COM-MABs: From Users' Feedback to Recommendation

Alexandre Letard, Tassadit Amghar, Olivier Camp, Nicolas Gutowski

COM-MABs: From Users' Feedback to Recommendation

Alexandre Letard
Université d'Angers - LERIA
Kara Technology - Dpt R&D
49100, Angers, France
alexandre.letard@kara.technology

Tassadit Amghar
Université d'Angers - LERIA
49100, Angers, France

Olivier Camp
ERIS - Groupe ESEO
49100, Angers, France

Nicolas Gutowski
Université d'Angers - LERIA
49100, Angers, France

Abstract

Recently, the COMbinatorial Multi-Armed Bandits (*COM-MAB*) problem has arisen as an active research field. In systems interacting with humans, those reinforcement learning approaches use a feedback strategy as their reward function. On the study of those strategies, this paper present three contributions: 1) We model a feedback strategy as a three-step process, where each step influences the performances of an agent ; 2) Based on this model, we propose a novel *Reward Computing* process, *BUSBC*, which significantly increases the global accuracy reached by optimistic *COM-MAB* algorithms – up to 16.2% – ; 3) We conduct an empirical analysis of our approach and several feedback strategies from the literature on three real-world application datasets, confirming our propositions.

1 Introduction

Multi-Armed Bandit (*MAB*) approaches (Robbins 1952) have been extensively applied in various fields of activities as finance, healthcare or recommendation systems (Bouneffouf and Rish 2019). Through an iterative process, MAB algorithms are designed, at each round, to choose from a set of actions the one that maximizes their expected gain. They obtain good performances from a global accuracy metric view point. However, in many applications, like recommender systems (Chen, Wang, and Yuan 2013), several answers may be correct and thus, more than one item should be selected at each round. COMbinatorial Multi-Armed Bandits (*COM-MAB*) (Anantharam, Varaiya, and Walrand 1987) have been specifically designed to allow the selection of multiple arms and are thus well suited. Both *MAB* and *COM-MAB* approaches are reinforcement learning approaches in which an agent performs an action, observes a reward and changes its state (Sutton and Barto 1998). For recommender systems, the performed action is a recommendation and the reward is computed from the corresponding user feedback from which the *MAB* agent learns (Gutowski 2019). Hence, several *feedback strategies* have been implemented for the *COM-MAB* approaches (Audibert, Bubeck, and Lugosi 2011). Those strategies can be used with either the user's feedback for every arm composing the recommendation — herein, referred to as *full feedback vector setup* (FV) — (Combes et al. 2015), a part of them — herein, referred to as *partial feedback vector setup* (P) — (Saha and Gopalan 2019) or with implicitly deduced feedback values from sparse vectors (Kveton et al. 2015). From an industrial view point, those topics are particularly relevant and impactful for applications in direct interactions with users aiming at performing top-k recommendation with restricted feedback $\psi \leq k$.

Without records of prior interactions between the recommender system and the users, a fitting feedback strategy becomes essential for the efficient learning of a *COM-MAB* algorithm. However, most previous works either employ the *Bandit* (Ito et al. 2019), *Semi-Bandit* (Combes et al. 2015) strategies, or partial variants considering the feedback given for the ψ arms with highest reward expectations from the initial recommendation (Saha and Gopalan 2019). Further investigation is needed to observe how those strategies can improve the performances of *COM-MAB* approaches.

Thus, inspired by Letard et al., we formally model a feedback strategy as the succession of three processes : a) "*Feedback Identification*"; b) "*Feedback Retrieval*"; and c) "*Reward Computing*". We argue that, there is no unique optimal feedback strategy and thus that, depending on the application settings and the chosen *COM-MAB* algorithm, those underlying processes can be worked on and combined to improve the performances of the learning agent. To confirm this assumption, we propose a new *Reward Computing* process, "*Bandit Under Semi-Bandit Conditions*" (*BUSBC*). Our approach is built-upon the combination of methods from the literature and aims at enhancing the accuracy of Upper Confidence Bounds (UCB) based *COM-MAB* algorithms. The impact of each process composing a feedback strategy on the global accuracy metric was evaluated under both full feedback vector and partial feedback vector setups. Our results confirm our hypothesis as *BUSBC Reward Computing* process significantly increases the performances of UCB-like algorithms. Depending on the application and method from the literature, the observed improvement range from 0.5% to 16.2%, with an average of 4.7%.

The paper is organized as follows. In section 2 we review works related to the *COM-MAB* problem and feedback strategies. Section 3 depicts our general model for a feedback strategy and our proposed method, *BUSBC*. Section 4 discusses our experimental evaluation. Finally, we conclude and open up new perspectives in Section 5.

2 Preliminaries

2.1 Combinatorial Multi-Armed Bandits

A *MAB* problem (Robbins 1952) includes a set $\mathcal{A} = \{a_1, ..., a_m\}$ of *m* independent arms, where each arm $a \in \mathcal{A}$ is an item to recommend. As part of a recommendation system, at each iteration $t \in [1, T]$, with *T* being a known horizon, a learning agent selects an arm $a_t \in \mathcal{A}$ according to its policy π and recommends it to the user. Herein, we consider the *COM-MAB* problem (Anantharam, Varaiya, and Walrand 1987) which is a generalization of the *MAB* problem employed to recommend sets of arms $A_k = \{a_1, ..., a_k\}$, $A_k \subseteq \mathcal{A}$, with $1 \leq k \leq m$, $\forall t \in [1, T]$. Among the existing *COM-MAB* approaches, we consider the "*Multiple Plays*" method (Anantharam, Varaiya, and Walrand 1987) which allows the sequential use of classical *MAB* algorithms in order to iteratively construct a "*Super-Arm*" as follows: while $|S_t| < k$, $S_t = \cup_{i=1}^{k}\{a_i\}$, where $a_i = \text{argmax}^{\pi}_{a \in \mathcal{A} \setminus S_t} \mathbb{E}[R_{t,a}]$. Super-Arm S_t is thus the subset of k arms having the highest reward expectations according to the *MAB* algorithm's policy π. Hence, any single play algorithm of the literature can be used in a combinatorial setting.

In a stochastic setting, where the rewards are i.i.d random variables, a *COM-MAB* algorithm of policy π aims at minimizing the cumulative regret $\rho^{\pi}(T) = T\mu^* - \sum_{t=1}^{T} r_t$, where μ^* is the reward expectation of the optimal super-arm, without prior knowledge of the rewards probabilities distribution $\mu_a \in [0, 1]$ over each arm a of $\mathcal{A}$. Many real-world applications prefer to consider the maximization of the global accuracy metric $Acc^{\pi}(T) = \frac{\sum_{t=1}^{T} r_t}{T}$. Traditionally, *MAB* algorithms update their policy by a summation of the observed rewards R_t for each arms : $SR_{t,a} = SR_{t-1,a} + R_{t,a}$. With R_t being either a scalar value or a vector $R_t = \{R_{t,1}, R_{t,2}, ..., R_{t,\psi}\}$, with ψ being the number of arms for which feedback has been provided at iteration t. As works on feedback strategies impact those observed rewards, we consider $r_t = 1$ if at least half of the suggested items in S_t satisfy the user, and $r_t = 0$ otherwise (See Subsection 4.1) as assessment variable for a fairer evaluation of the studied methods on the global accuracy metric.

2.2 Feedback Strategies

COM-MAB algorithms' learning is mainly led by the rewards observed at each round. Hence, the reward function of such algorithm is of utmost importance. Feedback strategies are reward functions specifically designed for agents in interaction with humans, such as in recommender systems. Several strategies have thus been implemented to consider the feedback at each iteration *t* in order to compute the reward R_t observed by a *COM-MAB* algorithm (Audibert, Bubeck, and Lugosi 2011). Hence, we denote $Y_t = \{Y_{t,1}, Y_{t,2}, ..., Y_{t,m}\}$, the feedback vector associating a specific feedback to each arm *a* of $\mathcal{A}$ at round *t*. One should notice that Y_t denotes an abstract concept, since feedback can not be acquired for not recommended arms in real-world applications. More realistically, Y_t can represents the user's possibilities of feedback while we denote the actual feedback given by user u_t as F_t, with $F_t \subseteq Y_t$.

To the best of our knowledge, most traditional approaches are variants of the four following models: a) *Full-Information* (Audibert, Bubeck, and Lugosi 2011), where an individual reward is observed for every arm *a* of $\mathcal{A}$, whether they were in S_t or not: $R_t^{FI} = F_t = Y_t$; b) *Semi-Bandit* (Combes et al. 2015), where the individual reward of each arm *a* of S_t are revealed: $R_t^{SB} = F_t = \{Y_{t,a} \mid a \in S_t\}$; c) *Bandit* (Ito et al. 2019), where only a cumulative reward[1] associated to S_t is observed by the agent: $R_t^B = S_t^{\top} R_t^{SB}$; d) *Cascading Bandits* models (Li et al. 2016), where the reward function is dependent of the application, and aims at deducing implicitly F_t, considering a stopping criterion. One may know, from the literature (Neu 2015), that *Semi-Bandit* and *Bandit* depict both an application setting, modelling constraints on user feedback acquisition, and a method for computing the rewards from the provided user feedback. Herein, for clarity's sake, we only refer to them as rewards computing processes and express the feedback acquisition constraints by *full* or *partial feedback vector setups*.

Audibert, Bubeck, and Lugosi have proven that, among those approaches, the *Semi-Bandit* method is often the most efficient. This approach has thus been extensively studied in the literature (Sankararaman 2016). However, for some real-world applications S_t can be large (Lagrée, Vernade, and Cappé 2016). To prevent the user from having to return an equally large feedback vector, partial variants have been proposed (Luedtke, Kaufmann, and Chambaz 2016). When partial feedback strategies are used, the observed feedback vector is only defined for a subset $P_t \subseteq S_t$. More formally, for non-partial approaches, the size of the observed feedback vector is $|F_t| = |S_t|$, while with partial methods $|F_t| = \psi$ with $\psi < |S_t|$. Hence P_t is a request vector for feedback while F_t is its corresponding answer provided by user u_t, both with same dimension ψ. Herein, in both cases, the goal of the agent is to perform top-k recommendation, while learning with $\psi \leq k$ user feedbacks at each iteration.

3 System Model

3.1 Feedback Strategy: A General Model

Let $\mathcal{U} = \{u_1, ..., u_n\}$ be a set of *n* users. At each iteration $t \in [1, T]$, a user u_t is pending for a recommendation S_t of *k* items from $\mathcal{A} = \{a_1, ..., a_m\}$, where $\mathcal{A}$ is the set of items known by a recommender system built-upon a *COM-MAB* algorithm of policy π. We argue that all the previously introduced feedback strategies, used by *COM-MAB* algorithms, conform to the following three-process general model:

***Feedback Identification*:** where a set of arms $P_t \subseteq \mathcal{A}$, for which feedback will be requested to the user, is determined. with *Cascading Bandits* models, this step is used to define a stopping criterion while the user interacts with the system, e.g., the selection of a preferred item associated with arm a_s. Under a partial-vector setup, this step defines how to construct P_t from S_t. Other approaches consider either $P_t = \mathcal{A}$ or $P_t = S_t$. This process can be considered

[1] To internally compute a cumulative reward, the *Bandit* strategy needs a reward vector equivalent to the one used with *Semi-Bandit*.

as problem dependant, to some extent. Indeed, the number ψ of obtained feedbacks at each iteration, is either specific to an application or restricted by user's fatigue. However, the choice of which arms for which feedback will be provided is problem free and can be decided by the recommender system. Besides, Active Learning (Elahi, Ricci, and Rubens 2016), an emerging research area, focus on dealing with user's feedback scarcity leading to the cold-start problem. The objective is to identify the items for which feedback will be most useful to better estimate the arms' reward expectation. In most works, sub-optimal arms are thus recommended to users for a sign-up period. However, Feedback Identification process can be used to perform active learning among the determined optimal arms, leading to an improvement over feedback usefulness without sacrificing accuracy.

***Feedback Retrieval*:** where an initial feedback vector F_t is built. Except for *cascading bandits* models, the other methods presented in sub-section 2.2 require an explicit feedback from the user. In *Cascading Bandits* based feedback strategies, this step is used to infer F_t by considering the relative distances of each arm *a* in S_t from the arm a_s triggering the defined stopping criterion. These methods, particularly used in applications where a great number of items are recommended simultaneously to users, allow a more flexible behavior from the recommender system and reduce users' fatigue. Hence, many works have shown the interest of the *Feedback Retrieval* process with *Cascading Bandits* variants (Li et al. 2016; Kveton et al. 2015).

***Reward Computing*:** where the final reward R_t, observed by the agent, is computed from F_t. Thus, R_t can either be the feedback vector F_t or the result of any processing on F_t, i.e., the inner product between F_t and P_t. This process is usually not problem-dependant : as we show in our experiments (see section 4.2), the fittest method for this step is mostly dependant of the chosen *COM-MAB* algorithm. As even minute changes can lead to significant differences on performances, we consider this step as an easy and consistent way to improve recommender systems.

3.2 Bandit under Semi-Bandit Conditions: BUSBC

Following the previously exposed general model (section 3.1), we implemented a novel *Reward Computing* process, *BUSBC*. This method aims at enhancing the performances of UCB-based *COM-MAB* algorithms.

As in any feedback strategy, *Feedback Identification* and *Feedback Retrieval* are performed before the reward computing. Herein, we consider that *Feedback Retrieval* is processed by explicitly requesting feedback from user u_t. Under the full feedback vector setup, $\forall a_i \in S_t$, a feedback is requested from the user, i.e., $P_t = S_t$. Under the partial feedback vector setup, feedback will only be given for $\psi < k$ arms. The feedback vector is thus only defined for a subset $P_t \subseteq S_t$ where P_t is constructed incrementally such that: $P_t = P_{t,\psi}$ with $P_{t,0} = \varnothing$ and $\forall j \in [1, \psi]; P_{t,j} = P_{t,j-1} \cup \{a_i\}$ where $a_i \in S_t$ is selected according to the chosen *Feedback Identification* process (lines 1 to 4 of algorithm 1). In this article, we consider and further explore the following methods proposed by Letard et al.:

Reinforce - RE: which depicts the most popular *Feedback Identification* process in the literature, and consists in a scale reduction of the full feedback vector setup. It selects the ψ arms from S_t with the highest reward expectations $\mathbb{E}[R_{t,a}]$:

$$a_i = argmax^{\pi}_{a \in S_t \setminus P_{t,j-1}} \mathbb{E}[R_{t,a}] \tag{1}$$

Optimal-Exploration - OE: This approach aims at maximizing the agent's knowledge on the reward expectation distribution $\{\mu_1, ..., \mu_k\}$. It selects the ψ arms from S_t for which the least feedbacks have been provided till *t*:

$$a_i = argmin_{a \in S_t \setminus P_{t,j-1}} obs_{a,t} \tag{2}$$

where $obs_{a,t}$ is the number of observed feedbacks for arm a up to iteration t.

In the *Feedback Retrieval* process, we construct the feedback vector F_t from Y_t by requesting feedback from user u_t for each arm *a* in P_t (line 5 of algorithm 1):

$$F_t = \{Y_{t,a} \mid a \in P_t\} \tag{3}$$

As the *Reward Computing* process, *BUSBC* firstly calculates a cumulative reward R^B_t from this feedback on the current recommendation (line 6 of algorithm 1), such that :

$$R^B_t = P_t^{\top} F_t \tag{4}$$

Then the *COM-MAB* algortihm observes this reward and updates its policy only for the arms in P_t for which a positive feedback has been provided (lines 7 to 11 of algorithm 1) :

$\forall a \in P_t$, if $F_{t,a} > 0$:

$$SR_{t,a} = SR_{t-1,a} + R^B_t \tag{5}$$

where $SR_{t,a}$ is the sum of observed rewards for arm a until iteration t.

Algorithm 1 depicts the *BUSBC* approach when used under a partial feedback vector setup with *OE* as the *Feedback Identification* process. To change the identification process to *RE*, line 3 should be replaced by the corresponding selection mechanism defined by equation 1. Similarly, to consider the full-vector application of *BUSBC*, lines 2 to 4 should be removed and replaced by $P_t = S_t$. The Bandit approach can be inefficient in some cases. This can be explained both by unrealistically high rewards and by rewards provided to unsatisfying arms in S_t. The second issue can be addressed by only providing successful arms in S_t with the cumulative reward as in a *Semi-Bandit* strategy with Bernoulli rewards. Concerning the first issue, we believe that for optimistic *COM-MAB* approaches, a cumulative reward may, in actual fact, be an advantage. Thus, *BUSBC* is mainly designed to enhance UCB-based *COM-MAB* algorithms.

Algorithm 1 *P-BUSBC*-OE

Require: S_t, Recommended Super arm.
Y_t, Feedback vector associated to $\mathcal{A}$ or user u_t.
π, Agent's policy.
ψ, Number of feedback allowed.

$P_t \leftarrow \emptyset$
while $|P_t| < \psi$ **do**
  Construct P_t with
  $P_t = P_t \cup \{argmin_{a \in S_t \setminus P_t} obs_{a,t}\}$
  (according to Equation 2)
end while
Feedback Retrieval: $F_t = \{Y_{t,a} \mid a \in P_t\}$
Compute cumulative reward: $R_t^B = P_t^\top F_t$
for $a \in P_t$ **do**
  if $F_{t,a} > 0$ **then**
    Update policy π with $SR_{t,a} = SR_{t-1,a} + R_t^B$
  end if
end for

4 Experiments

4.1 Experiment Settings

Datasets: Experiments are performed on several real-life datasets: **RSASM**, **Jester** and **MovieLens** (See Table 1). For each dataset and for each arm, we consider user feedback as Bernoulli variables i.e., for each arm, $F_{t,a} = 1$ if the user considers the recommended item (arm) as relevant or 0 otherwise. For Jester and MovieLens datasets, where feedbacks are a rating between 0 and 5, arms are considered relevant if their rating is greater than or equal to 4.

Datasets	Users	Arms	Interactions	Source
RSASM	2 152	18	$> 38K$	Kaggle
Jester	59 132	150	$> 1.7M$	Kaggle
MovieLens	942	1682	$> 100K$	Groupelens.org

Table 1: Datasets

Algorithms & Baselines: Our goal is to observe the impact of the studied feedback strategies on the performances of a *COM-MAB* algorithm, with regards to its policy π. Thus, in our experiments, we applied the *multiple plays* method (Anantharam, Varaiya, and Walrand 1987) to the following popular *MAB* approaches :

- ε-greedy (Sutton and Barto 1998), with $\varepsilon = 0.0009$
- Thompson Sampling (TS) (Agrawal and Goyal 2012)
- Upper Confidence Bounds (UCB) 1 & 2 (Auer 2002)

We chose those algorithms because they are the background methods of a number of state of the art algorithms and do not take advantage of any problem dependant optimisation. Therefore, we believe that the improvements observed on those algorithms would also be observed on their more specific derived methods. Nevertheless, in this work, the *COM-MAB* algorithms are mainly the required support to allow an evaluation of feedback strategies. We compare our proposed *BUSBC* Rewards Computing process with the following baselines, still used in the literature:

- Bandit (B) (Ito et al. 2019)
- Semi-Bandit (SB) (Combes et al. 2015)
- Bandit and Semi-Bandit (BSB) (Letard et al. 2020)

Evaluation Metric: We consider the global accuracy *(Acc)* metric (Gutowski 2019) as defined by the following equation:

$$Acc(T) = \frac{\sum_{t=1}^{T} r_t}{T} \tag{6}$$

As explained in sub-section 2.1, $r_t \in \{0, 1\}$, is an *unknown evaluation reward* modeling user u_t's overall opinion of recommendation S_t. Its only purpose is to be used in the computation of the global accuracy and thus remains unknown to the agent. The reward r_t is computed using the feedback provided for each arm a of S_t as follows:

$$r_t = \begin{cases} 1 & \text{if } \sum_{a=0}^{k} Y_{t,a} \geq \frac{k}{2} \text{, with } Y_{t,a} \in \{0,1\} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

The rewards observed by a *COM-MAB* algorithm (R_t) are built following the considered feedback strategy and based on the acquired feedback. Therefore, such an independent assessment variable (r_t) is essential for a fair evaluation of the competitive approaches.

Experimental Protocol: We compare each *Reward Computing* process by experimenting them for each algorithm and dataset, under both the full-vector and partial-vector setups. In the partial-vector setup, we consider *Optimal-Exploration (OE)* and *Reinforce (RE)* as *Feedback Identification* processes. For each experiment, we simulate 10 cyclical iterations with a finite horizon of $T = 10\ 000$ rounds. The studied feedback strategies are identified by their parameters and named *S-C-I*, with :

S: *Feedback Setup* (*FV* or *P*).

C: *Reward Computing* process (*B*, *SB*, *BSB* or *BUSBC*).

I: *Feedback Identification* process (*OE*, *RE* or "/" if FV).

Real-world applications are rarely equiprobabilistic, which can lead to biased performances when evaluating models. For example, only a third of the feedbacks stored in RSASM are positive. In other words, on this dataset, each user consider on average 6 arms as relevant. [2] Hence for each dataset, a more in-depth study of optimal values for k, the number of items to recommend at each iteration, should be done. However, as our purpose is to observe feedback strategies' impact on *COM-MAB* algorithms' global accuracy, we avoid those biaises by performing experiments with $k = 6$ for every chosen dataset (RSASM being the worst case scenario). Concerning the ψ number of feedback observed at each iteration, this study focus on worst case scenario to ensure benefits for any real-world application. Saha

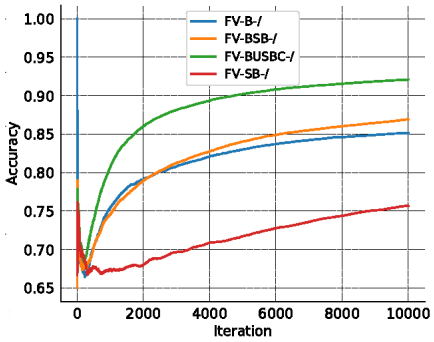
[2] Note that it is a characteristic of the original dataset.

and Gopalan have shown that, under partial feedback setup, $\psi = 2$ gave worst performances, which is thus the chosen value for our experiments. In Tables 2 and 3 we observe each algorithm's global accuracy when applied with a specific feedback strategy. Finally, in subsection 4.2 we analyse our results. We performed Kruskal-Wallis and Wilcoxon signed rank tests over the global accuracy to ensure the observed differences are significant.

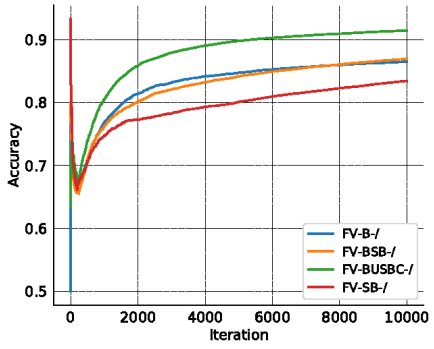
4.2 Results Analysis

Algorithm	Strategy	RSASM $Acc(T)$	Jester $Acc(T)$	MovieLens $Acc(T)$
ε-greedy	FV-B-/	0,490 $\pm$0,044	0,402 $\pm$0,003	0,879 $\pm$0,006
	FV-BSB-/	0,551 $\pm$0,008	0,416 $\pm$0,007	0,907 $\pm$0,007
	FV-BUSBC-/	**0,557** $\pm$0,008	*0,436* $\pm$0,005	*0,919* $\pm$0,004
	FV-SB-/	*0,555* $\pm$0,008	**0,453** $\pm$0,006	**0,928** $\pm$0,003
TS	FV-B-/	0,351 $\pm$0,071	0,193 $\pm$0,017	0,853 $\pm$0,005
	FV-BSB-/	0,390 $\pm$0,051	0,395 $\pm$0,018	0,855 $\pm$0,005
	FV-BUSBC-/	*0,528* $\pm$0,027	*0,415* $\pm$0,009	*0,883* $\pm$0,010
	FV-SB-/	**0,564** $\pm$0,004	**0,446** $\pm$0,005	**0,923** $\pm$0,003
UCB	FV-B-/	0,488 $\pm$0,004	0,402 $\pm$0,004	0,853 $\pm$0,004
	FV-BSB-/	*0,557* $\pm$0,005	*0,407* $\pm$0,004	*0,873* $\pm$0,004
	FV-BUSBC-/	**0,563** $\pm$0,004	**0,431** $\pm$0,005	**0,921** $\pm$0,004
	FV-SB-/	0,555 $\pm$0,004	0,404 $\pm$0,006	0,759 $\pm$0,004
UCB2	FV-B-/	0,488 $\pm$0,004	0,173 $\pm$0,005	0,871 $\pm$0,012
	FV-BSB-/	*0,545* $\pm$0,011	0,177 $\pm$0,004	*0,874* $\pm$0,005
	FV-BUSBC-/	**0,559** $\pm$0,005	**0,183** $\pm$0,003	**0,915** $\pm$0,005
	FV-SB-/	*0,545* $\pm$0,005	*0,178* $\pm$0,002	0,837 $\pm$0,002

Table 2: Results under Full-vector setup, best in bold, second in italic



(a) UCB1



(b) UCB2

Figure 1: Global accuracy's evolution on MovieLens dataset under full-vector setup

Algorithm	Strategy	RSASM $Acc(T)$	Jester $Acc(T)$	MovieLens $Acc(T)$
ε-greedy	P-B-OE	0,504 $\pm$0,037	0,418 $\pm$0,016	0,875 $\pm$0,006
	P-B-RE	0,448 $\pm$0,042	0,397 $\pm$0,014	0,860 $\pm$0,007
	P-BSB-OE	0,480 $\pm$0,042	**0,429** $\pm$0,009	0,871 $\pm$0,008
	P-BSB-RE	0,483 $\pm$0,036	0,411 $\pm$0,009	0,877 $\pm$0,005
	P-BUSBC-OE	0,511 $\pm$0,025	0,423 $\pm$0,008	0,878 $\pm$0,009
	P-BUSBC-RE	*0,527* $\pm$0,020	0,421 $\pm$0,007	*0,882* $\pm$0,006
	P-SB-OE	0,523 $\pm$0,012	0,424 $\pm$0,008	0,881 $\pm$0,004
	P-SB-RE	**0,529** $\pm$0,008	*0,425* $\pm$0,005	**0,893** $\pm$0,003
TS	P-B-OE	0,469 $\pm$0,053	0,407 $\pm$0,016	0,843 $\pm$0,008
	P-B-RE	0,361 $\pm$0,042	0,424 $\pm$0,009	0,847 $\pm$0,018
	P-BSB-OE	0,520 $\pm$0,030	*0,427* $\pm$0,012	*0,882* $\pm$0,006
	P-BSB-RE	0,498 $\pm$0,025	0,422 $\pm$0,006	0,872 $\pm$0,005
	P-BUSBC-OE	*0,533* $\pm$0,023	*0,427* $\pm$0,011	0,865 $\pm$0,008
	P-BUSBC-RE	0,507 $\pm$0,030	**0,443** $\pm$0,005	0,878 $\pm$0,011
	P-SB-OE	0,531 $\pm$0,007	0,420 $\pm$0,005	0,843 $\pm$0,006
	P-SB-RE	**0,548** $\pm$0,007	0,424 $\pm$0,002	**0,888** $\pm$0,003
UCB	P-B-OE	**0,547** $\pm$0,007	**0,433** $\pm$0,007	**0,858** $\pm$0,008
	P-B-RE	0,491 $\pm$0,021	0,395 $\pm$0,005	0,777 $\pm$0,005
	P-BSB-OE	0,534 $\pm$0,018	*0,422* $\pm$0,005	0,821 $\pm$0,007
	P-BSB-RE	0,500 $\pm$0,015	0,398 $\pm$0,003	0,784 $\pm$0,003
	P-BUSBC-OE	*0,545* $\pm$0,008	0,421 $\pm$0,006	*0,846* $\pm$0,006
	P-BUSBC-RE	0,505 $\pm$0,018	0,404 $\pm$0,005	0,826 $\pm$0,005
	P-SB-OE	0,510 $\pm$0,008	0,401 $\pm$0,006	0,754 $\pm$0,004
	P-SB-RE	0,503 $\pm$0,005	0,384 $\pm$0,004	0,714 $\pm$0,003
UCB2	P-B-OE	**0,517** $\pm$0,033	0,174 $\pm$0,007	*0,867* $\pm$0,006
	P-B-RE	0,344 $\pm$0,054	0,173 $\pm$0,007	0,849 $\pm$0,010
	P-BSB-OE	*0,512* $\pm$0,014	0,170 $\pm$0,005	0,842 $\pm$0,011
	P-BSB-RE	0,431 $\pm$0,048	**0,178** $\pm$0,004	0,852 $\pm$0,006
	P-BUSBC-OE	0,508 $\pm$0,023	0,175 $\pm$0,003	0,863 $\pm$0,014
	P-BUSBC-RE	0,466 $\pm$0,031	*0,176* $\pm$0,006	**0,875** $\pm$0,003
	P-SB-OE	0,467 $\pm$0,034	0,174 $\pm$0,002	0,805 $\pm$0,016
	P-SB-RE	0,473 $\pm$0,005	**0,178** $\pm$0,004	0,857 $\pm$0,005

Table 3: Results under Partial-vector setup, best in bold, second in italic

Summary of Experimental Results

Full feedback vector: Table 2 presents the global accuracy obtained by each algorithm on each dataset, depending on the *Reward Computing* process employed. Almost all of those results are significantly different (p-value < 0.05). Most of the non-significant differences are observed between the *FV-BSB-/* and *FV-B-/* feedback strategies, extracted from the literature. We can see that *UCB1* and *UCB2* algorithms reach higher global accuracy when using the proposed *BUSBC Reward Computing* process. Figure 1 confirms that *BUSBC* outperforms the concurring approaches after a few iterations till end of the horizon. Another point of interest it that while *BUSBC* has not been designed to enhanced the performances of greedy or bayesian algorithms, it still allows their second best results on each dataset.

Partial feedback vector: Table 3 presents the global accuracy obtained by each algorithm on each dataset, depending on the *Reward Computing* and *Feedback Identification* processes applied. The differences between the studied *Reward Computing* are smaller but still significant (p-value < 0.05). This can be explained by the feedback reduction leading to a closer behavior between the studied approaches. Concerning *Feedback Identification* process, we also observe an impact on the global accuracy of the *COMMAB* algorithm with *OE* outperforming *RE* most of the time. Note that the Bandit *Rewards Computing* (*B*) process is more competitive under a partial-vector setup than under

the full-vector setup, becoming the best choice for *UCB1* when applied with *OE*. This is mainly explained by the decrease of the number of unexpected rewarded arms. However, from the global accuracy evolution, we observed that, for both *UCB1* and *UCB2* algorithms and for each dataset, *P-BUSBC-OE* was becoming increasingly equivalent to *P-B-OE* . Hence, *P-BUSBC-OE* may, in fact, be a better choice when considering a greater horizon or when more feedback are allowed at each round. To top it all, an important point to note is that on this setting, the evaluation of the *Rewards Computing* processes is 1) restricted by the small number of feedbacks provided; 2) influenced by the *Feedback Identification* process employed. When averaging the results obtained with each *Reward Computing* process, that is, indepently from the *Feedback Identification* process employed, *BUSBC* actually still performs better than the Bandit (*B*) strategy with *UCB1* and *UCB2* for all datasets.

Discussion: The general feedback strategy model proposed in this article is defined as the application of three processes: a) *Feedback Identification* ; b) *Feedback Retrieval* ; c) *Reward Computing*. Based on our results, we observe that both the *Feedback Identification* and *Reward Computing* processes have a significant impact on the global accuracy. As the interest of the *Feedback Retrieval* process has previously been highlighted in the literature with *Cascading Bandits* models, we can conclude that any step of our general model can be used to significantly enhance the performances of a *COM-MAB* algorithm. A limitation of this study is the number of feedback employed under a partial-setup. The behaviors of the considered *Reward Computing* processes become increasingly closer as the number of perceived feedback become smaller. Similarly, with less feedback to identify, the impact observed from different *Feedback Identification* processes will also be smaller. In this study, since the number of feedback employed would be dependant of the application, we considered the worst case enabling differences between those approaches to ensure that the observed differences between each of the *Reward Computing* and *Feedback Identification* processes studied will be greater in real-world applications.

5 Conclusion

In this paper, we have formally defined a three-process model for feedback strategy and have empirically shown that for any *COM-MAB* algorithm, global accuracy can be improved by tuning any of those three processes. We proposed the *BUSBC* approach for the *Reward Computing* process, which gives significantly better results for *UCB*-based algorithms, without adding any constraint compared to previous methods. Opposing most of recents works, this paper only considers background algorithms and feedback strategies without considering any problem-dependant settings or optimizations (e.g. context availability). Our motivations were two-fold : 1) Showing that feedbacks strategies could be used to improve *COM-MAB* alorithms performances even on worst case scenario, i.e. when only items index and users' feedbacks can be used and thus only minute changes are possible ; 2) Illustrating benefits that can be extended to state of the art approaches built-upon the underlying *COM-MAB* algorithms studied. Hence, we believe that any recommender system, built-upon *COM-MAB* algorithms can benefit from this work. We also observed that the optimal feedback strategy for a *COM-MAB* algorithm may change over time, one of the promising perspectives would thus be to implement a portfolio-based algorithm to dynamically select different processes during the exploitation step.

References

Agrawal, S., and Goyal, N. 2012. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*.

Anantharam, V.; Varaiya, P.; and Walrand, J. 1987. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i: I.i.d. rewards. *IEEE TSA*.

Audibert, J.-Y.; Bubeck, S.; and Lugosi, G. 2011. Minimax policies for combinatorial prediction games. *COLT*.

Auer, P. 2002. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*.

Bouneffouf, D., and Rish, I. 2019. A survey on practical applications of multi-armed and contextual bandits. *ARXIV*.

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial multi-armed bandit: General framework and applications. In *ICML*.

Combes, R.; Shahi, M. S. T. M.; Proutiere, A.; and Lelarge, M. 2015. Combinatorial bandits revisited. In *NIPS*.

Elahi, M.; Ricci, F.; and Rubens, N. 2016. A survey of active learning in collaborative filtering recommender systems. *Computer Science Review*.

Gutowski, N. 2019. *Context-aware recommendation systems for cultural events recommendation in Smart Cities*. Ph.D. Dissertation, Université d'Angers.

Ito, S.; Hatano, D.; Sumita, H.; Takemura, K.; Fukunaga, T.; Kakimura, N.; and Kawarabayashi, K.-I. 2019. Improved regret bounds for bandit combinatorial optimization. In *NIPS*.

Kveton, B.; Wen, Z.; Ashkan, A.; and Szepesvari, C. 2015. Combinatorial cascading bandits. In *NIPS*.

Lagrée, P.; Vernade, C.; and Cappé, O. 2016. Multiple-play bandits in the position-based model. In *NIPS*.

Letard, A.; Amghar, T.; Camp, O.; and Gutowski, N. 2020. Partial bandit and semi-bandit: Making the most out of scarce users' feedback. In *ICTAI*.

Li, S.; Wang, B.; Zhang, S.; and Chen, W. 2016. Contextual combinatorial cascading bandits. In *ICML*.

Luedtke, A.; Kaufmann, E.; and Chambaz, A. 2016. Asymptotically optimal algorithms for multiple play bandits with partial feedback. *ARXIV*.

Neu, G. 2015. First-order regret bounds for combinatorial semi-bandits. In *JMLR*.

Robbins, H. 1952. Some aspects of the sequential design of experiments.

Saha, A., and Gopalan, A. 2019. Combinatorial bandits with relative feedback. In *NIPS*.

Sankararaman, K. A. 2016. Semi-bandit feedback : A survey of results. In *ARXIV CoRR*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.