Bandits-Manchots Combinatoires : du retour utilisateur à la recommandation

A Letard, T Amghar, O Camp, N Gutowski

▶ **To cite this version:**

A Letard, T Amghar, O Camp, N Gutowski. Bandits-Manchots Combinatoires : du retour utilisateur à la recommandation. CNIA 2021 : Conférence Nationale en Intelligence Artificielle, Jun 2021, Bordeaux, France. pp.52–59. hal-03321206

HAL Id: hal-03321206
https://hal.science/hal-03321206

Submitted on 17 Aug 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Bandits-Manchots Combinatoires: du retour utilisateur à la recommandation

A. Letard[1, 2], T. Amghar[2], O. Camp[3], N. Gutowski[2]

[1] Kara Technology - Dpt R&D, F-49124 St Barthélémy dÁnjou, France
[2] Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France
[3] Groupe ESEO - ERIS, F-49000 Angers, France

alexandre.letard@kara.technology

Résumé

*Récemment, le problème des Bandits-Manchots COMbinatoires (*COM-MAB*) a été sujet de nombreux travaux de recherche. Au sein de systèmes en interaction avec des humains, ces techniques, basées sur un apprentissage par renforcement, exploitent une stratégie de considération du retour utilisateur en guise de fonction de récompense. Dans l'étude de ces stratégies, cet article présente les contributions suivantes : 1) Nous proposons un modèle général de stratégie en trois étapes : Feedback Identification, Feedback Retrieval et Reward Computing, chacune influant sur les performances d'un agent ; 2) Suivant ce modèle, nous proposons une nouvelle méthode de Reward Computing,* BUSBC*, améliorant significativement la précision globale des algorithmes optimistes ; 3) Nous réalisons une analyse empirique de notre approche et d'autres stratégies issues de la littérature. Nos expérimentations, réalisées sur trois jeux de données issus d'applications réelles, confirment nos propositions avec des retours utilisateurs complets ou partiels.*

Mots-clés

Bandits-Manchots Combinatoires, Systèmes de Recommandations, Apprentissage par Renforcement

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, namely : Feedback Identification, Feedback Retrieval and Reward Computing, 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 ; 3) We conduct an empirical analysis on our approach and several feedback strategies from the literature. Our experiments, conducted on three real-world datasets, confirm our propositions with significant results whether full or partial feedback vector are used.*

Keywords

Combinatorial Multi-Armed Bandits, Recommender Systems, Reinforcement Learning

1 Introduction

Les techniques de Bandits-Manchots (*MAB*) [19] sont aujourd'hui employées dans de nombreux secteurs d'activités tels que la finance, le domaine médical ou les systèmes de recommandation [7]. Ces approches procurent de bons résultats en termes de précision globale lorsqu'un compromis entre exploitation et exploration doit être réalisé. Cependant, dans certaines applications, un agent doit être capable de recommander plusieurs éléments à chaque itération [8, 14]. Les Bandits-Manchots COMbinatoires (*COM-MAB*) [2, 8], spécialement conçus pour ces situations, constituent alors un choix judicieux. Les approches *MAB* et *COM-MAB* sont des techniques basées sur un apprentissage par renforcement : un agent réalise une action, observe une récompense et change son état [22]. Dans le cadre des systèmes de recommandations, une action correspond à une recommandation et une récompense est déterminée à partir du retour utilisateur émis suite à cette recommandation [10]. Ainsi, plusieurs stratégies de prise en compte du retour utilisateur – désignées sous le terme de "*stratégies*" dans la suite de cet article – ont été définies pour les approches *COM-MAB* [3]. Dans un cadre applicatif réel, ces stratégies peuvent être employées avec différents types de retours utilisateur : 1) Des *vecteurs complets*, lorsqu'un retour utilisateur est acquis pour chaque élément composant la recommandation [9] ; 2) Des *vecteurs partiels* [20, 15], lorsque l'utilisateur émets des retours pour une partie des éléments de la recommandation ; 3) Des vecteurs implicitement déduits des interactions entre l'usager et le système de recommandations [12, 16], tels que lors de l'emploi de modèles dits de *Bandits en Cascades*.
Pour les applications en directe interaction avec les usagers, ces sujets présentent un intérêt certain. Le jeu de donnée RSASM [1], issu d'une application réelle, en est un exemple représentatif où l'objectif est de recommander à des utilisateurs des activités à réaliser, sans connaissances initiales

1. *Recommendation System for Angers Smart City*

quant-à leurs préférences. Ce scénario correspond à un problème *MAB* ou *COM-MAB*, où les bras sont les activités pouvant être proposées et où les retours utilisateurs expriment l'opinion des usagers quant-aux activités qui leur sont recommandées. Dans de telles applications, plus le nombre d'éléments recommandés à chaque itération est important, plus il est difficile de demander aux usagers un retour pour chacun des bras de la recommandation.

En l'absence d'historique d'interactions entre le système et les utilisateurs, une stratégie adaptée est essentielle à l'apprentissage efficace d'un algorithme *COM-MAB*. Cependant, à notre connaissance, les principales approches aujourd'hui considérées sont *Bandit* [11], *Semi-Bandit* [9], ou des variantes de la stratégie *Semi-Bandit* employant des vecteurs de retours utilisateur partiels portant sur $\psi < k$ bras parmi les k éléments constituant la recommandation [20]. Des études plus approfondies sont nécessaires pour déterminer comment améliorer les approches *COM-MAB* avec ces stratégies d'apprentissage.

Ainsi, inspiré par de précédents travaux de la littérature [15, 3], nous relevons que toute stratégie peut être définie comme la succession de trois processus : a) "*Feedback Identification*" ; b) "*Feedback Retrieval*" ; c) "*Reward Computing*". Nous affirmons que chacun de ces processus influe sur les performances de l'algorithme *COM-MAB* utilisé et peut donc permettre un gain de précision globale par des ajustements adaptés. Afin de confirmer cette hypothèse, nous proposons une nouvelle méthode pour le processus "Reward Computing", "*Bandit Under Semi-Bandit Conditions*" (*BUSBC*). Notre approche est basée sur la combinaison de méthodes connues et vise à améliorer les performances des algorithmes *COM-MAB* de type *UCB*.

Nous avons réalisé des expérimentations avec plusieurs algorithmes *COM-MAB* sur trois jeux de données issus d'applications réelles. Nous avons évalué l'impact sur la précision globale de plusieurs approches pour chacun des processus identifié dans notre modèle général, avec des vecteurs de retours utilisateur complets et partiels. Nos résultats confirment nos hypothèses et montrent que *BUSBC* améliore significativement les performances des algorithmes de type *UCB*. Nous relevons également, parmi celles étudiées, des stratégies optimales pour chacun des algorithmes *COM-MAB* considérés.

Cet article est organisé comme suit. La section 2 présente les notions relatives au problème *COM-MAB* et aux stratégies de considération du retour utilisateur. La section 3 définit notre modèle général ainsi que notre approche *BUSBC*. La section 4 expose nos résultats expérimentaux. Enfin, nous concluons et ouvrons de nouvelles perspectives de recherches dans la section 5.

2 Préliminaires

2.1 Bandits-Manchots Combinatoires

Un problème *MAB* [19] implique un ensemble $\mathcal{A} = \{a_1, ..., a_m\}$ de *m* bras indépendants, où chaque bras $a \in \mathcal{A}$ est un élément à recommander. Au sein d'un système de recommandation, à chaque itération $t \in [1, T]$, *T* étant un horizon connu, un agent sélectionne un bras $a_t \in \mathcal{A}$ selon sa politique π et le recommande à l'utilisateur. Dans cet article, nous considérons le problème *COM-MAB* [2], consistant en une généralisation du problème *MAB* où l'agent doit recommander un ensemble de bras $A_k = \{a_1, ..., a_k\}$, $A_k \subseteq \mathcal{A}$, avec $1 \leq k \leq m$, $\forall t \in [1, T]$. Parmi les approches existantes, nous considérons la méthode "*Multiple Plays*" [2] qui permet l'emploi séquentiel d'un algorithm *MAB* pour définir incrémentalement un "*Super-Bras*" [8] comme suit : Tant que $|S_t| < k$, $S_t = \cup_{i=1}^{k}\{a_i\}$ où $a_i = argmax^{\pi}_{a \in \mathcal{A} \setminus S_t} \mathbb{E}[R_{t,a}]$. Le Super-Bras S_t est donc le sous-ensemble de k bras de plus haute espérance de récompense selon la politique π de l'algorithme employé. Ainsi, tout algorithme *MAB* de la littérature peut être employé dans un cadre combinatoire, qu'il soit stochastique [4], non-stochastique [6], Bayésien [1] ou avec adversaire [5].

Dans un cadre stochastique, où les récompenses sont considérées comme des variables aléatoires indépendantes et identiquement distribuées, un algorithme *COM-MAB* de politique π vise à minimiser le regret cumulé $\rho^{\pi}(T) = T\mu^* - \sum_{t=1}^{T} r_t$, où μ^* correspond à l'espérance de récompense du super-bras optimal, sans connaissances préalable quant à la distribution des espérances de récompenses $\mu_a \in [0, 1]$ parmi les bras a de $\mathcal{A}$. Dans de nombreuses applications réelles, il est préféré de considérer la maximisation de la précision globale $Acc^{\pi}(T) = \frac{\sum_{t=1}^{T} r_t}{T}$. Cette métrique est fréquemment employé pour évaluer les approches de Bandits-Manchots [10]. Dans cet article, nous considérons d'une part une "*récompense d'évaluation*" $r_t \in \{0, 1\}$ pour la recommandation S_t, inconnue de l'agent et telle que $r_t = 1$ si au moins la moitié des éléments recommandés dans S_t donnent satisfaction à l'utilisateur, et $r_t = 0$ sinon (voir sous-section 4.1.3). D'autre part, à chaque itération, l'agent *COM-MAB* observe une récompense R_t, calculée selon la stratégie appliquée. Cette récompense peut être soit un scalaire, soit un vecteur $R_t = \{R_{t,1}, R_{t,2}, ..., R_{t,\psi}\}$, avec ψ désignant le nombre de bras pour lesquels un retour utilisateur a été émis. Cette récompense est ensuite employée pour actualiser la connaissance de la distribution des espérances de récompenses des bras. Pour les algorithmes considérés, cette actualisation est réalisée au travers de la sommation des récompenses perçues jusqu'à l'itération courante, pour chacun des bras a de $\mathcal{A}$: $SR_{t,a} = SR_{t-1,a} + R_{t,a}$.

2.2 La considération du retour utilisateur

L es algorithmes *COM-MAB*, basés sur de l'apprentissage par renforcement, apprennent à l'aide des récompenses observées à chaque itération. La fonction de récompense de tels algorithmes est donc essentielle à leur fonctionnement. Une stratégie de prise en compte du retour utilisateur est une fonction de récompense particulière employée par des agents en interaction avec des humains, tels qu'au sein de systèmes de recommandations. Plusieurs stratégies ont donc été proposées pour les algorithmes *COM-MAB* [3]. Ainsi, soit $Y_t = \{Y_{t,1}, Y_{t,2}, ..., Y_{t,m}\}$, le vecteur de retours utilisateur associant un retour spécifique à chacun des bras *a* de $\mathcal{A}$ à l'itération *t*. Au sein d'applications réelles, puisqu'aucun retour de l'utilisateur ne peut être acquis pour les

bras non-recommandés, Y_t est un concept abstrait. Ainsi, soit $F_t \subseteq Y_t$ le vecteur de retours utilisateur réellement perçu par le système.

A notre connaissance, la plupart des approches traditionnelles sont des variantes des modèles suivants : a) *Full-Information* [3], où une récompense individuelle est observée pour tous les bras *a* de $\mathcal{A}$, qu'ils soient ou non [2] inclus dans S_t : $R_t^{FI} = F_t = Y_t$; b) *Semi-Bandit* [9], où une récompense individuelle, associée à chaque bras *a* de S_t, est révélée : $R_t^{SB} = F_t = \cup_a S_{t,a} Y_{t,a}$; c) *Bandit* [11], où seulement une récompense cumulée [3] associée à S_t est perçue par l'agent : $R_t^B = S_t^\top R_t^{SB}$; d) Les modèles de *Bandits en Cascade* [16], dépendants de l'application, et visant à déduire implicitement F_t en considérant un critère d'arrêt, p.ex., un clic de l'utilisateur sur le premier élément le satisfaisant dans S_t. Dans la littérature [3, 18], *Semi-Bandit* et *Bandit* désignent à la fois un niveau d'exhaustivité des retours utilisateur perçus et une méthode pour déterminer les récompenses observées par l'algorithme. Dans cet article, nous évoquerons uniquement ces approches en tant que procédés de calcul des récompenses et désignerons les contraintes liées à l'acquisition de retours utilisateur par les termes de *vecteurs complets* ou *partiels*.

Il a été démontré que, parmi ces méthodes, *Semi-Bandit* est plus efficace dans de nombreux cas [3]. Cette approche a donc été particulièrement étudiée par la littérature [21]. Cependant, dans certaines applications, S_t peut regrouper un grand nombre d'éléments [13, 14, 15]. Pour éviter à l'usager de fournir un aussi grand nombre de retours, des variantes considérant des vecteurs de retours utilisateur partiels ont été proposées [15, 17]. Dans ce cadre partiel, le vecteur de retours utilisateur n'est défini que pour un sous-ensemble $P_t \subseteq S_t$. Ainsi, P_t est un vecteur de bras pour lesquels un retour est demandé à l'utilisateur, tandis que F_t est le vecteur de retours fournit en réponse par l'utilisateur u_t, les deux vecteurs étant de même dimension ψ. Dans cet article, l'objectif d'un agent est de recommander les k meilleurs bras à chaque itération. Lors de l'emploi de vecteurs complets, cet objectif est suivi en percevant à chaque itération k retours utilisateur tandis que seulement $\psi < k$ retours sont prodigués lors de l'emploi de vecteurs partiels.

3 Modélisation

3.1 Stratégie de considération du retour utilisateur : un modèle général

Soit $\mathcal{U} = \{u_1, ..., u_n\}$ un ensemble de *n* utilisateurs. À chaque itération $t \in [1, T]$, un utilisateur u_t sollicite une recommandation S_t de *k* éléments extraits de $\mathcal{A} = \{a_1, ..., a_m\}$, où $\mathcal{A}$ est l'ensemble des éléments connus par un système de recommandation employant un algorithme *COM-MAB* de politique π. Nous affirmons que chacune des stratégies précédemment exposées s'ancre dans un modèle générique composé de trois processus successifs :

Feedback Identification **:** Un ensemble $P_t \subseteq \mathcal{A}$, pour lequel des retours utilisateurs seront attendus, est déterminé. Pour les modèles de *Bandits en Cascade*, cette étape définit le critère d'arrêt à relever dans l'interaction de l'utilisateur avec le système, p.ex., la sélection d'un élément préféré associé à un bras a_s de S_t. Lors de la considération d'un vecteur partiel de retours utilisateur, cette étape définit la méthode pour construire P_t à partir de S_t. Les autres approches considèrent soit $P_t = \mathcal{A}$ ou $P_t = S_t$.

Feedback Retrieval **:** Un vecteur de retours utilisateur initial F_t est construit. Pour les *Bandits en Cascades*, cette étape est employée pour déduire implicitement F_t en considérant les distances relatives de chaque bras a de S_t par rapport au bras a_s ayant déclenché le critère d'arrêt précédemment défini. Les autres méthodes présentées à la sous-section 2.2 sollicitent explicitement un retour de l'utilisateur pour chacun des bras constituant P_t.

Reward Computing **:** Une récompense finale R_t, observée par l'agent, est calculée à partir de F_t. Ainsi, R_t peut correspondre au vecteur de retours utilisateur F_t lui-même, ou au résultat de n'importe quel traitement appliqué sur F_t, p.ex., une pondération ou un produit scalaire entre F_t et P_t.

De nombreux travaux ont montré l'intérêt du processus *Feedback Retrieval* avec des variantes de *Bandits en Cascades* [16, 12]. Dans cet article, notre objectif est d'étudier l'impact sur la précision globale des processus de *Feedback Identification* et *Reward Computing*.

3.2 Bandit under Semi-Bandit Conditions : BUSBC

Suivant ce modèle général, nous avons implémenté un nouveau processus de *Reward Computing*, *BUSBC*. Cette méthode est basée sur la combinaison du principe de récompense cumulée de l'approche *Bandit* et des conditions d'octroi de récompenses individuelles de l'approche *Semi-Bandit*. L'objectif de cette démarche est d'améliorer les performances des algorithmes *COM-MAB* de type *UCB* en canalisant l'usage de la récompense cumulée.

Comme pour toute stratégie, les processus *Feedback Identification* et *Feedback Retrieval* doivent être appliqués avant d'employer notre méthode. Dans cet article, nous considérons que le processus *Feedback Retrieval* est réalisé par des demandes explicites de retours utilisateur à l'usager u_t. Ainsi, lors de l'emploi de vecteurs complets de retours utilisateur, $\forall a_i \in S_t$, un retour est sollicité auprès de l'utilisateur, c.-à-d., $P_t = S_t$. Avec des vecteurs partiels, des retours ne sont prodigués que pour $\psi < k$ bras. Le vecteur de retours utilisateur est donc seulement défini pour un sous-ensemble $P_t \subseteq S_t$, construit incrémentalement tel que : $P_t = P_{t,\psi}$ avec $P_{t,0} = \varnothing$ et $\forall j \in [1, \psi]; P_{t,j} = P_{t,j-1} \cup \{a_i\}$ où $a_i \in S_t$ est sélectionné selon le processus de *Feedback Identification* (lignes 1 à 4 de l'algorithme 1). Concernant ce processus de *Feedback Identification*, nous considérons et approfondissons l'étude des approches suivantes, extraites de la littérature [14, 15] :

Reinforce - RE : Cette méthode décrit le procédé le plus répandu pour le processus de *Feedback Identi-*

2. *Full-Information feedback* n'est possible que pour les applications n'étant pas en directe interaction avec les usagers.

3. En simulation ou pour le calcul objectif de cette récompense, un vecteur de retours utilisateur, tel qu'avec *Semi-Bandit* est nécessaire.

fication. Elle sélectionne les ψ bras de S_t présentant les plus hautes espérances de récompenses $\mathbb{E}[R_{t,a}]$:

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

Optimal-Exploration - OE : Cette méthode vise à maximiser la connaissance de l'agent sur la distribution des espérances de récompenses $\{\mu_1, ..., \mu_k\}$ des bras composant S_t sans considérer son état courant en regard de la politique π de l'agent. Elle sélectionne les ψ bras de S_t pour lesquels le moins de retours utilisateur ont été fournis à l'itération *t* :

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

Où $obs_{a,t}$ est le nombre de retours utilisateur émis pour le bras a jusqu'à l'itération t.

Lors de l'exécution du processus *Feedback Retrieval*, nous construisons le vecteur de retours utilisateur F_t à partir de Y_t en sollicitant l'utilisateur u_t pour un retour sur chacun des bras *a* inclus dans P_t (ligne 5 de l'algorithme 1) :

$$F_t = \{Y_{t,a} \mid a \in P_t\} \quad (3)$$

En tant que processus de *Reward Computing*, *BUSBC* détermine d'abord une récompense cumulée R^B_t (ligne 6 de l'algorithme 1), telle que :

$$R^B_t = P_t^{\top} F_t \quad (4)$$

L'algorithme *COM-MAB* observe ensuite cette récompense et l'utilise pour mettre à jour sa politique uniquement pour les bras de P_t jugés pertinents par l'utilisateur u_t (lignes 7 à 11 de l'algorithme 1) :

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

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

Où $SR_{t,a}$ est la somme des récompenses observées pour le bras a jusqu'à l'itération t.

L'algorithme 1 décrit l'utilisation de *BUSBC* dans une stratégie considérant des vecteurs partiels de retours utilisateur avec *OE* en tant que processus de *Feedback Identification*. Pour changer le processus de *Feedback Identification* pour *RE*, la ligne 3 devrait être remplacée par le mécanisme de sélection correspondant, défini par l'équation 1. De la même façon, pour employer *BUSBC* au sein d'une stratégie considérant des vecteurs complets de retours utilisateur les lignes 1 à 4 devraient être remplacées par $P_t = S_t$.
La stratégie *Bandit* peut être inefficace dans certains cas, notamment à cause de récompenses cumulées trop importantes et de l'octroi de ces récompenses à tous les bras de S_t, incluant ceux qui n'ont pas été satisfaisants en réalité. Le second problème peut être traité en n'attribuant les récompenses cumulées qu'aux bras de S_t dont le succès a été effectivement observé, tel que dans une approche *Semi-Bandit* avec des récompenses de Bernoulli. Concernant le premier problème, notre théorie est qu'une récompense cumulée pourrait en réalité être avantageuse pour les approches optimistes. Ainsi, *BUSBC* est principalement conçu pour les algorithmes *COM-MAB* de type UCB.

Algorithme 1 : *P-BUSBC*-OE

Entrées : S_t : Super-bras recommandé.
Y_t : Retours associés à $\mathcal{A}$ ou utilisateur u_t.
π : Politique de l'agent.
ψ : Nombre de retours utilisateur attendus.

1 $P_t \leftarrow \emptyset$
2 **tant que** $|P_t| < \psi$ **faire**
3 Construire P_t avec : $P_t = P_t \cup \{argmin_{a \in S_t \setminus P_t} obs_{a,t}\}$
4 **fin**
5 Acquérir les retours utilisateur : $F_t = \{Y_{t,a} \mid a \in P_t\}$
6 Calculer la récompense cumulée : $R^B_t = P_t^{\top} F_t$
7 **pour** $a \in P_t$ **faire**
8 **si** $F_{t,a} > 0$ **alors**
9 Mettre à jour la politique π avec : $SR_{t,a} = SR_{t-1,a} + R^B_t$
10 **fin**
11 **fin**

4 Expérimentations

4.1 Cadre Expérimental

4.1.1 Jeux de Données

Nos expériences sont exécutées sur plusieurs jeux de données issus d'applications réelles : **RSASM**, **Jester** et **MovieLens** (voir Tableau 1). RSASM est un jeu de données dédié à la recommandation de services, où les retours utilisateur sont des variables de Bernoulli, c.-à-d., pour chaque bras a, $F_{t,a} = 1$ si l'utilisateur considère l'élément associé au bras a comme pertinent ou $F_{t,a} = 0$ sinon. Jester traite de la recommandation de blagues et MovieLens est un jeu de données pour la recommandation de films très employé dans la littérature. Pour ces deux jeux de données, les retours utilisateur correspondent à des évaluations comprises entre 0 et 5. Nous considérons que $F_{t,a} = 1$ si l'évaluation est supérieure ou égale à 4, et $F_{t,a} = 0$, sinon.

Jeu de données	Utilisateurs	Bras	Interactions	Source
RSASM	2 152	18	$> 38K$	Kaggle
Jester	59 132	150	$> 1.7M$	Kaggle
MovieLens	942	1682	$> 100K$	Groupelens.org

Tableau 1 – Jeux de données

4.1.2 Algorithmes & Approches Concurrentes

Notre objectif est d'observer l'impact de stratégies de prise en compte du retour utilisateur sur la précision globale d'algorithmes *COM-MAB* en regard de leur politique π. Ainsi, lors de nos expériences, nous avons appliqué l'algorithme *Multiple Plays* [2] aux algorithmes *MAB* suivants, extraits de la littérature :

- *ε-greedy* [22], avec $\varepsilon = 0.0009$
- *Thompson Sampling (TS)* [1]
- *Upper Confidence Bounds 1 (UCB1)* [4]
- *Upper Confidence Bounds 2 (UCB2)* [4]

Ces algorithmes ont inspiré de nombreuses variantes au sein de la littérature. Puisqu'ils ne tirent avantage d'aucune optimisation dépendante du cadre applicatif (p.ex. disponibilité d'informations contextuelles), nous pensons que nos observations quant-à leurs performances seront également valables dans leurs déclinaisons plus récentes et spécifiques. Néanmoins, cette étude porte sur les stratégies de prise en compte du retour utilisateur ; les algorithmes *COM-MAB* fournissent un cadre d'évaluation et ne sont pas en compétition. Nous comparons donc notre méthode *BUSBC* avec les approches suivantes, extraites de la littérature :

- *Bandit (B)* [3, 11]
- *Semi-Bandit (SB)* [3, 9]
- *Bandit and Semi-Bandit (BSB)* [15]

À notre connaissance, cet article est également le premier a considérer des stratégies *Bandit (B)* avec des vecteurs de retours utilisateur partiels.

4.1.3 Métrique

Précision Globale. Nous considérons la précision globale [10], définie par l'équation suivante :

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

Comme expliqué à la sous-section 2.1, $r_t \in \{0, 1\}$, est une *récompense d'évaluation* modélisant l'opinion globale de l'utilisateur u_t concernant la recommandation S_t. Elle n'est employée que dans le calcul de la précision globale et demeure inconnue de l'agent. Cette récompense est calculée en considérant les retours utilisateur associés à chacun des bras a de S_t comme suit :

$$r_t = \begin{cases} 1 & \text{si } \sum_{a=0}^{k} F_{t,a} \geq \frac{k}{2} \text{ , avec } F_{t,a} \in \{0, 1\} \\ 0 & \text{sinon} \end{cases} \tag{7}$$

Les récompenses observées par l'algorithme *COM-MAB* (R_t) sont déterminées selon la stratégie employée à partir des retours utilisateur acquis. Une variable d'évaluation (r_t), opérant indépendamment de l'algorithme et de la stratégie est donc nécessaire à l'évaluation de ces approches. Nous pouvons ainsi observer objectivement les impacts de la stratégie employée et de la quantité de retours utilisateur perçus sur les performances de l'algorithme.

4.1.4 Protocole Expérimental

Les stratégies évaluées dans cet article impliquent les processus de *Reward Computing* suivants : *Bandit*, *Semi-Bandit*, *Bandit and Semi-Bandit*, *Bandit under Semi-Bandit Conditions*. Nous les évaluons pour chaque algorithme *COM-MAB* sur chaque jeu de données, avec des vecteurs de retours utilisateur complets et partiels. Lors de l'emploi de vecteurs partiels, nous comparons pour le processus de *Feedback Identification* les méthodes *Optimal-Exploration* et *Reinforce*. Les stratégies étudiées dans cet article sont identifiées par leurs paramètres et nommées comme suit :

S-C-I, avec :

S : Le niveau d'exhaustivité des retours utilisateur, des vecteurs complets (FV) ou des vecteurs partiels (P).

C : Le processus de *Reward Computing*, *Bandit* (*B*), *Semi-Bandit* (*SB*), *Bandit and Semi-Bandit* (*BSB*) ou *Bandit under Semi-Bandit Conditions* (*BUSBC*).

I : Le processus de *Feedback Identification*, *Optimal-Exploration* (*OE*), *Reinforce* (*RE*) ou "/" lors de l'emploi de vecteurs complets.

Le nombre k d'éléments recommandés à chaque itération à été choisi selon les jeux de données : RSASM dispose du plus faible nombre de bras (18) et seulement un tiers des retours exprimés sont positifs. Puisque 6 bras, en moyenne, peuvent satisfaire un utilisateur, nous considérons des recommandations de $k = 6$ éléments. Avec des retours utilisateur partiels, les algorithmes *COM-MAB* présentent leurs pires performances pour $\psi = 2$ [20]. Afin d'observer les pires performances des algorithmes selon la stratégie employée, nous considérons donc $\psi = 2$ dans ces expériences. Pour chaque expérience, nous exécutons 10 simulations d'horizon $T = 10\ 000$ itérations. Ainsi, avec des vecteurs complets, 60 000 retours auront été émis au terme de l'horizon T, contre 20 000 avec des vecteurs partiels. Pour simuler l'arrivée séquentielle d'utilisateurs, nous les sélectionnons aléatoirement un par un depuis le jeu de données.
Les Tableaux 2 et 3 présentent la précision globale obtenue par chaque algorithme en fonction de la stratégie employée. Le Tableau 4 expose les stratégies offrant les meilleures performances en moyenne, pour chaque algorithme, lors de l'emploi de vecteurs complets et partiels. La Figure 1 présente l'évolution de la précision globale de *UCB1* et *UCB2* sur MovieLens lors de la considération de vecteurs complets, en fonction du processus de *Rewards Computing* appliqué. La Figure 2 montre l'évolution de la précision globale de *ϵ-Greedy* et *UCB1* sur RSASM avec des vecteurs de retours partiels, selon les processus de *Rewards Computing* et *Feedback Identification* employés. Enfin, nous analysons nos résultats dans la sous-section 4.2. Nous vérifions que les résultats de précision globale obtenus avec les approches étudiées sont significativement différents en réalisant des tests de Kruskal-Wallis et de rangs signés de Wilcoxon.

4.2 Analyse des Résultats

4.2.1 Tests Statistiques

Nous réalisons des tests de *Kruskal-Wallis* afin de mettre en évidence les inégalités entre les résultats de chaque stratégies (pour chaque algorithme), c.-à-d., nous testons l'hypothèse nulle H_0 : "*Il n'y a pas de différences significatives entre les résultats des différentes stratégies (médianes) lorsqu'appliquées à un même algorithme*". Lorsque les tests de *Kruskal-Wallis* indiquent qu'il existe des différences significatives entre les résultats, nous réalisons des tests de rangs

signés de *Wilcoxon* deux à deux sur la précision globale, c.-à-d., nous testons l'hypothèse nulle H_0 : " *Il n'y a pas de différences significatives entre chaque paire de stratégies appliquées à un même algorithme*".

4.2.2 Synthèse des Résultats

Dans cette partie, nous présentons et analysons les résultats obtenus par les algorithmes avec différentes stratégies. Pour des raisons de clarté, nous nous limitons, dans cet article, à une synthèse de nos principales observations.

		RSASM	Jester	MovieLens
Algorithme	**Stratégie**	$Acc(T)$	$Acc(T)$	$Acc(T)$
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** $\pm_{0,003}$
UCB1	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 $\pm_{0,002}$

Tableau 2 – Résultats avec vecteurs complets de retours utilisateur, meilleurs résultats en gras, seconds en italique

Vecteurs complets : Le Tableau 2 présente les résultats de précision globale obtenus par chaque algorithme sur chaque jeu de données, en fonction du processus de *Reward Computing* employé. Ces résultats sont presque tous significativement différents (p-value < 0.05), la majorité des différences non significatives étant observées entre les résultats obtenus par les stratégies *FV-BSB-/* et *FV-B-/*, extraites de la littérature. Par ailleurs, nous observons que parmi les méthodes étudiées, notre approche *FV-BUSBC-/* est celle présentant le plus de différences significatives. Nos résultats indiquent que *UCB1* et *UCB2* obtiennent leur meilleure précision globale, sur chaque jeu de données, lors de son utilisation. De plus, la Figure 1 confirme que *BUSBC* permet rapidement et pour tout l'horizon de bien meilleurs résultats que les méthodes concurrentes. Concernant *ϵ-greedy* et *Thompson Sampling*, leurs meilleurs résultats sont obtenus avec le processus de *Reward Computing SB* (stratégie *FV-SB-/*). Il est néanmoins intéressant de noter que pour ces algorithmes, sur chaque jeu de données, *BUSBC* correspond au 2nd meilleur choix malgré le fait que cette approche n'est pas conçue pour améliorer leurs performances.

Vecteurs partiels : Le Tableau 3 présente les résultats de précision globale obtenus par chaque algorithme sur chaque jeu de données, lors de l'acquisition de $\psi = 2$ retours par itération, en fonction des processus de *Reward Computing* et de *Feedback Identification* appliqués. Dans ce cadre partiel, les différences significatives entre les processus de *Reward Computing* étudiés sont confirmés (p-value < 0.05). Cependant, ces différences sont moins importantes du fait

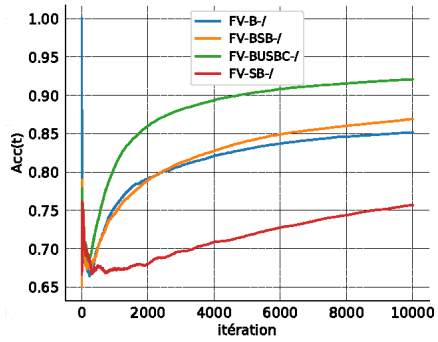
		RSASM	Jester	MovieLens
Algorithme	**Stratégie**	$Acc(T)$	$Acc(T)$	$Acc(T)$
ε-greedy	P-B-OE	0,504 $\pm_{0,037}$	0,418 $\pm_{0,016}$	0,875 $\pm_{0,006}$
	P-B-RE	0,448 $\pm_{0,042}$	0,397 $\pm_{0,014}$	0,860 $\pm_{0,007}$
	P-BSB-OE	0,480 $\pm_{0,042}$	**0,429** $\pm_{0,009}$	0,871 $\pm_{0,008}$
	P-BSB-RE	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** $\pm_{0,003}$
UCB1	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 $\pm_{0,005}$

Tableau 3 – Résultats avec vecteurs partiels de retours utilisateur, meilleurs résultats en gras, seconds en italique

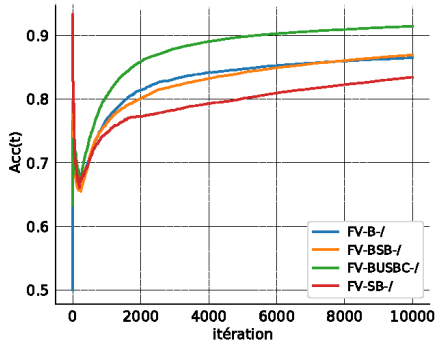
	Stratégie Optimale	
Algorithme	**FV**	**P**
ε-greedy	FV-SB-/	P-SB-RE
TS	FV-SB-/	P-SB-RE
UCB1	FV-BUSBC-/	P-B-OE
UCB2	FV-BUSBC-/	P-B-OE

Tableau 4 – Identification des combinaisons optimales avec vecteurs de retours utilisateur complets et partiels

de la réduction du nombre de retours utilisateur, menant à davantage de similarités dans les comportements des approches étudiées. Concernant les processus de *Feedback Identification*, nous observons également un impact sur la précision globale des algorithmes *COM-MAB*. Dans nos expériences, la plupart des différences observées sont statistiquement significatives, avec *OE* offrant de meilleures performances que *RE* dans la plupart des cas. Nous pouvons noter que le processus de *Rewards Computing* Bandit (stratégies *P-B-OE et P-B-RE*) présente des résultats plus compétitifs lors de l'emploi de vecteurs partiels, devenant le meilleur choix pour l'algorithme *UCB1* lorsque employé avec *OE* dans ce cadre. Ce constat s'explique notamment par la réduction du nombre de bras insatisfaisants percevant une récompense. Cependant, nous notons que sur Jester et RSASM, la précision globale obtenue par *UCB1* avec

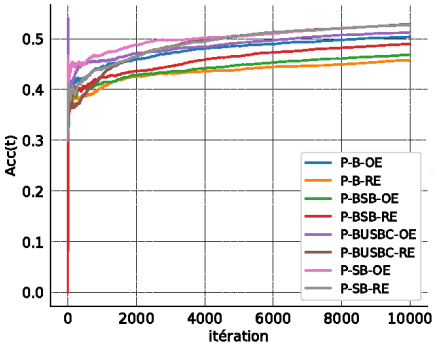


(a) UCB1

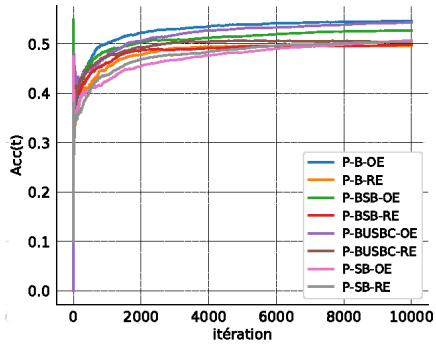


(b) UCB2

FIGURE 1 – Évolution de la précision globale sur MovieLens avec des vecteurs de retours utilisateur complets



(a) ϵ-Greedy



(b) UCB1

FIGURE 2 – Évolution de la précision globale sur RSASM avec des vecteurs de retours utilisateur partiels

P-BUSBC-OE devient équivalente à celle obtenue avec *P-B-OE* (cf. Figure 2b). Un point important à noter est que dans ce cadre, les performances des processus de *Rewards Computing* sont 1) restreintes par le faible nombre de retours perçus ; 2) influencées par le processus de *Feedback Identification* employé. Ainsi, pour un plus grand horizon ou lorsque davantage de retours utilisateur sont perçus, *P-BUSBC-OE* pourrait être un meilleur choix. Par ailleurs, en observant les résultats indépendamment du processus de *Feedback Identification* employé, nous pouvons constater que *BUSBC* reste en réalité un meilleur choix pour *UCB1* sur chaque jeu de données.

Discussion. Le Tableau 4 expose les stratégies optimales pour chaque algorithme *COM-MAB* selon l'exhaustivité des retours utilisateurs. Nous observons que, généralement, la stratégie optimale reste la même sur l'horizon. Cependant, des changements de rangs entre stratégies concurrentes peuvent être observés (voir figure 2a : *P-BUSBC-RE*). Globalement, nous notons que lorsque la stratégie optimale pour un algorithme *COM-MAB* est inconnue, les approches *BUSBC* et *OE* constituent de meilleurs choix.

Bien que la plupart des différences entre les résultats de précision globale obtenus soient significatives, certaines ne le sont pas. Ces situations résultent notamment de deux faits : a) les processus de *Reward Computing* étudiés ont des comportements proches ; b) les jeux de données présentent des biais. De plus, l'impact de ces faits augmente en proportion de la réduction du nombre de retours utilisateur exploités à chaque itération. Enfin, avec des vecteurs de retours utilisateur partiels, nous observons un certain nombre de différences non significatives entre les résultats obtenus par l'application de stratégies de composition totalement différentes (voir Figure 2b, *P-SB-OE* et *P-BUSBC-RE*). Ces observations illustrent que l'association des processus composant la stratégie impacte autant la précision globale que leur ajustement indépendant.

5 Conclusions et Perspectives

Dans cet article, nous avons établi que, pour les algorithmes *COM-MAB*, toute stratégie de prise en compte du retour utilisateur pouvait être définie au travers d'un modèle générique composé de trois processus. Nous avons également montré que la modification d'au moins un de ces processus impacte significativement la précision globale d'un algorithme *COM-MAB*. Ainsi il est possible d'améliorer les performances d'un algorithme par l'ajustement de sa stratégie. Nous avons proposé *Bandit under Semi-Bandit Conditions* (*BUSBC*), un nouveau processus de *Reward Computing*, qui accroît la précision globale des algorithmes de type *UCB*. De plus, notre analyse empirique illustre que, concernant le processus *Feedback Identification*, l'approche *OE* surpasse généralement *RE*, bien que cette seconde méthode décrive le procédé le plus répandu dans la littérature.

La collecte de retours utilisateur et leur usage reste un

défi dans le domaine de l'apprentissage automatisé. Dans la mesure où la stratégie optimale peut varier au cours du temps pour un même algorithme *COM-MAB*, une perspective particulièrement intéressante serait l'implémentation d'une approche portfolio pour sélectionner dynamiquement différentes variantes de chacun des processus durant l'exploitation. Sur le plan industriel, il semble également pertinent d'employer simultanément plusieurs stratégies. Par exemple, un algorithme *COM-MAB* pourrait employer une stratégie de type "Bandits en Cascades" pour acquérir un premier niveau de connaissances avec un minimum de contraintes pour l'utilisateur ainsi qu'une stratégie considérant un vecteur partiel de retours utilisateur pour enrichir cette connaissance lorsque l'utilisateur est disposé à fournir davantage d'informations.
Nous sommes convaincus que des approches de ce type ouvriront la voie à une nouvelle génération de systèmes de recommandation, plus adaptés aux attentes des utilisateurs et dotés d'un apprentissage plus efficace, capables de mieux répondre à de nombreuses problématiques rencontrées dans les applications réelles.

Remerciements

Ces travaux ont été réalisés avec le soutien de l'Association Nationale de la Recherche et de la Technologie (ANRT). Les auteurs tiennent également à remercier les relecteurs anonymes pour leurs pertinents commentaires.

Références

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *COLT*, 2012.

[2] Venkatachalam Anantharam, Pravin Varaiya, and Jean Walrand. Asymptotically efficient allocation rules for the multiarmed bandit problem with multiple plays-part i : I.i.d. rewards. *IEEE Transactions on Automatic Control*, 1987.

[3] Jean-Yves Audibert, Sebastien Bubeck, and Gabor Lugosi. Minimax policies for combinatorial prediction games. *COLT*, 2011.

[4] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *JMLR*, 3, 2002.

[5] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino : The adversarial multi-armed bandit problem. *IEEE 36th Annual Foundations of Computer Science*, 1995.

[6] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing - SICOMP*, 32(1), 2002.

[7] Djallel Bouneffouf and Irina Rish. A survey on practical applications of multi-armed and contextual bandits. *ARXIV*, 2019.

[8] Wei Chen, Yajun Wang, and Yang Yuan. Combinatorial multi-armed bandit : General framework and applications. In *International Conference on Machine Learning - ICML*, 2013.

[9] Richard Combes, Mohammad Sadegh Talebi Mazraeh Shahi, Alexandre Proutiere, and Marc Lelarge. Combinatorial bandits revisited. In *NIPS*. Curran Associates, Inc., 2015.

[10] Nicolas Gutowski. *Context-aware recommendation systems for cultural events recommendation in Smart Cities*. PhD thesis, Université d'Angers, Angers, France, 2019.

[11] Shinji Ito, Daisuke Hatano, Hanna Sumita, Kei Takemura, Takuro Fukunaga, Naonori Kakimura, and Ken-Ichi Kawarabayashi. Improved regret bounds for bandit combinatorial optimization. In *NIPS*. Curran Associates, Inc., 2019.

[12] Branislav Kveton, Zheng Wen, Azin Ashkan, and Csaba Szepesvari. Combinatorial cascading bandits. In *NIPS*. Curran Associates, Inc., 2015.

[13] Paul Lagrée, Claire Vernade, and Olivier Cappé. Multiple-play bandits in the position-based model. In *NIPS*, 2016.

[14] Alexandre Letard, Tassadit Amghar, Olivier Camp, and Nicolas Gutowski. Bandit et semi-bandit avec retour partiel : Une stratégie d'optimisation du retour utilisateur. In *APIA*, 2020.

[15] Alexandre Letard, Tassadit Amghar, Olivier Camp, and Nicolas Gutowski. Partial bandit and semi-bandit : Making the most out of scarce users' feedback. In *ICTAI*, 2020.

[16] Shuai Li, Baoxiang Wang, Shengyu Zhang, and Wei Chen. Contextual combinatorial cascading bandits. In *ICML*, 2016.

[17] Alexander Luedtke, Emilie Kaufmann, and Antoine Chambaz. Asymptotically optimal algorithms for multiple play bandits with partial feedback. *ARXIV*, 2016.

[18] Gergely Neu. First-order regret bounds for combinatorial semi-bandits. In *COLT*, Proceedings of Machine Learning Research. PMLR, 2015.

[19] Herbert Robbins. Some aspects of the sequential design of experiments. *Bull. of the AMS*, 1952.

[20] Aadirupa Saha and Aditya Gopalan. Combinatorial bandits with relative feedback. In *NIPS*, pages 985–995. Curran Associates, Inc., 2019.

[21] Karthik Abinav Sankararaman. Semi-bandit feedback : A survey of results. In *CoRR*, 2016.

[22] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning : An introduction*, volume 1. MIT press Cambridge, 1998.