Algorithmes de bandit pour les systèmes de recommandation : le cas de multiples recommandations simultanées

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

[1] *Institut de Recherche en Informatique de Toulouse, UMR 5505, CNRS, Université de Toulouse, France*
[2] *Institut de Mathématiques de Toulouse, UMR 5219, Université de Toulouse, France*

RÉSUMÉ. *Les systèmes de recommandation (SR) à tirages multiples font référence aux SR qui recommandent plusieurs objets aux utilisateurs. La plupart des SR s'appuient sur des modèles d'apprentissage afin de décider les objets à recommander. Parmi les modèles d'apprentissage, les algorithmes de bandit offrent l'avantage de permettre d'apprendre tout en exploitant les éléments déjà appris. Les approches actuelles utilisent autant d'instances d'un algorithme de bandit que le SR doit recommander d'objets. Nous proposons au contraire de gérer l'ensemble des recommandations par une seule instance d'un algorithme de bandit 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), permet d'obtenir un temps d'apprentissage jusqu'à treize fois plus rapide tout en obtenant des taux de clics équivalents. Nous montrons également que le choix de l'algorithme de bandit utilisé influence l'amélioration obtenue.*

ABSTRACT. *The multiple-plays recommender systems (RS) make a refer to RS which recommend several items to the users. Most of RS are based on learning models in order to choose items to be recommended. Among learning models, the bandit algorithms offer the advantage to learn and exploite the already learnt elements at the same time. The current approaches require running as many instance of single-play bandit algorithms as there are items to recommend. We propose to manage all recommendations simultaneously, by a single instance of one single-play bandit algorithm. We show on two benchmark datasets (Movielens and Jester) that our method, MPB (Multiple Plays Bandit), allows to obtain a learning rate until thirteen times faster while obtaining equivalent click-through rates. We also show that the choice of the bandit algorithm used impacts this improvement.*

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

KEYWORDS: *Information Retrieval, Recommender Systems, Bandits Problem.*

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, particulièrement adapté à la recommandation à tirages multiples. Ce problème peut également être vu comme la maximisation du nombre de fois où les utilisateurs sélectionnent au moins un objet parmi ceux recommandés simultanément.

Afin d'améliorer la pertinence des recommandations pour l'utilisateur courant, un SR peut considérer l'historique des interactions passées avec les utilisateurs. Pour cela, il convient de mettre en place une stratégie permettant d'apprendre sur la pertinence des objets tout en continuant de recommander des objets pertinents. Lorsque toutes les données sont connues, il est possible d'estimer la pertinence des objets, c'est le cadre d'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. De plus, le choix des objets à recommander à chaque interaction est réalisé en fonction des interactions passées. Un tel environnement s'inscrit dans le cadre de ce que l'on appelle l'*apprentissage par renforcement* (Sutton et Barto, 1999). Il s'agit d'implémenter une stratégie pour obtenir de nouvelles informations (exploration), tout en assurant que le SR recommande des objets pertinents (exploitation). Ce problème est connu sous le nom de "dilemme exploration/exploitation". Les algorithmes de bandit sont connus pour offrir des solutions à ce dilemme (Bubeck et Cesa-Bianchi, 2012).

Plusieurs travaux proposent déjà des approches pour utiliser les algorithmes de bandit 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 bandit 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 bandit. Selon notre approche, l'instance de bandit est utilisée plusieurs fois à chaque itération.

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 à chaque instant. 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 bandit 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 des instances de l'algorithme de bandit *UCB1* (jusqu'à treize fois plus rapide). Nous voyons aussi que notre approche ne permet pas d'améliorer les performances lorsque l'algorithme de bandit utilisé est 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 d'expliquer en partie ce phénomène. 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 bandit 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

Plusieurs approches utilisant des algorithmes de bandit peuvent adresser certaines problématiques de recommandations.

Considérons tout d'abord les problématiques contextuelles. Ils existent des approches capables d'induire de nouvelles informations sur des objets non recommandés et des utilisateurs non connectés. Pour un objet non recommandé, il est possible d'induire de l'information à partir du retour obtenu sur un objet similaire. Pour un utilisateur non connecté, il est possible d'induire de l'information à partir de l'observation faite sur un utilisateur similaire. Dans ce cas des recommandations personnalisées peuvent être effectuées. La plupart de ces approches sont conçues pour recommander un seul objet à chaque instant, comme par exemple les approches de bandit stochastique linéaire (Li *et al.*, 2010) et (Chu *et al.*, 2011). Toutefois il existe certaines approches capables de recommander plusieurs objets à chaque instant : Yue et Guestrin (2011) proposent l'algorithme *LSBGreedy*.

Les problématiques contextuelles 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 à chaque instant (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. Dans ce cadre, Radlinski *et al.* (2008) ont développé l'approche *Ranked Bandits Algorithm* (*RBA*) qui utilisent autant d'instances d'un algorithme de bandit à tirages uniques que de recommandations à soumettre. Ces instances peuvent être des algorithmes tels que *UCB1* (Auer *et al.*, 2002a), ϵ-*greedy* (Sutton et Barto, 1999) ou encore *Exp3* (Auer *et al.*, 2002b). 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).

Certains systèmes de recommandation proposent des recommandations ordonnées selon la pertinence des objets par rapport à un utilisateur. En recherche d'information c'est le cas également de nombreux moteurs de recherche. Ailon *et al.* (2013) propose l'algorithme *BanditRank* pour aborder la problématique d'ordonnancement. Pour cela l'algorithme soumet à chaque instant une permutation de l'ensemble des objets disponibles, dont l'objectif principal est de trier les objets par ordre de pertinence en fonction du retour utilisateur.

L'approche *Combinatorial Multi-Armed Bandit (CMAB)* proposée par Chen *et al.* (2013) utilise un seul algorithme de bandit 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 (voir fin section 3).

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.

3. Modèles de bandit pour la recommandation

Dans cet article une recommandation sélectionnée par l'utilisateur est considérée comme pertinente : 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 :

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

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. À partir de là, 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 :

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

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 :

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

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 basée sur le principe de classement par probabilité (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 (Chen et 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, la combinaison diversifiée 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 de la proportion d'abandon. C'est généralement le cas, cependant il est possible de construire des cas où la combinaison indépendante est meilleure.

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

Dans cette section, nous commençons par présenter les deux approches que nous utiliserons comme approches de références : *RBA* et *IBA*. 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 bandit à K bras

Algorithme 1 : Ranked Bandits Algorithm

BTS_i : instance d'un algorithme de bandit 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

à tirages simples. À chaque instant t, les m objets choisis par les m instances d'algorithme de bandit 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 avec le taux de clics le plus haut quand l'objet avec le taux de clics le plus haut n'est pas 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 bandit à K bras 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.

Algorithme 2 : Independent Bandits Algorithm

BTS_i : instance d'un algorithme de bandit 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

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 terme (lorsque t est grand). Cependant, Kohli *et al.* (2013) ont montré dans leurs simulations 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, l'ensemble des m instances d'un algorithme de bandit à K bras à tirages simples fonctionnent de façon (quasiment) indépendants les uns 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 bandit 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 deux approches précédentes est que nous utilisons une seule instance d'un algorithme de bandit. Pour recommander simultanément une liste de m objets, le premier objet est sélectionné selon la règle utilisée par le bandit. Le

Algorithme 3 : Multiple Plays Bandit

BTS : instance d'un algorithme de bandit 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

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 bandit. 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 très 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 très vite 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 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 valeursont été utilisées : 2 et 4. Lorsque le seuil est 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 apprécié. La même logique est suivie pour le seuil de valeur 2.

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 comme seuil à partir duquel les blagues recommandées sont considérées comme pertinentes.

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 clic 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 tous en utilisant les algorithmes de bandit 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.

Le premier, l'algorithme ϵ-*greedy* (Auer *et al.*, 2002b), ajoute une dimension aléatoire dans le choix du objet à recommander : à chaque instant t, un objet choisi au hasard (uniformément) est recommandé avec une probabilité de ϵ, sinon, avec une probabilité de $(1 - \epsilon)$, l'objet avec le meilleur taux de clics jusque là est recommandé. Cet algorithme est celui qui donne les meilleurs résultats dans le papier de Kohli *et al.* (2013). Dans cet article, les auteurs précisent que $\epsilon = 0.05$ donnent les meilleurs résultats dans leurs tests initiaux, nous prendrons donc également cette valeur.

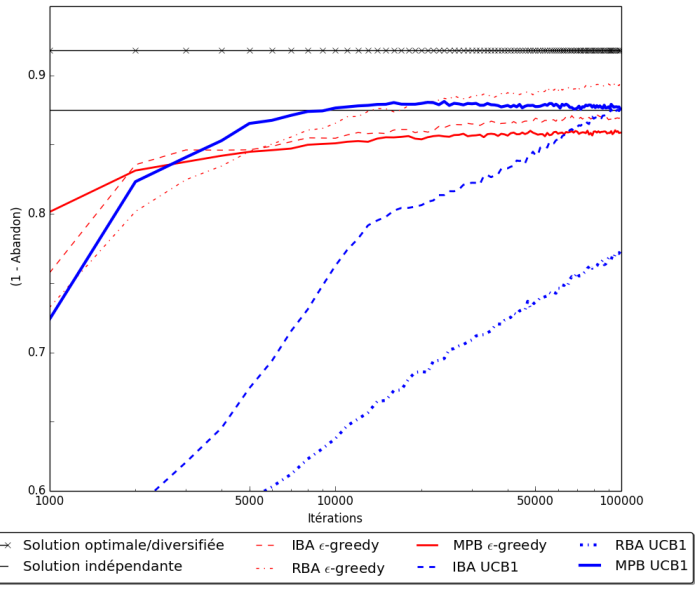

Figure 1. *Jeu de données MovieLens avec un seuil de pertinence = 2*

L'algorithme *UCB1* (Auer *et al.*, 2002b) est un algorithme optimiste utilisant les bornes supérieures de confiance. À chaque instant, *UCB1* recommande l'objet qui maximise la borne de confiance $\bar{x}_i + \sqrt{\frac{2\log(t)}{t_i}}$ où $\bar{x}_i$ est le taux de clics observé à l'instant t pour l'objet i et t_i est le nombre de fois où l'objet i a été recommandé. Plus un objet a déjà été proposé, moins le second terme de l'équation sera important. Il s'agit donc de privilégier les objets qui n'ont pas été suffisamment testés.

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 représentés par les courbes horizontales. Dans les trois expérimentations mises en place, la combinaison diversifiée est la même que la combinaison optimale. L'approche *IBA* vise la combinaison indépendante tandis que l'approche *RBA* et notre approche visent la combinaison diversifiée. La différence entre ces deux combinaisons est approximativement de 1% pour l'expérimentation sur la collection Movielens et 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 et le seuil de pertinence 2, avec un écart de 4% environ (voir figure 1).

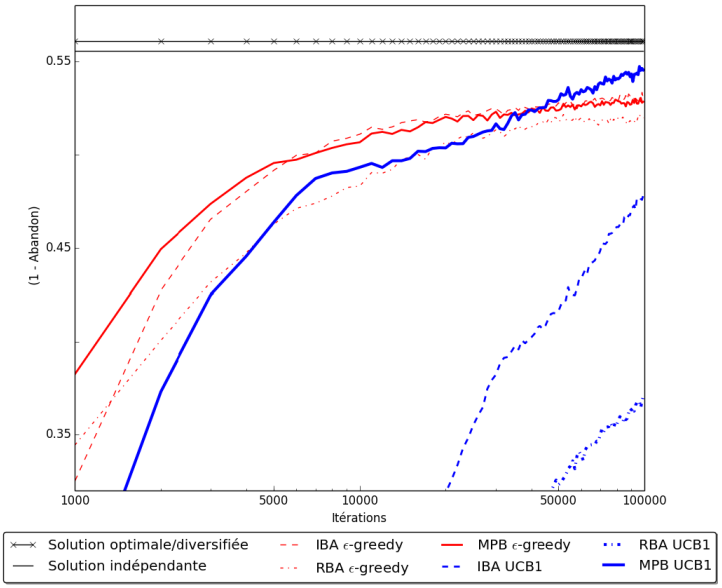

Figure 2. *Jeu de données MovieLens avec un seuil de pertinence = 4*

% (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
MovieLens seuil = 4	**IBA-Egreedy**	0,82	0,84
	RBA-Egreedy	8,0e-6	0,00032
Jester seuil = 7	**IBA-Egreedy**	9,3e-11	0,0012
	RBA-Egreedy	<2.2e-16	1,3e-15

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 l'algorithme de bandit ϵ-greedy et cette même quantité en utilisant l'approche MPB. Hypothèse alternative : l'approche MPB apprend plus vite.*

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 bandit *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

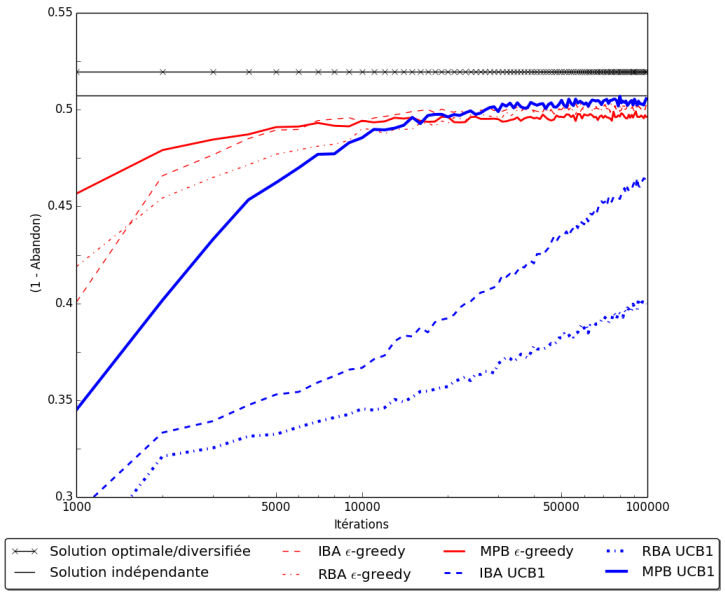

Figure 3. *Jeu de données Jester avec un seuil de pertinence = 7*

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

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 bandit UCB1 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)*

3 expérimentations (voir courbes bleues sur les figures 1, 2 et 3). En outre, pour l'approche *MPB* le temps d'apprentissage est bien plus court. Pour vérifier la pertinence statistique de cette amélioration, 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 pertinence 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 deux seuils ne sont pas atteint 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 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 (voir figure 1). Pour la collection Jester, 9000 itérations sont nécessaires pour atteindre 95% de la combinaison indépendante, et 18000 itérations pour la valeur seuil de 98% (voir figure 3). 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) (voir figure 2).

ϵ-greedy Regardons maintenant comment se comporte l'approche *MPB* lorsque l'algorithme de bandit *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 (voir courbes rouges sur les figures 1, 2 et 3). D'après les Tableaux 1 et 2, une 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%, où sur la collection MovieLens avec le seuil de pertinence 4, *MPB* n'atteint pas ce seuil en moyenne (voir figure 2). Pour la collection MovieLens avec le seuil de pertinence 2, le seuil est atteint mais deux fois moins rapidement qu'avec l'approche *IBA* (voir figure 1). Enfin sur la collection Jester, trois fois plus d'itérations sont nécessaires pour atteindre le seuil (voir figure 3).

L'algorithme de bandit *ϵ-greedy* réalise son exploration en choisissant quelques objets au hasard avec une probabilité de ϵ. En utilisant un seul algorithme de bandit 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 bandit que de objets à recommander, comme dans les approches *RBA* 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 bandit *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 bandit 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 bandit 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 bandit à tirages simples. Dans le cas de l'utilisation de l'algorithme de bandit *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 celle de l'approche la plus rapide de l'état de l'art, *IBA*. Cependant en utilisant l'algorithme de bandit 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.

Perspectives De récents travaux en apprentissage par renforcement proposent un nouvel algorithme de bandit capable de directement gérer le tirage de plusieurs objets à chaque instant : *Exp3_M* (Uchiya *et al.*, 2010) (Bubeck et Cesa-Bianchi, 2012). Cet algorithme est dérivé de l'algorithme de bandit à tirages uniques : *Exp3*. Dans l'article (Louedec *et al.*, 2015), nous montrons que l'utilisation d'*Exp3_M* directement permet d'obtenir de meilleurs résultats que l'utilisation de *Exp3* via les approches *RBA* et *IBA*. 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 bandit *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 bandit à tirages uniques *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).

7. Bibliographie

Ailon N., Hatano K., Takimoto E., « Bandit Online Optimization Over the Permutahedron », *CoRR*, 2013.

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

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

Bubeck S., Cesa-Bianchi N., « Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems », *Foundations and Trends in Machine Learning*, vol. 5, n° 1, p. 1-122, 2012.

Chen H., Karger D. R., « Less is more : probabilistic models for retrieving fewer relevant documents », *29th international ACM SIGIR conference on Research and development in information retrieval*, p. 429-436, 2006.

Chen W., Wang Y., Yuan Y., « Combinatorial multi-armed bandit : General framework and applications », *Proceedings of the 30th International Conference on Machine Learning*, p. 151-159, 2013.

Chu W., Li L., Reyzin L., Schapire R. E., « Contextual bandits with linear payoff functions », *International Conference on Artificial Intelligence and Statistics*, p. 208-214, 2011.

Hastie T., Tibshirani R., Friedman J., *The Elements of Statistical Learning*, 2nd edn, Springer, 2009.

Kohli P., Salek M., Stoddard G., « A Fast Bandit Algorithm for Recommendation to Users With Heterogenous Tastes », *27th AAAI Conference on Artificial Intelligence*, p. 1135-1141, 2013.

Li L., Chu W., Langford J., E.Schapire R., « A Contextual-Bandit Approach to Personalized News Article Recommandation », *Proc. of 19th International World Wide Web Conference*, p. 661-670, 2010.

Louedec J., Chevalier M., Garivier A., Gerchinovitz S., Mothe J., « A Multiple-play Bandit Algorithm Applies to Recommender Systems », *Proceedings of the 28th International Florida Artificial Intelligence Research Society Conference*, 2015.

Radlinski F., Kleinberg R., Joachims T., « Learning Diverse Rankings with Multi-Armed Bandits », *25th International Conference on Machine Learning*, p. 784-791, 2008.

Ricci F., Rokach L., Shapira B., « Introduction to Recommender Systems Handbook », *Recommender Systems Handbook*, Springer, p. 1-35, 2011.

Robertson S. E., « The probability ranking principle in IR », *Journal of documentation*, vol. 33, n° 4, p. 294-304, 1977.

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

Uchiya T., Nakamura A., Kudo M., « Algorithms for Adversarial Bandit Problems with Multiple Plays », *Algorithmic Learning Theory*, LNCS Springer, p. 375-389, 2010.

Yue Y., Guestrin C., « Linear submodular bandits and their application to diversified retrieval », *Advances in Neural Information Processing Systems*, p. 2483-2491, 2011.