Algorithmes de bandits pour la recommandation à tirages multiples

Jonathan Louëdec [1,2], **Max Chevalier** [2], **Aurélien Garivier** [1], **Josiane Mothe** [2]

1. *Institut de Mathématiques de Toulouse*
   UMR 5505, CNRS, Université de Toulouse, France
   prenom.nom@math.univ-toulouse.fr

2. *Institut de Recherche en Informatique de Toulouse*
   UMR 5219, Université de Toulouse, France
   prenom.nom@irit.fr

RÉSUMÉ. *Les systèmes de recommandation (SR) à tirages multiples font référence aux SR recommandant plusieurs objets en même temps aux utilisateurs. La plupart des SR s'appuient sur des modèles d'apprentissage afin de décider les objets à recommander. Parmi ces modèles, les algorithmes de bandits offrent l'avantage d'apprendre tout en exploitant les éléments déjà appris. Les approches actuelles utilisent autant d'instances d'un algorithme de bandits que le nombre d'objets que doit recommander le SR. Nous proposons au contraire de gérer l'ensemble des recommandations par une seule instance d'un algorithme de bandits pour rendre l'apprentissage plus efficace. Nous montrons sur deux jeux de données de références (Movielens et Jester) que notre méthode, MPB (Multiple Plays Bandit), obtient des temps d'apprentissage jusqu'à treize fois plus rapides tout en obtenant des taux de clics équivalents. Nous montrons également que le choix de l'algorithme de bandits utilisé influence l'amélioration obtenue.*

ABSTRACT. *The multiple-play recommender systems (RS) are RS which recommend several items to the users. RS are based on learning models in order to choose the items to recommend. Among these models, the bandit algorithms offer the advantage to learn and exploite the learnt elements at the same time. Current approaches require running as many instances of a bandit algorithm as there are items to recommend. As opposed to that, we handle all recommendations simultaneously, by a single instance of a bandit algorithm. We show on two benchmark datasets (Movielens and Jester) that our method, MPB (Multiple Plays Bandit), obtains a learning rate about thirteen times faster while obtaining equivalent click-through rates. We also show that the choice of the bandit algorithm used impacts the level of improvement.*

MOTS-CLÉS : *Recherche d'Information, Systèmes de Recommandation, Algorithme de Bandits.*

KEYWORDS: *Information Retrieval, Recommender Systems, Bandit Algorithm.*

DOI:10.3166/DN.x.y.1-21 © 2015 Lavoisier

1. Introduction

Les systèmes de recommandation (SR) actuels sont utilisés dans de nombreuses applications, particulièrement en ligne : il peut s'agir de recommander des objets marchants comme des films, articles, musiques ou des contenus comme des pages web, des traitements médicaux, ... De nombreux SR recommandent à chaque instant plusieurs objets à un utilisateur simultanément; ils sont qualifiés de SR à tirages multiples.

L'utilisateur peut choisir de sélectionner un ou plusieurs objets parmi ceux qui lui ont été recommandés. Les recommandations sélectionnées sont généralement considérées comme pertinentes. L'utilisateur peut ne pas sélectionner de recommandation dans la liste qui lui est proposée. Il s'agit d'un *abandon* qui correspond donc au cas où aucune des recommandations n'est sélectionnée par l'utilisateur. Dans le but d'optimiser les performances d'un SR, nous considérons dans cet article le problème de la minimisation de l'abandon. Ce problème peut également être vu comme la maximisation du nombre de fois où les utilisateurs sélectionnent au moins un objet parmi ceux recommandés.

Afin d'améliorer la pertinence des recommandations pour un utilisateur, un SR peut considérer l'historique des interactions passées avec les utilisateurs. Lorsque toutes les données sont connues, il est alors possible d'estimer la pertinence des objets, il s'agit alors du cadre de l'apprentissage supervisé (Hastie *et al*., 2009). Ce n'est pas un cadre réaliste pour un SR : de nouveaux utilisateurs et de nouveaux objets apparaissent continuellement. Par ailleurs il est souhaitable de pouvoir apprendre continuellement tout en recommandant des objets. Un tel environnement s'inscrit dans le cadre de ce que l'on appelle l'*apprentissage par renforcement* (Sutton, Barto, 1999). Il s'agit d'implémenter une stratégie pour explorer de nouvelles données d'apprentissage (les interactions des utilisateurs), tout en assurant que les résultats puissent être directement exploités. Ce problème est connu sous le nom de "dilemme exploration/exploitation". Les algorithmes de bandits sont connus pour offrir des solutions à ce dilemme (Bubeck, Cesa-Bianchi, 2012). Les algorithmes de bandits sont généralement conçus pour recommander un seul objet à chaque instant : ce sont des algorithmes à tirages simples. Plusieurs familles d'algorithmes de bandits se distinguent de par la stratégie choisie pour appréhender le "dilemme exploration/exploitation" (voir section 2.1).

Plusieurs travaux proposent des approches pour utiliser les algorithmes de bandits dans le cadre de la recommandation à tirages multiples. Deux de ces approches forment les approches de référence auxquelles nous comparons notre proposition : *Ranked Bandits Algorithm* (*RBA*) (Radlinski *et al*., 2008) et *Independent Bandits Algorithm* (*IBA*) (Kohli *et al*., 2013). Ces approches utilisent autant d'instances d'un algorithme de bandits qu'il y a de recommandations à proposer. À chaque instant chaque instance gère une seule recommandation qui composera la liste finale de recommandation. Dans cet article, au contraire, l'approche proposée, nommée *Multiple Plays Bandit* (*MPB*), permet de gérer simultanément l'ensemble des recommandations avec une seule instance d'un algorithme de bandits. Selon notre approche, l'instance de

bandit est utilisée plusieurs fois à chaque itération. De cette manière l'algorithme de bandits utilisé par l'approche **MPB** va prendre en compte les nouvelles informations sur plusieurs objets à chaque instant, lorsque chaque instance de l'algorithme de bandits utilisé par les approches *RBA* et *IBA* prend en compte uniquement l'information liée à la recommandation effectuée par cette instance.

Par construction, les trois approches évoquées précédemment, *RBA*, *IBA* et *MPB*, ne visent pas exactement le même objectif. Les approches *RBA* et *MPB* tentent d'optimiser le nombre de fois où au moins un clic est observé, peu importe le nombre de clics, tandis que l'approche *IBA* tente d'optimiser le nombre total de clics accumulés. L'objectif visé par les approches *RBA* et *MPB* est plus adapté dans le cadre de la minimisation de l'abandon (voir fin section 3).

Afin de comparer au mieux notre approche (*MPB*) aux deux approches de l'état de l'art (*RBA* et *IBA*), nous utilisons les mêmes algorithmes de bandits que dans l'expérimentation de l'article de Kohli *et al.* (2013), à savoir les algorithmes *UCB1* et ϵ-*greedy*. Nous montrons dans cet article que notre proposition, l'approche *MPB*, améliore grandement la vitesse d'apprentissage par rapport aux méthodes de l'état de l'art lorsque ces dernières utilisent l'algorithme de bandits *UCB1* (jusqu'à treize fois plus rapide). Cependant notre approche ne permet pas d'améliorer les performances en terme de temps d'apprentissage lorsque l'algorithme de bandits utilisé est ϵ-*greedy*, mais elle permet tout de même d'obtenir des taux de clics équivalents. L'utilisation des approches *RBA* et *IBA* avec l'algorithme *UCB1* était nettement moins efficace en terme de taux de clics et de temps d'apprentissage que lorsque ces approches utilisent l'algorithme ϵ-*greedy*. Néanmoins l'amélioration apportée par l'approche *MPB* lorsque l'algorithme *UCB1* permet d'obtenir un compromis entre le temps d'apprentissage et le taux de clics rivalisant avec les approches *RBA* et *IBA* utilisant l'algorithme ϵ-*greedy*. Expérimentalement, nous avons pu déceler des différences de comportement entre les approches *RBA* et *IBA* d'un côté et l'approche *MPB* de l'autre permettant de comprendre en partie ces phénomènes. Les expérimentations sont réalisées sur deux jeux de données de référence : Jester et MovieLens.

La suite de l'article est organisée de la manière suivante : dans la section 2, nous présentons et classons en deux catégories les travaux utilisant des algorithmes de bandits dans un cadre de recommandation d'information. Dans la section 3, nous formalisons le problème de recommandation et définissons les deux objectifs différents présentés ci-dessus. Dans la section 4 nous présentons plus en détail les deux approches de référence, *RBA* et *IBA*, ainsi que l'approche proposée, *MPB*. L'évaluation et les résultats obtenus sont présentés dans la section 5. Enfin, nous indiquons des pistes pour nos travaux futurs dans la conclusion de cet article.

2. Etat de l'art

2.1. Algorithmes de bandits

Les algorithmes de bandits sont connus pour proposer des solutions efficaces au dilemme exploration/exploitation. Considérons un agent, il tire à chaque instant un bras parmi *K*, et obtient une récompense. Nous nous plaçons dans un cadre où le comportement des bras ne varie pas dans le temps. L'objectif de l'agent est de maximiser la somme des récompenses qu'il obtient, Il doit pour cela trouver le bras optimal, c'est-à-dire celui qui a l'espérance de récompense la plus forte. La performance d'un algorithme de bandits est mesurée par le regret : il s'agit de la différence à l'instant t entre la somme des récompenses obtenues en tirant le bras optimal (inconnu a priori), et la somme des récompenses obtenues avec la stratégie utilisée. le but est de faire en sorte que le regret croisse le plus lentement possible avec le nombre t de tirages. Si ce cadre peut paraître peu réaliste en recommandation, il est important d'être capable d'obtenir des stratégies amenant à de fortes garanties théoriques dans ce cadre, afin de pouvoir en proposer des extensions prenant en compte les spécificités de la recommandation.

2.1.1. Stratégies à tirages simples

L'algorithme ϵ**-greedy** et ses différentes variantes sont probablement les stratégies les plus simples de la littérature sur les problèmes de bandits. L'algorithme ϵ*-greedy* (Watkins, 1989) choisit d'ajouter une part d'aléatoire dans le choix du bras à tirer. A chaque instant, avec une probabilité ϵ, l'exploration sera réalisée en tirant un bras tiré aléatoirement selon une loi uniforme. Avec une probabilité $1 - \epsilon$, le bras ayant l'espérance de récompense estimée la plus forte est tiré. Cette idée peut mener à un regret sous-linéaire, si comme dans la variante ϵ*-decreasing* (Cesa-Bianchi, Fischer, 1998) on laisse le paramètre epsilon décroître au fur et à mesure des tirages.

Afin de canaliser l'exploration vers les bras qui en valent vraiment la peine, les stratégies de type **UCB (Upper Confidence Bound)** ont été développées. Elles utilisent non pas des estimateurs, mais des bornes de confiance supérieures pour l'espérance de chaque bras comme indices de qualité. La stratégie est dite "optimiste dans l'incertain", c'est-à-dire que le bras ayant potentiellement l'espérance la plus forte est recommandé. Ce choix peut se faire au détriment d'un bras ayant une meilleure espérance estimée, mais dont la borne de confiance supérieure est plus faible, car il a été beaucoup tiré par le passé. Les stratégies de type UCB se distinguent principalement par la manière dont la borne de confiance supérieure est calculée. *UCB1* (Auer, Cesa-Bianchi, Fischer, 2002) utilise uniquement une borne calculée en se basant sur le nombre d'actions effectués par l'agent depuis le début et le nombre de fois où le bras en question a été tiré. *UCB-V* (Audibert *et al*., 2009) permet de prendre en compte la variance des résultats obtenus pour chaque bras. *KLUCB* (Garivier, Cappé, 2011) utilise la divergence de Kullback-Leibler pour affiner le calcul de la borne supérieure de confiance, lui permettant d'obtenir de meilleures performances en terme de regret que les deux versions précédentes.

Plusieurs autres stratégies peuvent également être citées : celles de type **softmax**, l'algorithme *Exp3* (Auer, Cesa-Bianchi, Freund, Schapire, 2002) par exemple, où le bras tiré est choisi selon une loi de probabilité privilégiant les bras ayant les espérances observées les plus fortes. Un algorithme ancien d'inspiration bayésienne est aujourd'hui considéré comme l'un des plus prometteurs à la fois d'un point de vue théorique et d'un point de vue pratique : **Thompson sampling** (Thompson, 1933). Cet algorithme va à chaque instant recommander le bras avec l'espérance, tiré aléatoirement selon la loi de probabilité associée à ce bras. La loi de probabilité est ensuite mise à jour en fonction du retour utilisateur obtenu. Si l'algorithme a plus de 80 ans, son étude récente permet d'obtenir des garanties théoriques fortes, et en pratique, de bonnes performances en terme de regret (Agrawal, Goyal, 2012 ; Kaufmann *et al.*, 2013).

Dans certains domaines, comme la recommandation, ces approches peuvent paraitre trop naïves pour obtenir de bonnes performances, car elles ne prennent pas en compte les spécificités propres à ce domaine. C'est pourquoi récemment des extensions de ces algorithmes ont été proposées. Il est possible de se placer dans un cadre où l'agent doit tirer plusieurs bras à chaque instant, les stratégies à tirages multiples ont été développées en ce sens. D'autres travaux proposent des algorithmes capables de prendre en compte un contexte. En effet il parait peu réaliste en recommandation par exemple de dire que l'espérance en récompense est la même quel que soit l'utilisateur.

2.1.2. Stratégies à tirages multiples

Lorsque plusieurs bras sont tirés à chaque instant (tirages multiples), il convient non plus de trouver le bras optimal, mais de trouver la combinaison de bras avec l'espérance de récompense la plus forte. Les algorithmes présentés précédemment sont conçus pour tirer uniquement un bras à chaque instant. De récents travaux en apprentissage par renforcement proposent des algorithmes de bandits capables de tirer plusieurs bras à chaque instant, c'est le cadre de la recommandation à tirages multiples. *Exp3_M* (Uchiya *et al.*, 2010 ; Bubeck, Cesa-Bianchi, 2012) est dérivé de l'algorithme de bandits de type *softmax* à tirages simples : *Exp3*. Dans l'article (Louedec *et al.*, 2015), nous montrons que l'utilisation d'*Exp3_M* permet d'obtenir de meilleurs résultats par rapport à l'utilisation de *Exp3* via les approches *RBA* et *IBA* présentées plus en détails dans la suite de cet article. Plus récemment, une version à tirages multiples de l'approche *Thompson sampling* a également été proposée, l'algorithme *Thompson Sampling Multiple-Play* (Komiyama *et al.*, 2015).

2.1.3. Stratégies contextuelles

En recommandation notamment, le contexte, utilisateur ou objet, peut permettre d'optimiser l'espérance en récompense obtenue, notamment en personnalisant les recommandations proposées. Les algorithmes de bandits contextuels ont été conçus pour aborder cet aspect. Avant de tirer un bras, l'agent dispose à présent d'une information contextuelle. Elle peut être de deux types : soit l'agent a à sa disposition une infor-

mation qui varie à chaque instant et la récompense associée à chaque bras dépend de cette information, soit l'agent dispose d'une information fixe pour chaque bras.

La plupart de ces stratégies sont conçues pour tirer un seul bras à chaque instant, comme par exemple l'algorithme de bandits stochastique linéaire *LinUCB* (Li *et al*., 2010 ; Chu *et al*., 2011). Tandis que les algorithmes de bandits présentés précédemment estiment l'espérance de récompense de chaque bras uniquement en les testant, l'approche *LinUCB* considère que l'espérance de chaque bras est une combinaison linéaire entre un vecteur contenant l'information disponible a priori et un vecteur de coefficients inconnu, estimé en fonction des récompenses et des vecteurs d'informations associés passés.

En 2015, Tang *et al*. (2015) propose l'algorithme *Online BootStrap Bandit* qui, tout comme l'approche *LinUCB*, va estimer l'espérance de chaque bras à l'aide d'une fonction prenant en compte le vecteur contenant l'information disponible a priori et un vecteur de coefficients inconnu. Dans l'article de Tang *et al*. (2015), au lieu d'effectuer une combinaison linéaire, les auteurs utilisent un modèle de régression logistique. Les estimateurs sont construits en utilisant des tirages bootstrap en ligne (Oza, 2005), notamment pour compenser le manque d'information lors des premières actions de l'agent. Les méthodes de bootstrap ont pour objectif de créer un jeu de données utilisables pour l'apprentissage en effectuant un tirage avec remise parmi les données à notre disposition. De plus les méthodes de bootstrap en ligne qui, au lieu de refaire le tirage à chaque instant, se contente de le compléter, permettent d'obtenir des temps de réponse bien plus faibles qu'avec une approche de bootstrap classique. Un autre avantage de l'algorithme *Online Bootstrap Bandit* est de n'avoir aucun paramètre affectant la répartition entre l'exploration et l'exploitation, l'algorithme adapte ainsi cette répartition en fonction du besoin.

Si les deux algorithmes précédents sont conçus pour tirer un seul bras à chaque instant, ils existent des travaux abordant le cas du tirage multiple. Par exemple, Yue et Guestrin (2011) proposent l'algorithme *LSBGreedy*. Il estime dans un premier temps l'espérance de récompense de chaque bras en utilisant le vecteur contenant l'information disponible a priori et un vecteur résultat de la régression linéaire entre les récompenses obtenues et les vecteurs contenant l'information a priori liés. Dans un second temps, la combinaison regroupant plusieurs bras est construite. Pour cela l'approche *LSBGreedy* calcule le gain, en terme de diversité, apporté par l'ajout de chaque bras à la combinaison en construction. Une borne de confiance supérieure est alors calculée pour chacun des gains estimés, et le bras dont la borne associée est la plus forte est ajoutée à la combinaison en construction.

Certains algorithmes de bandits proposent de tirer les bras de manière ordonnée selon l'espérance estimée des bras par rapport à un utilisateur. Ailon *et al*. (2014) propose l'algorithme *BanditRank* pour aborder la problématique d'ordonnancement. Pour cela l'algorithme soumet à chaque instant une permutation de l'ensemble des bras disponibles, dont l'objectif principal est de trier les bras par ordre de pertinence en fonction des récompenses obtenues.

2.2. Bandits et recommandation à tirages multiples

De nombreux domaines d'applications sont possibles pour ces stratégies : les études cliniques, intelligence artificielle, gestion des tâches, "Yield management" (gestion des prix des billets d'avion par exemple), etc. Dans cet article, nous nous intéressons à l'application de ces stratégies dans le domaine de la recommandation. Ainsi les bras deviennent des objets, l'agent devient un système de recommandation, les récompenses deviennent des clics utilisateurs, et les vecteurs contenant l'information a priori contiennent des informations relatives aux utilisateurs et/ou aux objets. L'objectif est de maximiser le nombre de clics obtenus.

Si certains algorithmes précédemment cités sont facilement adaptables pour être utilisés en recommandation à tirages multiples (voir section 2.1.2), des approches sont conçues pour pouvoir utiliser les algorithmes à tirages simples dans ce cadre. Les algorithmes contextuels sont utilisables lorsque de nombreuses informations sur les utilisateurs et les objets sont disponibles. Cependant, ce n'est pas toujours le cas, particulièrement lorsque le système de recommandation interagit avec un nouvel utilisateur (départ à froid côté utilisateur). Cela induit en particulier que les recommandations ne peuvent pas être personnalisées et qu'une solution satisfaisant la majorité des utilisateurs semble plus adaptée.

Les premières approches de recommandation utilisant des algorithmes de bandits sont antérieures aux versions à tirages multiples de la littérature statistique. Au delà de l'aspect contextuel, se posait la question de comment utiliser ces algorithmes conçus pour recommander un seul objet à chaque instant, lorsque plusieurs recommandations sont nécessaires. Dans ce cadre, Radlinski *et al*. (2008) ont développé l'approche *Ranked Bandits Algorithm* (*RBA*) qui utilise autant d'instances d'un algorithme de bandits à tirages simples que de recommandations à soumettre. Ces instances peuvent être des algorithmes tels que *UCB1* (Auer, Cesa-Bianchi, Fischer, 2002), ϵ*-greedy* (Sutton, Barto, 1999) ou encore *Exp3* (Auer, Cesa-Bianchi, Freund, Schapire, 2002). Plus récemment Kohli *et al*. (2013) ont créé l'approche *Independent Bandits Algorithm* (*IBA*). La principale différence avec l'approche *RBA* est la manière dont l'action de l'utilisateur est prise en compte (voir section 4).

L'approche *Combinatorial Multi-Armed Bandit (CMAB)* proposée par W. Chen *et al*. (2013) utilise un seul algorithme de bandits pour effectuer plusieurs recommandations. L'approche *Multiple Plays Bandit (MPB)* que nous proposons dans cet article repose sur la même idée, mais se démarque sur la manière dont est considéré le retour utilisateur. Alors que l'approche *CMAB* optimise le nombre total de clics à chaque instant, l'approche *MPB* tend à optimiser le nombre de fois où au moins un clic est observé. Cette différence permet d'assurer une certaine diversité dans les résultats.

Notre approche étant modélisée dans un cadre où aucune information sur les utilisateurs n'est disponible, elle ne prend pas en compte l'aspect contextuel des SR. Son objectif est plutôt de trouver une solution satisfaisant la majorité des utilisateurs, tout comme les méthodes *RBA* et *IBA*. C'est pourquoi lors des expérimentations ces approches serviront de références. Ces approches ainsi que l'approche *MPB* présentées

dans cet article utilisent les algorithmes de bandits à tirages simples pour recommander plusieurs éléments. Afin de les comparer, nous utiliserons les algorithmes *UCB1* et ϵ-*greedy*.

3. Modèles de bandits pour la recommandation

3.1. Formalisme

Dans cet article une recommandation cliquée par l'utilisateur est considérée comme pertinente comme cela est généralement le cas en recommandation. La principale source d'information pour évaluer l'efficacité d'un SR est le retour utilisateur (Ricci *et al.*, 2011).

Considérons une collection de k objets notés I_i avec $i \in \{1, \ldots, K\}$. À chaque instant t, m objets sont recommandés à un utilisateur. Si cet utilisateur clique sur au moins un objet, nous obtenons une récompense de 1, sinon la récompense est de 0. L'objectif est de minimiser la fraction de 0 obtenus, autrement dit de minimiser la proportion d'abandon. P_m^K est l'ensemble des combinaisons de m objets qu'il est possible d'obtenir avec K objets. Un utilisateur est représenté par un vecteur de pertinence $X = \{0, 1\}^K$ où $X_i = 1$ si l'objet i est cliqué par l'utilisateur. À chaque instant t, l'utilisateur est représenté par X_t et une combinaison $A_t \in P_m^K$ lui est recommandée.

Z_t est la récompense obtenue pour la combinaison A_t associée au vecteur de pertinence X_t. Il est défini par :

$$Z_t = \max_{i \in A_t} X_{i,t}$$

Chaque composant $i \in \{1, \ldots, K\}$ du vecteur X suit une distribution de Bernoulli de paramètre p_i inconnu. La probabilité p_i peut être estimée à chaque instant t par :

$$\hat{p}_i(t) = \frac{1}{N_i(t)} \sum_{t \,:\, i \in A_t} X_{i,t} \quad \text{avec} \quad N_i(t) = \sum_{t \,:\, i \in A_t} 1$$

La proportion d'utilisateurs qui considèrent comme pertinent au moins un objet de A_t est $E[Z_t]$, l'espérance de la variable Z_t. Maximiser $E[Z_t]$ est équivalent à minimiser la proportion d'abandon. La ou les combinaisons optimales A^* sont donc la ou les combinaisons qui provoquent au moins un clic chez un maximum d'utilisateurs. A^* est définie par :

$$A^* = \operatorname*{argmax}_{A \in P_m^K} E[Z]$$

Le but d'un SR dans le contexte temps réel peut être défini comme la minimisation de la différence entre $\sum_{t=1}^{T} Z^*$ (la somme des récompenses obtenues en utilisant une combinaison optimale A^*) et $\sum_{t=1}^{T} Z_t$ (la somme des récompenses obtenues avec la

combinaison recommandée par l'approche utilisée). Cette différence est appelée le regret cumulé :

$$R(T) = T \times E[Z^*] - \sum_{t=1}^{T} E[Z_t]$$

3.2. Solutions sous-optimales

Trouver une combinaison optimale A^* est un problème NP-complet (Radlinski *et al*., 2008). C'est pourquoi une combinaison sous-optimale mais plus facile à atteindre semble plus appropriée au contexte temps réel. Kohli *et al*. (2013) utilisent deux autres combinaisons : la combinaison indépendante et la combinaison diversifiée.

La combinaison indépendante est imaginée selon le principe de classement par probabilité proposé par Robertson (1977). Elle est définie comme la combinaison des m objets les plus cliqués.

$$A^{\text{indépendante}} = \underset{A \in P_m^K}{\operatorname{argmax}} \sum_{i \in A} E[X_i]$$

Cette combinaison est visée par l'approche *IBA* qui est conçue pour maximiser :

$$Z_t^{\text{indépendante}} = \sum_{i \in A_t} X_{i,t}$$

La notion de diversité est essentielle en recommandation, elle n'est pourtant pas prise en compte avec la combinaison indépendante (H. Chen, Karger, 2006). Prenons un exemple de recommandation de films : admettons que les films les plus populaires sont ceux l'hexalogie "Star Wars". Dans ce cadre, il y a des chances que ces 6 films soient aimés par des personnes aux goûts similaires, qui représentent la majorité des utilisateurs. Toutefois pour tous les utilisateurs qui ont des goûts différents, aucune de ces 6 propositions ne les satisfera. Une combinaison composée de films de genre différents est plus appropriée, particulièrement dans le cas de la minimisation de l'abandon et/ou dans le cas où peu d'informations sur les objets/utilisateurs sont disponibles.

Pour prendre en compte la notion de diversité, une notion de combinaison diversifiée a été définie (Radlinski *et al*., 2008). Il s'agit de la combinaison qui propose l'objet le plus populaire en première position, et ensuite les objets les plus populaires lorsque les objets aux positions précédentes ne sont pas cliqués :

$$A_1^{\text{diversifiée}} = \underset{i \in K}{\operatorname{argmax}} E[X_i] \quad \text{et}$$

$$A_k^{\text{diversifiée}} = \underset{i \in K / \left\{A_{1,\ldots,k-1}^{\text{diversifiée}}\right\}}{\operatorname{argmax}} E\left[X_i | X_j = 0 \ \forall j \in \left\{A_{1,\ldots,k-1}^{\text{diversifiée}}\right\}\right]$$

En prenant les objets les plus populaires lorsque les objets aux positions précédentes ne sont pas cliqués, cette combinaison prend en compte la notion de diversité.

Il s'agit tout de même d'une combinaison sous-optimale qui peut être différente de la combinaison ayant la plus grande probabilité d'obtenir au moins un clic. En effet la combinaison diversifiée recommande en première position l'objet le plus populaire, cet objet ne fait pas obligatoirement partie de la combinaison optimale A^*.

La combinaison indépendante $A^{\text{indépendante}}$ est considérée comme moins bonne que la combinaison diversifiée $A^{\text{diversifiée}}$ lorsque le but est la minimisation du nombre d'abandon (Radlinski *et al*., 2008).

4. Adaptation des algorithmes à tirages simples pour le cas à tirages multiples

Dans cette section, nous détaillons les deux approches déjà évoquées, *RBA* et *IBA*, et nous les utiliserons comme approches de références. L'approche *RBA* vise la combinaison diversifiée, tandis que l'approche *IBA* vise la combinaison indépendante. Nous présenterons dans un deuxième temps notre proposition, l'approche *MPB*, qui vise la combinaison diversifiée.

4.1. Ranked Bandits Algorithm (RBA)

L'approche *RBA (Algorithme 1)* a été développée par Radlinski *et al*. (2008) et nécessite l'utilisation en parallèle de m instances d'un algorithme de bandits à tirages simples. À chaque instant t, les m objets choisis par les m instances d'algorithme de bandits sont recommandés à l'utilisateur. L'information de chaque instance est mise à jour de la manière suivante : l'instance correspondant au premier objet cliqué obtient une récompense de 1, tandis que tous les autres obtiennent une récompense de 0. De cette manière, la première instance tend à recommander l'objet avec le taux de clics le plus haut. La deuxième instance peut recevoir une récompense de 1 uniquement si le premier objet n'est pas cliqué. Elle tend ainsi à recommander l'objet qui obtient le meilleur taux de clics conditionnellement au fait que l'objet avec le taux de clics le plus haut n'ai pas été cliqué. Et ainsi de suite pour les instances suivantes.

4.2. Independent Bandits Algorithm (IBA)

Plus récemment, l'approche *IBA (Algorithme 2*) a été développée par Kohli *et al*. (2013). Tout comme l'approche *RBA*, elle nécessite l'utilisation en parallèle de m instances d'un algorithme de bandits à tirages simples. La principale différence entre les deux approches est la manière dont le retour utilisateur est pris en compte. Dans l'approche *RBA* seulement le premier objet est considéré comme pertinent, tandis que dans l'approche *IBA* tous les objets cliqués sont considérés comme pertinents. Ce qui induit que la $k^{\text{ième}}$ instance tend à recommander l'objet avec le $k^{\text{ième}}$ taux de clics le plus haut. L'approche *IBA* vise la combinaison indépendante.

La proportion d'abandon est généralement plus grande pour la combinaison indépendante que pour la combinaison diversifiée (cf. section 3), ce qui induit que l'approche *RBA* est, dans la plupart des cas, meilleure que l'approche *IBA* sur le très long

Algorithme 1 : Ranked Bandits Algorithm (*RBA*)

BTS_i : instance d'un algorithme de bandits à tirages simples pour la recommandation en position i
pour $t = 1, \ldots, T$ **faire**
  pour $i = 1, \ldots, m$ **faire**
    $a_i \leftarrow$ SélectionnerObjet(BTS_i, K)
    si $a_i \in \{a_1, \ldots, a_{i-1}\}$ **alors**
      $a_i \leftarrow$ choix arbitraire parmi les documents non sélectionnés
    fin
  fin
  $A_t \leftarrow \cup_i a_i$
  Recommander A_t à l'utilisateur, récupérer le retour utilisateur X_t
  pour $i = 1, \ldots, m$ **faire**
    Retour utilisateur :

$$z_i = \begin{cases} 1 & \text{si l'objet } a_i \text{ est le premier cliqué} \\ 0 & \text{sinon} \end{cases}$$

    MiseAJour(BTS_i, z_i)
  fin
fin

Algorithme 2 : Independent Bandits Algorithm (*IBA*)

BTS_i : instance d'un algorithme de bandits à tirages simples pour la recommandation en position i
pour $t = 1, \ldots, T$ **faire**
  pour $i = 1, \ldots, m$ **faire**
    $a_i \leftarrow$ SélectionnerObjet(BTS_i, $K \setminus A_{t,i-1}$)
  fin
  $A_t \leftarrow \cup_i a_i$
  Recommander A_t à l'utilisateur, récupérer le retour utilisateur X_t
  pour $i = 1, \ldots, m$ **faire**
    Retour utilisateur :

$$z_i = \begin{cases} 1 & \text{si l'objet } a_i \text{ est cliqué} \\ 0 & \text{sinon} \end{cases}$$

    MiseAJour(BTS_i, z_i)
  fin
fin

terme (lorsque t est grand). Cependant, Kohli *et al*. (2013) ont montré dans leurs si-

Algorithme 3 : Multiple Plays Bandit

BTS : instance d'un algorithme de bandits à tirages simples pour la recommandation en position i
pour $t = 1, \ldots, T$ **faire**
  pour $i = 1, \ldots, m$ **faire**
    $a_i \leftarrow$ SélectionnerObjet(BTS, $K \backslash A_{t,i-1}$)
  fin
  $A_t \leftarrow \cup_i a_i$
  Recommander A_t à l'utilisateur, récupérer le retour utilisateur X_t
  pour $i = 1, \ldots, m$ **faire**
    Retour utilisateur :

$$z_i = \begin{cases} 1 & \text{si l'objet } a_i \text{ est le premier cliqué} \\ -1 & \text{si } \forall a_j \text{ tel que } j <= i, \text{ aucun } a_j \text{ n'a été cliqué} \\ 0 & \text{sinon} \end{cases}$$

    MiseAJour(BTS, z_i)
  fin
fin

mulations que le temps d'apprentissage nécessaire pour l'approche *IBA* est significativement plus court (la proportion d'abandon diminue donc plus rapidement qu'avec l'approche *RBA*), ce qui est un indicateur important pour la qualité d'un SR, en particulier dans un environnement qui change souvent.

Dans les deux approches précédentes, les m instances d'un algorithme de bandits à tirages simples fonctionnent de façon (quasiment) indépendantes les unes des autres. À chaque instant t, chaque instance apprend uniquement sur un seul objet. Dans ces conditions, les deux approches convergent uniquement lorsque toutes les instances ont convergé, ce qui augmente le regret. Au contraire, une approche permettant d'utiliser une seule instance d'un algorithme de bandits pour gérer l'ensemble des recommandations permettrait d'apprendre sur m objets à chaque instant. On peut attendre que l'apprentissage soit alors plus efficace.

4.3. Multiple Plays Bandit (MPB)

Nous présentons ici l'approche Multiple Plays Bandit (*MPB*) *(Algorithme 3)*. La principale différence avec les approches *RBA* et *IBA* est que nous utilisons une seule instance d'un algorithme de bandits. Pour recommander simultanément une liste de m objets, le premier objet est sélectionné selon la règle utilisée par le bandit. Le second objet est sélectionné parmi l'ensemble des objets disponibles exceptés celui précédemment recommandé, et ainsi de suite pour les recommandations suivantes. Les estimateurs de la pertinence des m objets sont mis à jour, estimateurs gérés par la

même instance d'un algorithme de bandits. De cette manière, m estimateurs seront mis à jour à chaque instant, ces estimateurs de la pertinence d'un objet approcheront plus rapidement les véritables valeurs associées, et ainsi la vitesse d'apprentissage sera améliorée.

Une autre différence importance concerne la façon dont le retour de l'utilisateur est pris en compte. À chaque instant une liste de m recommandations classées par ordre de pertinence estimée décroissant, est proposée à l'utilisateur. Lors de la recommandation, tous les objets non cliqués et qui sont recommandés avant le premier objet cliqué sont pénalisés, avec une récompense négative (-1). De cette manière, les objets qui sont peu cliqués sont vite mis de côté. Le choix de la valeur -1 est arbitraire, l'ajustement de cette valeur sera une piste pour de futurs travaux. Cette méthode permet de prendre en compte plus facilement le vieillissement d'un objet : si un objet était populaire mais ne l'est plus, beaucoup de récompenses négatives seront retournées et l'estimateur de la pertinence lié à cet objet va rapidement décroître. Pour les autres objets, seulement le premier cliqué obtient une récompense positive (1), tandis que les autres obtiennent tous une récompense nulle (0). De cette manière cette approche vise la combinaison diversifiée.

5. Évaluation

5.1. Cadre expérimental

Pour évaluer notre approche et la comparer aux méthodes de l'état de l'art, nous utilisons le cadre expérimental défini par Kohli *et al*. (2013). Les expérimentations font appel aux jeux de données MovieLens-100 et Jester.

Le premier jeu de données, MovieLens-100, contient 943 utilisateurs qui ont noté 100 films. Les notes sont comprises entre 1 (mauvais) et 5 (bon) (si un film n'est pas noté par un utilisateur, la note minimale de 1 lui est affectée). Pour traduire les notes en actions utilisateurs, Kohli *et al*. (2013) ont choisi de fixer un seuil de pertinence. Dans leurs expérimentations, deux valeurs ont été utilisées : 2 et 4. Lorsque le seuil est 2 (resp. 4), tous les films qui ont une note strictement supérieure à 4 sont considérés comme pertinents, c'est-à-dire que l'utilisateur les a cliqués.

Le deuxième jeu de données, Jester, contient 25000 utilisateurs qui ont noté 100 blagues. Les notes sont comprises entre -10 (pas drôle) et 10 (très drôle). Kohli *et al*. (2013) ont choisi de fixer le seuil de pertinence à 7 : toutes les blagues qui ont une note strictement supérieure à 7 sont considérées comme cliquées par l'utilisateur.

Pour simuler l'aspect temps réel du SR, à chaque instant t, un utilisateur est choisi aléatoirement et m=5 objets lui sont recommandés. L'utilisateur choisi est toujours considéré comme un nouvel utilisateur. Ce dernier clique sur un objet seulement si la note associée est strictement supérieure au seuil choisi. Si l'utilisateur clique au moins sur un objet, la récompense obtenue est de $Z_t = 1$, sinon $Z_t = 0$. Notre objectif est de maximiser la somme des Z_t, autrement dit de minimiser l'abandon (voir section 2).

Chaque expérimentation est réalisée sur un intervalle de temps de longueur T = 100000. Les figures 1, 2 et 3 représentent le taux de clics sur les 1000 derniers instants, moyenné sur 200 expérimentations de Monte Carlo.

Afin d'avoir une comparaison efficace entre notre approche et les approches existantes (*RBA* et *IBA*), nous avons choisi de les implémenter tout en utilisant les algorithmes de bandits ϵ-*greedy* et *Upper Confidence Bound 1 (UCB1)*. Ce sont les algorithmes utilisés dans l'article de Kohli *et al*. (2013) qui nous sert de référence pour le cadre expérimental.

Les résultats expérimentaux sont représentés dans la figure 1, présentant l'expérimentation sur la collection Movielens avec comme seuil de pertinence 2, la figure 2, présentant l'expérimentation sur la collection Movielens avec comme seuil de pertinence 4, et la figure 3 présentant l'expérimentation sur la collection Jester avec comme seuil de pertinence 7. Dans ces figures, la combinaison optimale et la combinaison indépendante sont représentées par les courbes horizontales. Dans les trois expérimentations mises en place, la combinaison diversifiée est la même que la combinaison optimale. Pour rappel la combinaison indépendante est constituée des m objets les plus cliqués, et la combinaison diversifiée est constituée de l'objet le plus populaire en première position et des objets les plus populaires lorsque les objets aux positions précédentes ne sont pas cliqués. L'approche *IBA* vise la combinaison indépendante tandis que l'approche *RBA* et l'approche *MPB* visent la combinaison diversifiée. La différence entre ces deux combinaisons est approximativement de 1% pour l'expérimentation sur la collection Movielens avec le seuil de pertinence 4 (voir figure 2) et l'expérimentation sur la collection Jester (voir figure 3). Néanmoins, elle est plus importante dans le cadre de l'expérimentation sur la collection MovieLens avec le seuil de pertinence 2, avec un écart de 4% environ (voir figure 1).

5.2. Commentaires sur les résultats

UCB1 Nous avons souhaité observer le comportement des trois approches étudiées lorsque celles-ci sont implémentées avec l'algorithme de bandits *UCB1*. L'approche *MPB* obtient des proportions d'abandon équivalentes ou meilleures (c'est-à-dire un proportion d'abandon plus faible) que les approches *RBA* et *IBA* au terme des 3 expérimentations. En outre, pour l'approche *MPB* le temps d'apprentissage est bien plus court. Pour vérifier la pertinence statistique de l'amélioration du temps d'apprentissage, un test unilatéral de Wilcoxon a été mis en place. Ce test permet de donner un niveau de confiance sur l'hypothèse que la vitesse d'apprentissage de l'approche *MPB* est moins bonne ou identique à l'approche comparée, l'hypothèse alternative étant que l'approche *MPB* apprend plus vite. Très clairement, l'approche *MPB* obtient de meilleurs résultats, avec des p-values toutes inférieures à 2.2×10^{-16}.

Pour quantifier l'amélioration de la vitesse d'apprentissage, nous comparons le nombre d'étapes de temps nécessaire aux différentes approches pour atteindre 95% et 98% de la combinaison indépendante en moyenne. Les méthodes de l'état de l'art atteignent ces valeurs uniquement sur la collection MovieLens avec le seuil de perti-

Tableau 1. Résultats du Test de Wilcoxon unilatéral pour comparer le nombre d'étapes nécessaire pour atteindre 95% et 98% de la combinaison indépendante en utilisant les méthodes de l'état de l'art avec les algorithmes de bandits ϵ-greedy et UCB1 et cette même quantité en utilisant l'approche MPB. Hypothèse alternative : l'approche MPB apprend plus vite. L'hypothèse alternative est retenue pour l'ensemble des comparaisons, hormis la comparaison avec l'approche IBA utilisant l'algorithme ϵ-greedy

% (1 - abandon) de la combinaison indépendante		**95 %**	**98 %**
collection	approche	p-value	p-value
MovieLens seuil = 2	**IBA-Egreedy**	0,0010	0.9385
	RBA-Egreedy	1,0e-11	0,0067
	RBA-UCB1	<2.2e-16	<2.2e-16
	IBA-UCB1	<2.2e-16	<2.2e-16
MovieLens seuil = 4	**IBA-Egreedy**	0,82	0,84
	RBA-Egreedy	8,0e-6	0,00032
	RBA-UCB1	<2.2e-16	<2.2e-16
	IBA-UCB1	<2.2e-16	<2.2e-16
Jester seuil = 7	**IBA-Egreedy**	9,3e-11	0,0012
	RBA-Egreedy	<2.2e-16	1,3e-15
	RBA-UCB1	<2.2e-16	<2.2e-16
	IBA-UCB1	<2.2e-16	<2.2e-16

Tableau 2. Ratios approximatifs entre le nombre d'étapes nécessaire pour atteindre 95% et 98% de la combinaison indépendante en utilisant les méthodes de l'état de l'art avec l'algorithme de bandits ϵ-greedy et cette même quantité en utilisant l'approche MPB. ((+) : l'approche MPB atteint le seuil, mais pas l'approche comparée, (-) : aucune des deux approches n'atteint le seuil). L'approche MPB atteint le seuil de 95% au moins aussi rapidement que les approches comparées. Cependant, l'approche MPB met plus de temps à atteindre le seuil 98% lorsque l'algorithme ϵ-greedy est utilisé que les approches RBA et IBA

collection	approche	ratio (95%)	ratio (98%)
MovieLens seuil = 2	**IBA-Egreedy**	$\approx$ 1,0	$\approx$ 0,5
	RBA-Egreedy	$\approx$ 2,0	$\approx$ 0,3
	IBA-UCB1	(+)	(+)
	RBA-UCB1	(+)	(+)
MovieLens seuil = 4	**IBA-Egreedy**	$\approx$ 1,0	(-)
	RBA-Egreedy	(+)	(-)
	IBA-UCB1	(+)	(+)
	RBA-UCB1	(+)	(+)
Jester seuil = 7	**IBA-Egreedy**	$\approx$ 1,5	$\approx$ 0,3
	RBA-Egreedy	$\approx$ 2,5	$\approx$ 0,6
	IBA-UCB1	(+)	(+)
	RBA-UCB1	(+)	(+)

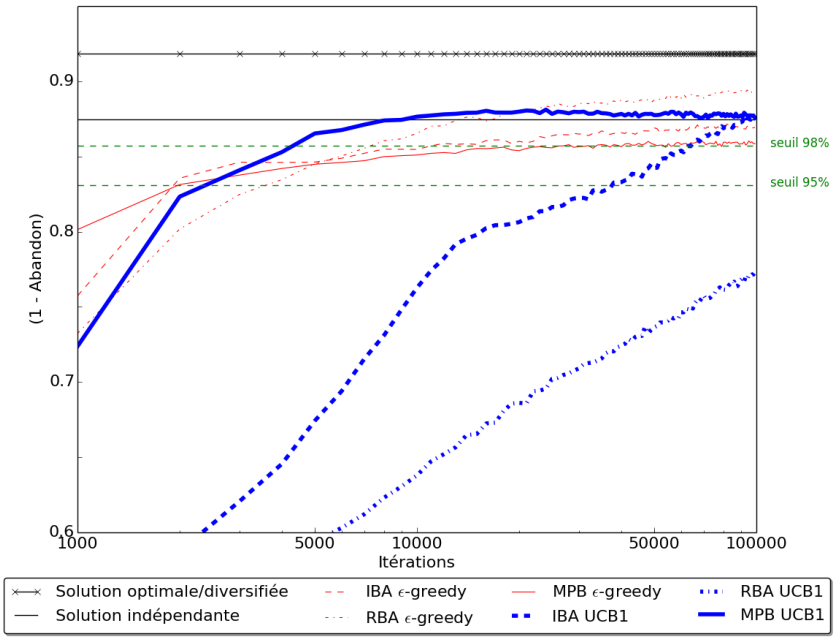

Figure 1. Comparaison des différentes approches avec le jeu de données MovieLens avec un seuil de pertinence = 2

nence 2 (figure 1). Pour cette expérimentation, l'approche *MPB* atteint treize fois plus vite les deux seuils. Concernant les expérimentations sur la collection MovieLens avec le seuil de pertinence 4 et sur la collection Jester, les seuils 95% et 98% ne sont pas atteints en moyenne par les approches (*RBA* et *IBA*). La forme des courbes laisse supposer que ces deux approches n'ont pas eu suffisamment de 100000 étapes de temps pour atteindre les deux valeurs seuils. Cependant dans la figure 1, l'approche *MPB* atteint la valeur seuil de 95% en 3000 itérations et 98% en 5000 itérations en moyenne sur la collection MovieLens avec le seuil de pertinence de 2. Dans la figure 3, pour la collection Jester, 9000 itérations sont nécessaires pour atteindre 95% de la combinaison indépendante, et 18000 itérations pour la valeur seuil de 98%. Dans la figure 2, Avec le seuil de pertinence 4 sur la collection MovieLens, les valeurs seuils sont atteintes moins rapidement, mais toujours avant les 100000 itérations (95% en 46000 itérations, 98% en 96000 itérations en moyenne.

ϵ-greedy Regardons maintenant comment se comporte l'approche *MPB* lorsque l'algorithme de bandits *ϵ-greedy* est utilisé. Pour les 3 expérimentations mises en oeuvre, globalement l'approche *MPB* obtient des proportions d'abandon équivalentes à celles obtenues par les méthodes de l'état de l'art. D'après les Tableaux 1 et 2, une

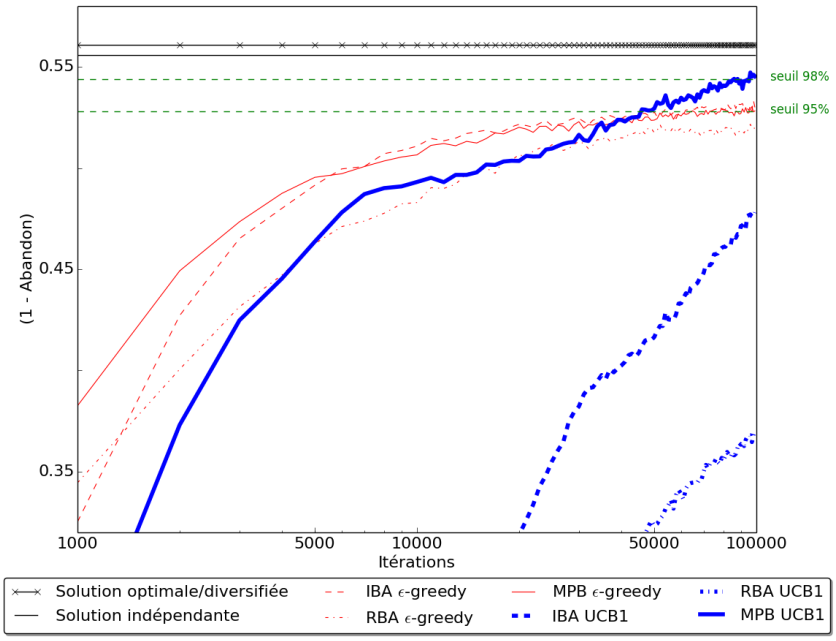

Figure 2. Comparaison des différentes approches avec le jeu de données MovieLens avec un seuil de pertinence = 4

amélioration significative de la vitesse d'apprentissage peut être constatée en comparant l'approche *MPB* avec l'approche *RBA*. Par contre, cette implémentation avec ϵ-*greedy* ne permet pas d'observer une nette amélioration de la vitesse d'apprentissage en comparant l'approche *MPB* avec l'approche *IBA*. Il apparait tout de même que pour le seuil de 95%, notre approche est au moins aussi rapide que les approches existantes. Ce n'est pas le cas pour le seuil 98% dans la figure 2, où sur la collection MovieLens avec le seuil de pertinence 4, *MPB* n'atteint pas ce seuil en moyenne. Pour la collection MovieLens avec le seuil de pertinence 2 de la figure 1, le seuil est atteint mais deux fois moins rapidement qu'avec l'approche *IBA*. Enfin dans la figure 3 avec la collection Jester, trois fois plus d'itérations sont nécessaires pour atteindre le seuil.

L'algorithme de bandits ϵ-*greedy* réalise son exploration en choisissant quelques objets au hasard avec une probabilité de ϵ. En utilisant un seul algorithme de bandits comme dans l'approche *MPB*, le risque est de recommander les objets avec les meilleures proportions de clics très rarement, au profit d'autres. Ceci est particulièrement vrai si ces objets ne sont pas cliqués les premières fois qu'ils sont recommandés. Expérimentalement nous avons pu observer que l'utilisation d'autant d'instances d'un algorithme de bandits que de objets à recommander, comme dans les approches *RBA*

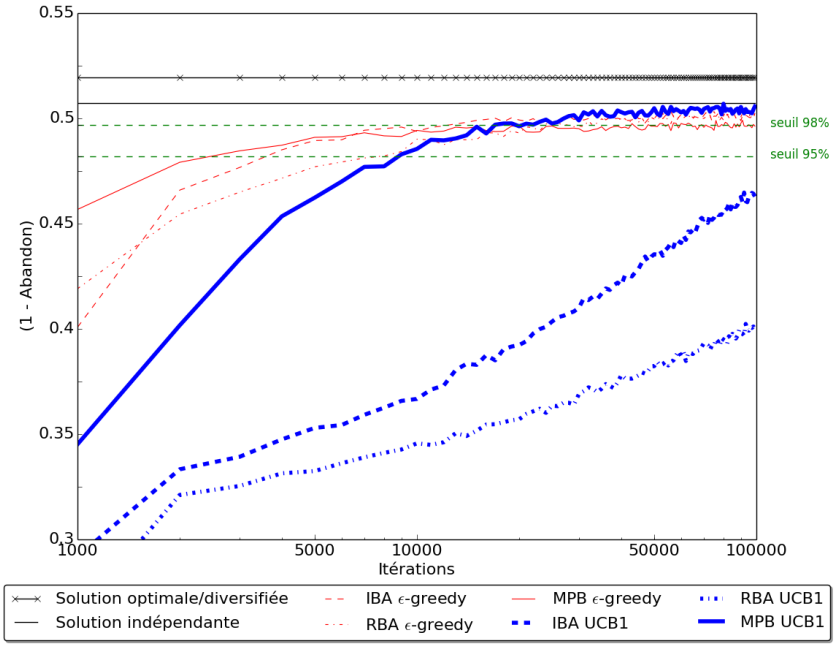

Figure 3. Comparaison des différentes approches avec le jeu de données Jester avec un seuil de pertinence = 7

et *IBA*, permet d'estomper ce problème. En effet si un objet n'est pas cliqué à un instant donné, une seule instance le prendra en compte, laissant ainsi toutes ces chances d'être recommandé par l'une des autres instances.

L'amélioration significative des vitesses d'apprentissage apportée par l'approche *MPB* lorsque l'algorithme de bandits *UCB1* est utilisé est encourageante. Un ajustement de la manière dont est calculée la borne de confiance supérieure pourrait améliorer le gain. En visant la combinaison diversifiée, cette approche atteint des proportions d'abandon inférieures aux autres approches sur le long terme ($t > 50000$). Cependant, lorsque l'algorithme de bandits ϵ-*greedy* est utilisé, l'approche *MPB* n'améliore pas significativement la vitesse d'apprentissage, et l'approche *IBA* reste la plus rapide.

6. Conclusion

Dans cet article, nous nous sommes intéressés au problème de la recommandation à tirages multiples. Habituellement, les approches de l'état de l'art utilisent autant d'instances d'un algorithme de bandits à tirages simples qu'il y a d'objets à recommander. Pour améliorer la vitesse d'apprentissage, nous avons proposé de gérer

l'ensemble des recommandations simultanément, en utilisant une seule instance d'un algorithme de bandits à tirages simples. Dans le cas de l'utilisation de l'algorithme de bandits *UCB1*, l'approche proposée *MPB* apprend beaucoup plus rapidement (jusqu'à treize fois plus rapidement). De plus elle obtient des proportions d'abandon équivalentes à celles de l'approche la plus rapide de l'état de l'art, *IBA*. Cependant en utilisant l'algorithme de bandits ϵ-*greedy*, l'approche *MPB* n'atteint pas toujours des proportions d'abandon aussi faibles que *IBA*, pour une vitesse d'apprentissage équivalente. Nous avons pu observer expérimentalement que la nature aléatoire de l'algorithme ϵ-*greedy* peut expliquer qu'aucune amélioration n'est observable (voir fin section 5). L'évaluation a été réalisée sur deux jeux de données de référence, MovieLens et Jester.

Les résultats présentés dans la section 5 du présent article montrent que le temps d'apprentissage est nettement amélioré lorsque l'approche *MPB* est utilisée avec l'algorithme de bandits *UCB1* par rapport aux approches *RBA* et *IBA*. Cela indique que l'algorithme *UCB1* apprend plus rapidement si, à chaque instant, plusieurs retours utilisateurs sont observés. Nous envisageons dans nos travaux futurs de réfléchir à une version dérivée de l'algorithme de bandits à tirages simples *UCB1*, avec de fortes garanties théoriques, capable de gérer le tirage de plusieurs objets simultanément.

Remerciements Les travaux de Jonathan Louëdec sont financés par le LabEx CIMI. Les auteurs remercient le soutien apporté par l'Agence Nationale de la Recherche (ANR-13-CORD-0020, project ALICIA).

Bibliographie

Agrawal S., Goyal N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In *Proceedings of the 25th Conference on Learning Theory (COLT)*.

Ailon N., Hatano K., Takimoto E. (2014). Bandit online optimization over the permutahedron. In *Proceedings of the 25th International Conference in Algorithmic Learning Theory (ALT)*.

Audibert J.-Y., Munos R., Szepesvári C. (2009). Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, vol. 410, n° 19, p. 1876–1902.

Auer P., Cesa-Bianchi N., Fischer P. (2002). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, vol. 47, n° 2-3, p. 235–256.

Auer P., Cesa-Bianchi N., Freund Y., Schapire R. E. (2002). The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, vol. 32, n° 1, p. 48–77.

Bubeck S., Cesa-Bianchi N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, vol. 5, n° 1, p. 1-122.

Cesa-Bianchi N., Fischer P. (1998). Finite-time regret bounds for the multiarmed bandit problem. In *Proceedings of the 15th International Conference on Machine Learning (ICML)*, p. 100–108.

Chen H., Karger D. R. (2006). Less is more: probabilistic models for retrieving fewer relevant documents. In *Proceedings of the 29th international ACM Conference on Research and Development in Information Retrieval (SIGIR)*, p. 429–436.

Chen W., Wang Y., Yuan Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the 30th international conference on machine learning (icml)*, p. 151–159.

Chu W., Li L., Reyzin L., Schapire R. E. (2011). Contextual bandits with linear payoff functions. In *Proceedings of the 14th international conference on artificial intelligence and statistics*, p. 208–214.

Garivier A., Cappé O. (2011). The kl-ucb algorithm for bounded stochastic bandits and beyond. In *Proceedings of the 24th Conference on Learning Theory (COLT).*

Hastie T., Tibshirani R., Friedman J. (2009). *The elements of statistical learning* (2nd éd.). Springer.

Kaufmann E., Korda N., Munos R. (2013). Thompson sampling: An asymptotically optimal finite-time analysis. In *Proceedings of the 23th International Conference in Algorithmic Learning Theory (ALT).*

Kohli P., Salek M., Stoddard G. (2013). A fast bandit algorithm for recommendation to users with heterogenous tastes. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence*, p. 1135–1141.

Komiyama J., Honda J., Nakagawa H. (2015). Optimal regret analysis of thompson sampling in stochastic multi-armed bandit problem with multiple plays. In *Proceedings of the 32th International Conference on Machine Learning (ICML).*

Li L., Chu W., Langford J., E.Schapire R. (2010). A contextual-bandit approach to personalized news article recommandation. In *Proceedings of 19th International World Wide Web Conference*, p. 661–670.

Louedec J., Chevalier M., Garivier A., Gerchinovitz S., Mothe J. (2015). A multiple-play bandit algorithm applied to recommender systems. In *Proceedings of the 28th International Florida Artificial Intelligence Research Society Conference (FLAIRS).*

Oza N. C. (2005). Online bagging and boosting. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, vol. 3, p. 2340–2345.

Radlinski F., Kleinberg R., Joachims T. (2008). Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, p. 784–791.

Ricci F., Rokach L., Shapira B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook*, p. 1-35. Springer.

Robertson S. E. (1977). The probability ranking principle in ir. *Journal of documentation*, vol. 33, n° 4, p. 294–304.

Sutton R. S., Barto A. G. (1999). Reinforcement learning. *Journal of Cognitive Neuroscience*, vol. 11, n° 1, p. 126–134.

Tang L., Jiang Y., Zeng L. L. C., Li T. (2015). Personalized recommendation via parameter-free contextual bandits. In *Proceedings of the 38th international ACM Conference on Research and Development in Information Retrieval (SIGIR).*

Thompson W. R. (1933). On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, p. 285–294.

Uchiya T., Nakamura A., Kudo M. (2010). Algorithms for adversarial bandit problems with multiple plays. In *Algorithmic learning theory*, p. 375-389.

Watkins C. J. C. H. (1989). Learning from delayed rewards.

Yue Y., Guestrin C. (2011). Linear submodular bandits and their application to diversified retrieval. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems (NIPS)*, p. 2483–2491.