A_j that corresponds to the jth ranking position according to the given reward (Line 19). Finally, we update the online linear regressor $\mathcal{W}$ to consider the given reward and corresponding weighting scheme (Line 20). Note that, the main differences from Algorithm 2 refer to lines 4–5 and 20. In these lines, we consider the computed weights and update them, respectively.

5. Experimental results

In this section, we investigate the performance of our method in two real-world datasets related to news article recommendation (*Yahoo! Today News*) and online advertisement recommendation (*KDD Cup 2012 Online Ads*) (Section 5.1). Next, we describe the evaluation protocol (Section 5.2) and several multi-armed bandit algorithms used to instantiate our algorithm (Section 5.3). Finally, in Section 5.4 we discuss experimental results.

We emphasize that the datasets used in our experiments are publicly available on the Web [1], and that the code for the method and baselines can be find in the Github repository.[3]

5.1. Datasets

We used two real-world, large-scale datasets to evaluate our algorithm: Yahoo! Today News and KDD Cup 2012 anline advertising.

Yahoo! Today News

Personalized news recommendation is the problem of displaying relevant news articles for target users on web pages based on the prediction of their individual tastes. The recommender system uses feedback from users to build a prediction model of user's tastes. However, there is a distinct feature in this application: its "partial-label" nature, in which the user feedback (click or not) for an article is observed *only when* this article is showed.

We evaluate our method on clickstream data made available by Yahoo! as part of the *Yahoo! Webscope program* [1]. We use the R6A version of the dataset, which contains more than 45 million user view/click log events for articles displayed on the *Yahoo! Today Module* of *Yahoo! Front Page* during a period of ten contiguous days. We used 4.5 million events as a warm-up to bandit algorithms (10%), and th e remaining log events as test data.

Each user/system interaction event consists of four components: (i) the random article displayed to the user, (ii) user/item feature vectors, (iii) whether the user clicked on the displayed article, and (iv) the timestamp. In this dataset, displayed articles were randomly selected from an article pool. This makes this dataset ideal for unbiased, offline evaluation of multi-armed bandit algorithms. Detailed information on the data collection process and the dataset properties can be found in [21].

KDD Cup 2012 online advertising

Online advertising systems present relevant advertisements (ads) to target users in order to maximize the click-through rate (see Eq. (11)) on recommended ads. Sponsored search is a well-known example of online advertising, where the search engine selects the most relevant ads based on the user profile and a set of search keywords.

The system needs to be able to select new ads to users and learn from their feedback to improve the overall CTR estimation.

We use an advertisement dataset published by KDD Cup 2012, track 2. This dataset refers to a click log on www.soso.com (a large-scale search engine serviced by Tencent), which contains 149 million ad impressions (view of advertisements). Each instance is an ad impression that consists of: (i) the user profile (i.e., gender and age range), (ii) search keywords, (iii) displayed ad information, and (iv) the click count. We processed the data as follows. The dimension of the vector of features of the displayed ad information is 1,070,866. We first conducted a dimensionality reduction using the Latent Dirichlet Allocation topic model [4] using the keywords, title and description of ads. We used 6 topics to represent ads. Another issue related to this dataset is that click information is extremely sparse due to the large number of ads. To tackle this problem, we selected the top 50 ads that have the most impressions in the evaluation, as proposed by the authors in [32]. The generated data set contains 9 million user visit events, from which 0.9 million events were used as warm up to bandit algorithms (10%) and the remaining events as test data. Note that the competition originally refers to a batch scenario. Hence, it would be unfair to consider the competition winner solution [37] as a baseline, given that here we investigate an online learning recommendation problem.

5.2. Evaluation protocol

In an ideal scenario, the evaluation of a new multi-armed bandit algorithm should be conducted using a bucket test, where we run the algorithm to serve a fraction of live user traffic in a real recommender system. This evaluation protocol has several drawbacks. First, not only it is an expensive method, since it requires substantial engineering efforts for deployment in the real system, but it may also have negative impact on user experience. Moreover, there is no guarantee of replicable comparison using bucket tests as online metrics vary significantly over time [21].

On the other hand, evaluating online learning algorithms with past data is a challenging task. Here, we use the state-of-the-art method for unbiased and offline evaluation of bandit algorithms, namely *BRED* (*Bootstrapped Replay on Expanded Data*) [23]. The method is based on bootstrapping techniques and enables an unbiased evaluation by utilizing historical data. The use of bootstrapping allows to estimate the distribution of predictions while reducing their variability.

The BRED method assumes that the set of available items in the dataset is static to perform the bootstrap resamples. We follow the same evaluation protocol presented in [23], and assume a static world on small portions of the Yahoo! front page dataset. We took the smallest number of portions such that a given portion has a fixed number of items, resulting in 433 portions. For the KDD Cup 2012 online ads data, since we selected the top 50 most clicked ads, we have only a single portion, which contains just the 50 mentioned ads.

Let us consider each portion as a single dataset D of size T with M possible choices (items). Then, from D, at each recommendation step, we generate P new datasets $D_1, D_2, \ldots, D_P$ of size $M \times T$ by sampling with replacement.

For each dataset i of size D_i with M_i items, we compute the estimated click-through rate (CTR) by averaging the CTR of each MAB on 100 random permutations of the data. We are considering a recommendation task of top-k items, i.e., we recommend, for each target user, a list of the most relevant items with size k. Hence, we also evaluate the performance of our algorithm for a variable list size. The averaged reward is equivalent to the metric CTR, which is the total reward divided by the total number of trials:

$$CTR@k = \frac{1}{T}\sum_{t=1}^{T} r_t, \qquad (11)$$

[3] www.github.com/to-be-announced (available after publication).

where r_t is the sum of rewards on a ranking with size k, and T is the total number of trials. We report the algorithm *relative CTR*, which is the algorithm's CTR divided by the random recommender.

5.3. Base bandit algorithms

The proposed multi-objective ranked Bandit algorithm runs a MAB instance $\mathcal{A}_i$ for *each ranking position i*. Each of the k copies of the multi-armed bandit algorithm is independent from each other. Recall that the scores computed by these algorithms are used to estimate the accuracy term in Eq. (4).

MAB algorithms are faced with the challenge of sequentially balancing exploitation and exploration. This is because while it is important to maximize rewards (*exploitation*), it is also crucial to better understand the options to explore uncertainties in the representation of the world (*exploration*). Different algorithms work in different ways to find the best trade-off between these two objectives. For background, we refer the reader to [7] and a recent survey on regret minimization[4] bandits [6]. Following, we detail the base MAB algorithms used to instantiate our algorithm.

- Random: it randomly chooses an arm to pull.
- ϵ-greedy (ϵ): it chooses a random arm with probability ϵ, and chooses the arm with the highest CTR estimate with probability $(1-\epsilon)$ [34].
- UCB1: it chooses the arm with the highest UCB (*Upper Confidence Value*) score, according to [3].
- UCB2: the same as the above, but with a different UCB score computation [3].
- Thompson sampling (TS): it is a Bayesian MAB algorithm where the (unknown) reward values are inferred from past data and summarized using a posterior distribution. It chooses the arm proportionally to its probability of being optimal under this posterior [33].
- Bayes UCB: it is an adaption of UCB to the Bayesian setting, where the upper confidence bound score is given by an upper quantile on the posterior mean [16].
- LinUCB (α): is a classical contextual bandit algorithm that computes the expected reward of each arm by finding a linear combination of the previous rewards of the arm [20]. The contextual aspect of this algorithm refers to its capability of considering user features, such as gender and age, to recommend items. The α parameter is used to control the balance of exploration and exploitation.

Note that we are unable to use multi-objective multi-armed bandit algorithms (MO-MAB) as base algorithms (Section 2). First, MO-MAB algorithms consider the selection of a single item for each trial, and we are interested in returning a ranking of items to users. Even if we consider MO-MAB algorithms for the case where $k=1,$ it is not simple to understand the behavior of these algorithms as base MABs. The challenge is that MO-MAB algorithms try to consider multiple objectives by construction, and so does our multi-objective ranked bandit algorithm. In this case, it turns out to be really difficult to separate the contributions of each multi-objective strategy (MO-MAB and MO ranked bandits) in the final results if we combine them. Investigating modifications in the multi-objective ranked bandits algorithms to consider MO-MABs is an interesting line of research.

5.4. Experimental results

In this section, we empirically analyze the performance of our multi-objective algorithm with different policies to define the priorities of different quality metrics. The results are compared with the state-of-the-art method of the literature for single-objective optimization, referred in this section as SO (ranked bandits with optimized accuracy).

We compare static multi-objective prioritization schemes, presented in [17], with the dynamic schemes introduced in Section 4, namely DMO-USER, DMO-RANK, DMO-SYSTEM. In the static scheme, we fix the priority of each quality metric of interest beforehand using a weight vector $w=(w_a, w_d, w_n),$ where we set fixed weights for the priority of accuracy (w_a), diversity (w_d), and novelty (w_n). We evaluate the following four priority schemes:

- SMO-EQ [$w=(0.30, 0.30, 0.30)$]: establishes equal priorities to each objective.
- SMO-ACC [$w=(0.50, 0.25, 0.25)$]: prioritizes accuracy over diversity and novelty.
- SMO-DIV [$w=(0.25, 0.50, 0.25)$]: prioritizes diversity over accuracy and novelty.
- SMO-NOV [$w=(0.25, 0.25, 0.50)$]: prioritizes novelty over accuracy and diversity.

5.4.1. On overall CTR

In Tables 2 and 3, we present the results for Yahoo! and KDD datasets, respectively. We test the algorithms with different values of CTR@k, where $k \in \{1, 3, 5, 10\}$ is the number of recommendations displayed (and, consequently, the number of MABs instantiated). Since we have similar conclusions for all values, we present only CTR@5 in this section. Sections 5.4.2 and 5.4.3 analyze the impact of parameter k on the methods' performance. Note that for base bandit algorithms ϵ-greedy (EG) and LinUCB, we enumerate different values for parameters ϵ and α, respectively. We highlight gains in relative CTR@5 over the Single-Objective baseline (SO) in **bold**.

We start by comparing the overall CTR values for the linear and Chebyshev scalarization functions. The results for Chebyshev scalarization are better than linear scalarization on both datasets. Considering the overall relative CTR, we have gains of 9.3 and 11.2% for linear and Chebyshev scalarization functions, respectively. We believe this happens because, under certain conditions, the Chebyshev scalarization function is able to find all the points in a non-convex Pareto set [9], which makes it able to find a better compromise between different objectives (metrics), recommending more relevant items to users. Note that, for the KDD dataset, the best overall improvement in terms of CTR happens for the combinations of the Chebyshev function with the Thompson sampling and dynamic multi-objective weighting scheme. When considering the Yahoo! dataset, the best results were obtained using the combination of the Chebyshev function with the ϵ-greed algorithm ($\alpha=0.10$) and dynamic multi-objective weighting scheme.

Considering the overall values of CTR obtained when using static or dynamic prioritization of recommendation quality metrics (accuracy, diversity, and novelty), for both datasets the best CTR performance happens when using dynamic weighting. Specifically, for the Yahoo! dataset, we reach an 8.0% overall CTR improvement of dynamic weighting over static weighting schemes, whereas in the KDD dataset the improvement is of 2.5%.

For the Yahoo! dataset, the best performance of static weighting scheme is achieved when priorityzing novelty over other recommendation metrics (i.e., SMO-NOV). For the KDD dataset, in turn, the prioritization of diversity (i.e., SMO-DIV) reaches better performance in terms of relative CTR. Since the weights are static, the recommendation of diverse and novel rankings is more likely to find relevant items than just prioritizing accuracy, for instance.

Considering dynamic weighting schemes, in both datasets the best scheme is to consider a specific prioritization for each ranking position (DMO-RANK). When we consider the dynamic

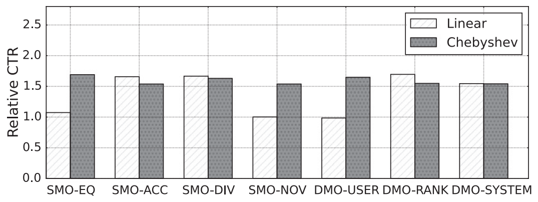
[4] Note that minimizing the regret is equivalent to maximizing the rewards.

Table 2
Relative CTR@5 on Yahoo! Today News data.

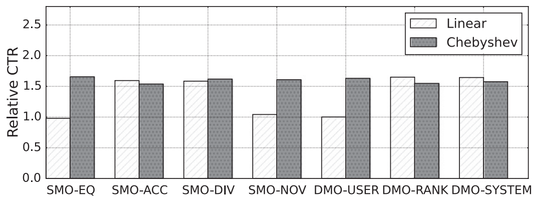
Linear scalarization								
	Weighting scheme							
		Static				Dynamic		
Algorithm	SO	SMO-EQ	SMO-ACC	SMO-DIV	SMO-NOV	DMO-USER	DMO-RANK	DMO-SYSTEM
Random	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
EG(0.01)	1.5291	**1.5336**	1.4660	1.5197	1.4348	1.5009	**1.5984**	1.4885
EG(0.05)	1.5565	1.5538	1.5227	1.4957	**1.5565**	1.4840	**1.5800**	1.5114
EG(0.10)	1.5443	1.5125	1.5314	1.5085	1.5190	1.5025	1.5287	1.5163
EG(0.20)	1.5060	**1.5403**	**1.5825**	1.4812	1.4870	**1.5197**	**1.5422**	**1.5307**
EG(0.30)	1.4948	**1.5132**	**1.5204**	1.4854	**1.5643**	1.4848	**1.5942**	**1.5045**
LinUCB(0.01)	1.5379	1.5148	1.4833	1.5276	1.5235	1.4875	**1.5619**	1.5294
LinUCB(0.10)	1.5271	1.5264	1.5006	1.4675	1.5243	1.4844	**1.5322**	1.5170
LinUCB(0.30)	1.5471	1.5358	1.4692	1.4752	1.5005	1.4583	1.5138	1.5146
TS	1.5581	1.4661	1.4568	1.5082	1.5191	1.4825	**1.6269**	**1.6112**
UCB1	1.4880	1.4144	1.3887	1.4309	1.4347	1.4306	**1.5811**	**1.5381**
UCB2	1.5187	**1.5525**	**1.5390**	**1.5581**	**1.5365**	1.4991	**1.6052**	**1.6198**
BayesUCB	1.5339	1.4411	1.4091	1.4366	1.4377	1.4264	**1.6224**	**1.5915**
Chebyshev scalarization								
Random	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
EG(0.01)	1.5291	**1.5919**	**1.5506**	**1.6141**	**1.6503**	**1.6179**	1.5091	1.4887
EG(0.05)	1.5565	**1.6485**	1.5126	**1.5966**	**1.6522**	**1.6227**	1.5296	1.5121
EG(0.10)	1.5443	**1.6365**	**1.5708**	**1.5910**	**1.6644**	**1.6265**	**1.5535**	1.5082
EG(0.20)	1.5060	**1.6305**	**1.5116**	**1.5828**	**1.6386**	**1.5921**	**1.5404**	**1.5476**
EG(0.30)	1.4948	**1.5842**	1.4906	**1.5917**	**1.6450**	**1.5722**	**1.5617**	1.4600
LinUCB(0.01)	1.5379	1.5288	1.4676	1.4379	1.5265	1.5356	1.5058	1.5099
LinUCB(0.10)	1.5271	**1.5347**	1.4837	1.4856	**1.5371**	1.5201	**1.5527**	**1.5517**
LinUCB(0.30)	1.5471	1.5199	1.5011	1.4770	1.5181	1.5338	1.5357	1.5322
TS	1.5581	1.5463	1.5230	**1.5607**	**1.5920**	**1.5900**	**1.5714**	1.5169
UCB1	1.4880	**1.5192**	1.4669	**1.5661**	**1.5221**	**1.5030**	1.4782	1.4760
UCB2	1.5187	1.4970	1.4933	**1.5469**	**1.5814**	**1.5540**	1.4490	1.4542
BayesUCB	1.5339	**1.5574**	1.5233	**1.5405**	1.5306	**1.5651**	**1.5430**	1.4832

Table 3
Relative CTR@5 on KDD Cup 2012 online ads data.

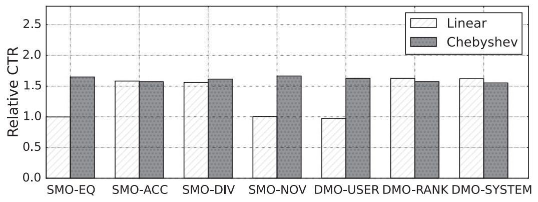
Linear scalarization								
	Weighting scheme							
		Static				Dynamic		
Algorithm	SO	SMO-EQ	SMO-ACC	SMO-DIV	SMO-NOV	DMO-USER	DMO-RANK	DMO-SYSTEM
Random	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
EG(0.01)	1.7504	1.6444	1.4809	1.5681	1.6351	1.5406	**1.8756**	1.5867
EG(0.05)	1.6702	1.6297	**1.8049**	1.5532	**1.7044**	1.6264	**1.7112**	1.5364
EG(0.10)	1.5525	**1.7390**	1.5511	**1.7796**	**1.6323**	1.4538	**1.7901**	1.4300
EG(0.20)	1.5997	**1.7662**	1.5939	1.4301	**1.7022**	1.4578	**1.7498**	**1.7998**
EG(0.30)	1.8229	1.5608	1.6831	1.4801	1.4996	1.7413	1.6511	1.6796
LinUCB(0.01)	1.8719	1.6399	1.5800	1.5747	1.5612	1.4865	1.6069	1.5834
LinUCB(0.10)	1.7580	1.6798	1.5675	1.5340	1.7559	1.6096	1.6919	1.4677
LinUCB(0.30)	1.9600	1.4623	1.5262	1.5713	1.4506	1.5624	1.6790	1.6760
TS	1.8472	1.4284	1.7169	1.6447	1.5593	1.4457	**1.9494**	**1.9081**
UCB1	1.6841	1.4890	1.4037	1.3775	1.4787	1.4941	**1.8884**	1.6600
UCB2	1.5925	1.5065	**1.8330**	**1.6817**	1.5109	**1.6799**	**1.9039**	**1.7978**
BayesUCB	1.7012	1.4583	1.4412	1.6279	1.4945	1.5782	1.6030	1.6006
Chebyshev scalarization								
Random	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
EG(0.01)	1.7504	**1.8296**	1.6705	**1.7794**	**2.0273**	**1.7734**	**1.9141**	1.7002
EG(0.05)	1.6702	**1.8499**	**1.9400**	**1.7724**	**1.9430**	**1.9100**	**1.7773**	1.6401
EG(0.10)	1.5525	**2.0298**	**1.6972**	**2.0300**	**1.9175**	**1.6726**	**1.9660**	1.5086
EG(0.20)	1.5997	**2.0166**	**1.6233**	**1.6239**	**2.0228**	**1.6221**	**1.8832**	**1.9673**
EG(0.30)	1.8229	1.7457	1.7740	1.6896	1.6747	**1.9929**	1.7285	1.7535
LinUCB(0.01)	1.8719	1.7756	1.6747	1.5834	1.6702	1.6350	1.6547	1.6706
LinUCB(0.10)	1.7580	**1.8146**	1.6576	1.6609	**1.9105**	**1.7686**	**1.8427**	1.5955
LinUCB(0.30)	1.9600	1.5360	1.6668	1.6854	1.5595	1.7612	1.8312	1.8226
TS	1.8472	1.6020	**1.9409**	1.8270	1.7450	1.6489	**2.0395**	**1.9435**
UCB1	1.6841	**1.7116**	1.5813	1.6017	1.6759	1.6788	**1.9117**	**1.7085**
UCB2	1.5925	1.5441	**1.9254**	**1.7908**	**1.6548**	**1.8742**	**1.8597**	**1.7355**
BayesUCB	1.7012	1.6809	1.6633	**1.8807**	1.7010	**1.8614**	1.6229	1.5902



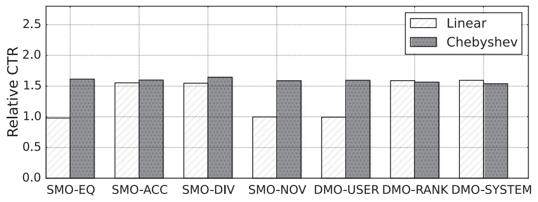
(a) $k = 1$



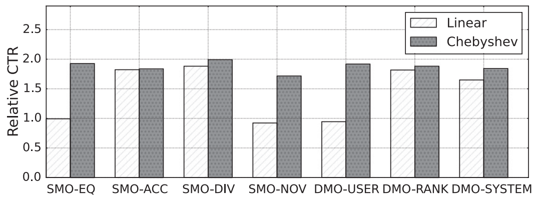
(b) $k = 3$



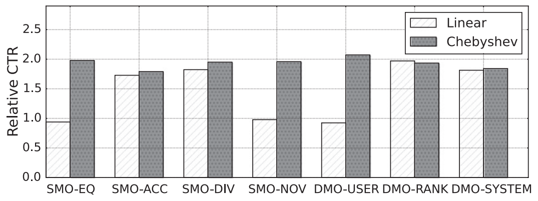
(c) $k = 5$



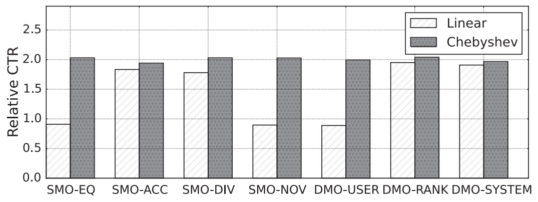
(d) $k = 10$



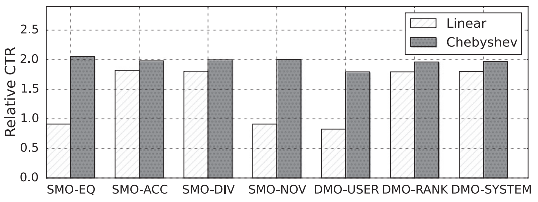
(e) $k = 1$



(f) $k = 3$



(g) $k = 5$



(h) $k = 10$

Fig. 3. Relative CTR's on multi-objective weighting schemes for different ranking lengths, (i.e., different k values). Fig. (a– d) for Yahoo! Today News data and Fig. (e– h) KDD Cup 2012 online ads data.

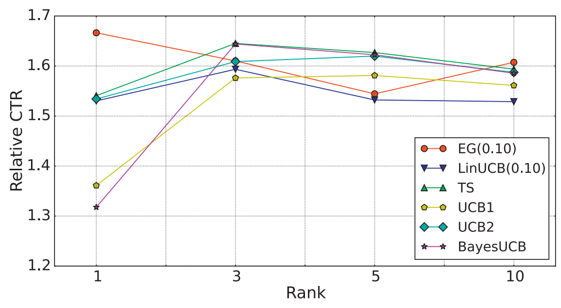
multi-objective weighting per user (DMO-USER), we are able to improve baseline performance mostly considering Chebyshev scalarization. It seems hard to find good recommendation metric weighting schemes when considering a linear relationship between these metrics. Furthermore, the dynamic multi-objective weighting per system (DMO-SYSTEM) presents the worst performance when compared to other weighting schemes. Since it considers a common prioritization scheme for all users and ranking positions, we believe that it is harder for it to correctly prioritize the recommendation metrics.

Also observe that, for all base algorithms, there is a combination of weighting scheme and scalarization function that achieves a better CTR performance than the state-of-the-art single-objective baseline. The best performance for the Yahoo! dataset is obtained by the ϵ-greedy algorithm (EG0.10), with a CTR improvement of 7.8%. For the KDD dataset, the best performance is obtained by the Thompson sampling algorithm, with gains of 10.4% over the baseline in terms of CTR. On average the gains over the single-objective algorithm are 4.7% and 15.8%, for the Yahoo! and KDD datasets, respectively.

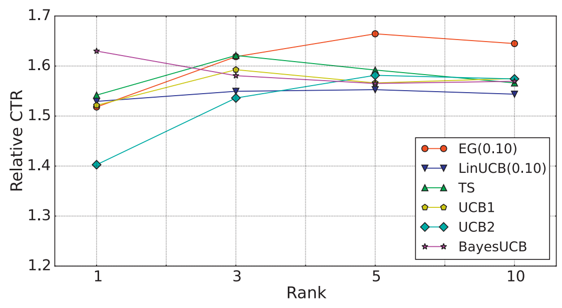
Furthermore, for the Yahoo! dataset, we observe that the best single-object base algorithm is the Thompson sampling. However, this pattern changes as we start considering diversity and novelty, which leads ϵ-greedy to achieve the best CTR. We have a similar pattern in the KDD dataset, where the best single-objective base algorithm is LinUCB, whereas the multi-objective base algorithm is the Thompson sampling. Once we have gains in both situations, we may conclude that considering multiple objectives simultaneously is important to provide better recommendations to users.

Moreover, the choice of weighting prioritization scheme and scalarization functions to provide a better performance are dataset dependent. For instance, for the KDD dataset, the multi-objective versions of LinUCB and Bayes UCB exhibit no gains over single-objective baseline using linear scalarization. However, when considering Chebyshev scalarization, the performance achieves gains of up to 8.7% for the same algorithms (i.e., LinUCB and Bayes UCB).

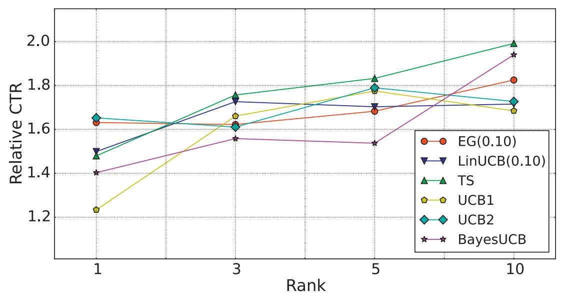
In conclusion, we present gains in CTR performance over baselines when considering multiple objectives simultaneously. Second, we show improvements using dynamic weighting schemes over static ones. This shows that our state-space model is able to correctly prioritize among the recommendation metrics. It is well-known that the recommendation quality metrics may be conflicted among themselves [24]. For instance, it is relatively easy to have a high diversified ranking that is poor in accuracy. In our experiments, we show that the Chebyshev scalarization function deals better with these conflicts, providing better results. Possibly, once the Chebyshev scalarization function is able to find all points in a non-convex Pareto set, it is more likely to select the set of items that is best for each trial.



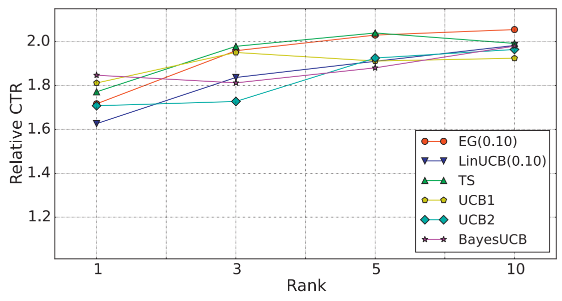
(a) Linear scalarization



(b) Chebyshev scalarization



(c) Linear scalarization



(d) Chebyshev scalarization

Fig. 4. Relative CTR's on different ranking lengths. Fig. (a– b) for Yahoo! Today News data and Fig. (c– d) KDD Cup 2012 online ads data.

If we were to recommend a single algorithm, scalarization function and weighting scheme, we would suggest, for the Yahoo! dataset, the ϵ-greedy [EG(0.10)] algorithm, Chebyshev scalarization function and SMO-NOV scheme. Whereas, for the KDD dataset, the suggestion is different: the Thompson sampling algorithm, the Chebyshev scalarization function and the DMO-RANK scheme.

5.4.2. *On the scalarization function*

We also analyze the performance considering the Thompson sampling algorithm as the MAB algorithm to suggest k items per trial, where $k \in \{1, 3, 5, 10\}$ for different scalarization functions. We chose these values for k to approximate the usual number of items suggested by real recommender systems, and the Thompson sampling algorithm because it obtained the best overall results in the experiments reported in the previous section. In Fig. 3(a)–(d) and Fig. 3(e)–(h), we show the results for the Yahoo! and KDD datasets, respectively. We report the relative CTR of multi-objective weighting schemes divided by the CTR of the single-objective baseline.

Again, the Chebyshev scalarization funtion reaches a better performance than the linear scalarization function. For instance, in the KDD dataset we have gains in all weighting schemes for $k = 1, 3,$ and 10 (Fig. 3(e), (g), (h)). A similar pattern occurs in the Yahoo! dataset. However, it seems that when we have dynamic weighting prioritization schemes for each ranking position (DMO-RANK) the linear scalarization function has a slightly better performance (see Fig. 3(a)–(d)).

5.4.3. *On number of items recommended*

In this section, we investigate the impact on CTR performance when varying the number k of displayed recommendations for different MAB algorithms. In Fig. 4(a) and (b), we present the best CTR values for the Yahoo! dataset, whereas in Fig. 4(c) and (d), we show the CTR values for the KDD dataset. To help understanding the graphs, we select a representative subset of base bandit algorithms, namely, (i) ϵ-greedy, with $\epsilon = 0.05$ [EG(0.05)], (ii) LinUCB with $\alpha = 0.10$ [LinUCB(0.10)], (iii) Thompson sampling (TS), (iv) UCB1, (v) UCB2, and (vi) Bayes UCB.

As expected, for almost all algorithms (except for EG(0.10) in Fig. 4(a) and Bayes UCB in Fig. 4(b) and (d)), it is more difficult to recommend relevant items in the first ranking position (i.e., $k = 1$). As the size of the ranking increases, the relative CTR improves. (i.e., $k = 3, 5, 10$). Fig. 4 also depicts that there is not a single base bandit algorithm that has the better performance in the evaluated scenarios, although TS and EG(0.1) are overall more consistent in their results. For instance, in Fig. 4(a), we show the results for the Yahoo! dataset considering the linear scalarization. Here, the maximum CTR@1 is given by the ϵ-greedy algorithm, whereas in Fig. 4(b), Bayes UCB algorithm has the best performance. Finally, we also see that we have different ranges of CTR values depending on the dataset used. Specifically, in Fig. 4(a) and (b), for the Yahoo! dataset, the relative CTR values range from approximately 1.3–1.7. Meanwhile, in Fig. 4(c) and (d), for the KDD dataset the relative CTR values range from approximately 1.2–2.1.

6. Conclusions and future work

This paper proposed a multi-objective ranking bandits algorithm for online recommendation. The algorithm relies on four main components: a scalarization function, a set of recommendation quality metrics, a dynamic prioritization scheme for weighting these metrics and a base MAB algorithm.

We have built upon previous work to come up with non-linear scalarization functions, capable of finding all points in a non-convex Pareto, and proposed a method to adapt the weighting scheme online for three different recommendation quality metrics, namely accuracy, novelty and diversity, based on a state-space model. Three variants of the dynamic model weighting method were proposed, considering the perspectives of the user, the recommender system and the ranking. The algorithm is flexible enough to consider different multi-armed bandit algorithms proposed in the literature.

Extensive experiments were performed with two real-world large-scale datasets, where the results in terms of click-through rate were compared with the single-objective version of the method considering different variations of the four components aforementioned. The results showed that a Chebyshev scalarization function together with e dynamic weighting scheme provide the best results for both datasets when compared to the single-objective version and other scalarization/weighting strategies. The results also show that when using the Thompson sampling and ϵ-greedy algorithms we obtain more consistent results considering recommendation lists of different sizes. Particularly, our approach provides an 7.8% and 10.4% lift in CTR compared to the state-of-the-art approach, for Yahoo! Today News dataset and KDD Cup 2012 online ads dataset, respectively

As future work, a promising extension is to analyze the impact of other relevant objectives to recommendation quality, such as coverage and serendipity [29]. Another possible extension is to consider different base MABs for each ranking position. In this case, an analyses of both the empirical performance and the theoretical properties of this mixed ranked bandit algorithm would be of interest. Since we were successful in considering multiple objectives in recommender systems, it would also be interesting to adapt the algorithm to account for queries and documents in a search engine within the online learning to rank setting. We also intend to investigate situations where we have no usage history of users and items, known as the cold-start problem [30], by modeling it as an exploration/exploitation problem. Finally, in order to deal with sparsity, we believe another good line of investigation is adding sparsity as an objective to the proposed algorithm.

Acknowledgments

The authors gratefully acknowledge the finantial support of CNPq; CEFET-MG under Proc. PROPESQ-10314/14; FAPEMIG under Proc. FAPEMIG-PRONEX-MASWeb, Models, Algorithms and Systems for the Web, process number APQ-01400-14, Microsoft Azure Sponsorship–CEFET-MG.

References

[1] Yahoo! Webscope Program, (http://webscope.sandbox.yahoo.com).
[2] C. Anderson, The Long Tail: How Endless Choice is Creating Unlimited Demand, Random House, New York City, 2007.
[3] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2–3) (2002) 235–256.
[4] D.M. Blei, Probabilistic topic models, Commun. ACM 55 (4) (2012) 77–84.
[5] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowl.-Based Syst. 46 (2013) 109–132.
[6] S. Bubeck, N. Cesa-Bianchi, Regret analysis of stochastic and nonstochastic multi-armed bandit problems, (2012) arxiv:1204.5721.
[7] N. Cesa-Bianchi, G. Lugosi, Prediction, learning, and games, Cambridge University Press, New York City, 2006.
[8] P. Cremonesi, Y. Koren, R. Turrin, Performance of recommender algorithms on top-n recommendation tasks, in: Proceedings of the Fourth ACM Conference on Recommender Systems, ACM, 2010, pp. 39–46.
[9] M.M. Drugan, A. Nowe, Designing multi-objective multi-armed bandits algorithms: a study, in: Proceedings of the International Joint Conference on Neural Networks, IEEE, 2013, pp. 1–8.
[10] G. Eichfelder, Adaptive Scalarization methods in Multiobjective Optimization, Springer, Berlin, 2008.
[11] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: evaluating recommender systems by coverage and serendipity, in: Proceedings of RecSys, ACM, 2010, pp. 257–260.
[12] L.A. Granka, T. Joachims, G. Gay, Eye-tracking analysis of user behavior in www search, in: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2004, pp. 478–479.
[13] F. Guo, C. Liu, Y.M. Wang, Efficient multiple-click models in web search, in: Proceedings of the Second ACM International Conference on Web Search and Data Mining, ACM, 2009, pp. 124–131.
[14] K. Hofmann, A. Schuth, A. Bellogin, M. De Rijke, Effects of position bias on click-based recommender evaluation, in: Advances in Information Retrieval, Springer, 2014, pp. 624–630.
[15] B. Horsburgh, S. Craw, S. Massie, Learning pseudo-tags to augment sparse tagging in hybrid music recommender systems, Artif. Intell. 219 (2015) 25–39.
[16] E. Kaufmann, O. Cappé, A. Garivier, On Bayesian upper confidence bounds for bandit problems, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2012, pp. 592–600.
[17] A. Lacerda, Contextual bandits with multi-objective payoff functions, in: Proceedings of the 4th Brazilian Conference on Intelligent Systems, IEEE, 2015, pp. 1–8.
[18] A. Lacerda, A. Veloso, N. Ziviani, Exploratory and interactive daily deals recommendation, in: Proceedings of the 7th ACM Conference on Recommender Systems, ACM, New York, NY, USA, 2013, pp. 439–442, doi:10.1145/2507157.2507228.
[19] K. Lerman, T. Hogg, Leveraging position bias to improve peer recommendation, PloS one 9 (6) (2014) e98914.
[20] L. Li, W. Chu, J. Langford, R. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of WWW, 2010, pp. 661–670.
[21] L. Li, W. Chu, J. Langford, X. Wang, Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms, in: Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, ACM, 2011, pp. 297–306.
[22] M. Lubatkin, S. Chatterjee, Extending modern portfolio theory into the domain of corporate diversification: does it apply? Acad. Manag. J. 37 (1) (1994) 109–136.
[23] J. Mary, P. Preux, O. Nicol, Improving offline evaluation of contextual bandit algorithms via bootstrapping techniques, in: Proceedings of the 31st International Conference on Machine Learning, 2014, pp. 172–180.
[24] S.M. McNee, J. Riedl, J.A. Konstan, Being accurate is not enough: how accuracy metrics have hurt recommender systems, in: Proceedings of the Extended Abstracts on Human Factors in Computing Systems, ACM, 2006, pp. 1097–1101.
[25] K.P. Murphy, Machine learning: a probabilistic perspective, 2012.
[26] F. Radlinski, R. Kleinberg, T. Joachims, Learning diverse rankings with multi--armed bandits, in: Proceedings of the 25th International Conference on Machine Learning, ACM, 2008, pp. 784–791.
[27] M.T. Ribeiro, A. Lacerda, A. Veloso, N. Ziviani, Pareto-efficient hybridization for multi-objective recommender systems, in: Proceedings of the Sixth ACM Conference on Recommender Systems, ACM, 2012, pp. 19–26.
[28] M.T. Ribeiro, N. Ziviani, E.S.D. Moura, I. Hata, A. Lacerda, A. Veloso, Multiobjective Pareto-efficient approaches for recommender systems, ACM Trans. Intell. Syst. Technol. 5 (4) (2015) 53:1–53:20.
[29] F. Ricci, L. Rokach, B. Shapira, P.B. Kantor, Recommender systems handbook, 1, Springer, New York City, 2011.
[30] A.I. Schein, A. Popescul, L.H. Ungar, D.M. Pennock, Methods and metrics for cold-start recommendations, in: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2002, pp. 253–260.
[31] A. Slivkins, F. Radlinski, S. Gollapudi, Ranked bandits in metric spaces: learning diverse rankings over large document collections, J. Mach. Learn. Res. 14 (1) (2013) 399–436.
[32] L. Tang, Y. Jiang, L. Li, T. Li, Ensemble contextual bandits for personalized recommendation, in: Proceedings of the 8th ACM Conference on Recommender systems, ACM, 2014, pp. 73–80.
[33] W.R. Thompson, On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, Biometrika (1933) 285–294.
[34] M. Tokic, Adaptive ε-greedy exploration in reinforcement learning based on value differences, in: Proceedings of the KI 2010: Advances in Artificial Intelligence, Springer, 2010, pp. 203–210.
[35] S. Vargas, Novelty and diversity enhancement and evaluation in recommender systems, (Ph.D. thesis), Universidad Autonoma de Madri, 2015.
[36] S. Vargas, P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, in: Proceedings of RecSys, 2011, pp. 109–116, doi:10.1145/2043932.2043955.
[37] K.-W. Wu, C.-S. Ferng, C.-H. Ho, A.-C. Liang, C.-H. Huang, W.-Y. Shen, J.-Y. Jiang, M.-H. Yang, T.-W. Lin, C.-P. Lee, et al., A two-stage ensemble of diverse models for advertisement ranking in KDD cup 2012, in: Proceedings of the ACM SIGKDD KDD-Cup WorkShop, 2012.
[38] S. Yahyaa, M.M. Drugan, B. Manderick, Scalarized and Pareto knowledge gradient for multi-objective multi-armed bandits, in: Transactions on Computational Collective Intelligence XX, Springer, 2015, pp. 99–116.
[39] S.Q. Yahyaa, M.M. Drugan, B. Manderick, The scalarized multi-objective multi--armed bandit problem: an empirical study of its exploration vs. exploitation tradeoff, in: Proceedings of the International Joint Conference on Neural Networks, IEEE, 2014, pp. 2290–2297.
[40] M. Zhang, N. Hurley, Avoiding monotony: improving the diversity of recommendation lists, in: Proceedings of RecSys, 2008, pp. 123–130.
[41] C. Ziegler, S. McNee, J. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the WWW, 2005, pp. 22–32.
[42] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. Da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evolut. Comput. 7 (2) (2003) 117–132.
[43] Y. Zuo, M. Gong, J. Zeng, L. Ma, L. Jiao, Personalized recommendation based on evolutionary multi-objective optimization, IEEE Comput. Intell. Mag. 10 (1) (2015) 52–62.

Anísio M. Lacerda received the Bachelor degree in computer science and the MS and Ph.D. degrees in computer science from the Universidade Federal de Minas Gerais (UFMG), Brazil, in 2005, 2008, and 2013, respectively. He has been an associate professor of computer engineering at the Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) since 2015, His research interests include is on computer science, with emphasis on Information Retrieval and Recommender Systems, acting on the following topics: algorithm design, web advertising and knowledge representation.