Intégration des séries temporelles dans les A/B-Tests

Emmanuelle Claeys*, Pierre Gançarski *, Myriam Maumy-Bertrand**

*ICube **IRMA
Université de Strasbourg – France
{claeys, gancarski}@unistra.fr, mmaumy@math.unistra.fr

Résumé. Récemment en e-commerce, de nouvelles méthodes prometteuses optimisent les A/B-Tests en utilisant une allocation dynamique des items aux variations permettant ainsi de déterminer plus rapidement la meilleure variation et donc de réduire les coûts du test. Cependant, ces méthodes qui peuvent s'apparenter à un apprentissage par renforcement, restent limitées à des données statiques et ne peuvent prendre en compte des données temporelles évolutives. Nous présentons ici deux nouvelles méthodes basées sur une approche commune, qui permettent d'intégrer des séries temporelles dans le profil des visiteurs. Nous montrons dans cet article qu'elles améliorent l'allocation dynamique des A/B-Tests en présentant des résultats obtenus sur des données issues de tests réels.

1 Introduction

Dans de nombreux domaines économiques, industriels voire sociaux, il peut être intéressant d'évaluer la pertinence d'une modification sur une entité (médicament, page web,...) par rapport à un ou plusieurs objectifs (par exemple augmenter le nombre de survies, le nombre de clics, la valeur d'achats,...) en comparant directement et concrètement *in vivo* les différentes variations. Il s'agit alors d'établir un processus permettant d'évaluer différentes alternatives d'une entité et ainsi de choisir l'optimale, c'est-à-dire la meilleure dans le contexte donné et pour un objectif défini. Ainsi, un A/B-Test consistera à évaluer concrètement ces différentes alternatives par rapport à un objectif défini a priori. Les A/B-Tests ont connu récemment un regain d'intérêt notamment de par leur utilisation dans le e-marketing, pour par exemple, tenter d'améliorer une page web en production. Une méthode classique consiste à diviser le trafic du site à parts égales sur les différentes variations testées. Chaque décision d'*affectation* d'un *visiteur* à une variation, c'est-à-dire, le choix de la page web à afficher au visiteur, est irrévocable : un visiteur verra toujours la même page web à chacune de ses visites. De cette contrainte forte découle le fait qu'il est impossible de savoir quel aurait été le gain obtenu en cas d'affectation du visiteur à une autre variation. De plus, les visiteurs soumis au test n'ont généralement pas connaissance d'être testés et ignorent qu'il existe des variations différentes. À la fin du test, les résultats de chacune des variations sont comparés en fonction des gains réellement obtenus et la variation optimale est déterminée. Pour mettre en place un tel test, l'utilisateur, en amont du test, fixe une période

de test et observe les résultats à l'issue de celle-ci. Cependant, en pratique, il lui est très difficile de définir cette durée a priori. L'expérience montre qu'il a tendance du coup à la surestimer. Or, si une des variations testées dégrade fortement les résultats, une période de test trop grande aura pour conséquence d'affecter trop longtemps des visiteurs à une variation sous-optimale produisant un gain inférieur à celui qu'aurait produit la variation optimale qui est bien sûr inconnue a priori ; l'objectif de test étant par définition, de la déterminer. Ces affectations sous-optimales induisent un coût de test, souvent appelé *regret*, pouvant être d'autant plus important que la dégradation est forte.

Pour répondre à ce problème, de nouvelles techniques d'A/B-Tests réalisent une allocation dynamique du trafic via des algorithmes de bandits (Lattimore et Szepesvári, 2019). L'*allocation dynamique* consiste à adapter l'allocation des visiteurs en fonction des gains obtenus et ainsi progressivement basculer les visiteurs vers la variation optimale. Ayant montré leur intérêt dans de nombreuses situations mais aussi leurs limites, ces algorithmes de bandits ont été étendus pour prendre en compte le *contexte* des visiteurs, c'est-à-dire leurs caractéristiques (âge, provenance, sexe,...) lors de l'affectation. Malheureusement, ces *bandits contextuels* (Zhou, 2015) limitent les contextes à des vecteurs à valeurs numériques ou symboliques statiques. Pourtant, les recherches en marketing ont montré qu'un affichage de pages personnalisé en fonction du comportement évolutif du visiteur permettait d'augmenter le gain espéré d'une page web. Par analogie, il pourrait être intéressant d'intégrer des données temporelles dans le contexte descriptif d'un visiteur. Ainsi l'utilisation d'informations temporelles (liste des sites visités par un nouveau visiteur avant d'arriver sur le site hébergeant le test, pages parcourues et options cliquées par celui-ci avant d'arriver à la page testée...) devrait, à notre avis, améliorer fortement les mécanismes d'allocation.

Nous proposons ici une méthode d'allocation dynamique prenant en compte des informations temporelles sur les visiteurs, informations pouvant être mises à jour pendant la navigation. La section 2 présente plus en détails le contexte applicatif, les types des données observées et introduit le principe des bandits contextuels ainsi que les principales méthodes existantes. La section 3 décrit notre proposition de deux algorithmes innovants DBA-LinUcb et DBA-Ctree-Ucb. La section 4 présente les résultats expérimentaux. Enfin, la section 5 conclut et donne les perspectives à nos travaux.

2 Contexte applicatif et méthodes existantes

2.1 Comportements évolutifs et e-commerce

Dans (Pomirleanu et al., 2013) les auteurs proposent de différencier les visiteurs suivant leurs comportements sur le site. Quatre *profils-type* y sont définis à partir de l'observation du comportement des visiteurs sur un grand nombre de sites e-commerce différents et de leurs propensions à cliquer et/ou acheter :

— Les *prospects parfaits* savent exactement ce qu'ils recherchent, se dirigent naturellement vers les fiches produit et cherchent les meilleures conditions d'achats.
— Les *acheteurs potentiels* savent plus ou moins ce qu'ils veulent, leur besoin est clairement défini mais leur choix n'est pas encore établi avec certitude.

— Les *indécis* font en quelque sorte du « lèche vitrine ». Ils n'ont pas de besoin clairement défini mais peut-être achèteront-ils un article ?
— Ceux qui sont *arrivés par hasard* sur le site et qui n'ont pas l'intention d'acheter.

Les expériences ont montré que la sensibilité d'un test varie d'un profil-type à un autre, par exemple les *prospects parfaits* achètent quelle que soit la variation affichée, les *acheteurs potentiels* sont plus sensibles au design d'une page... Ainsi, la prise en compte des profils-type lors de l'allocation dynamique devrait permettre d'améliorer son fonctionnement. Pour déterminer ces comportements et donc les profils-types, l'utilisateur devra disposer d'informations temporelles comme par exemple :

— des données temporelles historisées produites avant l'arrivée du visiteur sur le site (données de navigations antérieures sur ce site ou d'autres sites comportant un historique des pages visitées, des achats effectués,...).
— Des données temporelles évolutives, générées dynamiquement à partir des actions effectuées par le visiteur entre son arrivée sur le site et son arrivée sur la page testée (clics, achats,...). En effet, dans le cadre des sites e-marchands, grâce aux informations collectées à la volée, le responsable du site peut connaître pour chaque visiteur le temps passé sur le site à chaque visite, le nombre et les pages visitées, ou encore les historiques d'achats depuis sa première connexion sur le site... Ces données forment le *parcours d'achat*.

Néanmoins la collecte de données temporelles historisées est légalement fortement limitée. Par exemple, le partage d'informations entre différents sites sans accord du visiteur est interdit. De fait, dans la pratique, les A/B-Tests effectués en e-commerce ne considèrent que des données temporelles évolutives pouvant être légalement et dynamiquement collectées par les sites testés. Dans la suite, avant de présenter notre proposition d'allocation dynamique autorisant des données temporelles, nous introduisons le concept de l'allocation dynamique contextuelle et non contextuelle.

2.2 Allocation dynamique

Pour limiter le coût dû à une allocation statique, une solution couramment mise en place est basée sur l'utilisation d'un *modèle de bandits* (Lai et Robbins, 1985; Gittins et Jones, 1974). Les bandits sont des algorithmes d'allocation dynamique dont l'objectif est de limiter le *regret*, c'est-à-dire la différence entre le gain maximal potentiel et le gain réel obtenu par l'utilisateur. Or, ce gain maximal correspond à l'affectation de tous les visiteurs à la variation optimale qui est inconnue a priori. Ce regret ne peut donc servir que de métrique de performance a posteriori. Dans la pratique, un algorithme de bandit cherche à maximiser le gain total obtenu à l'issue du test. Pour cela, l'algorithme choisit une variation dans l'ensemble des variations possibles pour le visiteur à son arrivée sur la page testée. La récompense obtenue est alors utilisée pour mettre à jour les estimations de gains. En effet, les propriétés d'une variation ne sont pas (ou que très partiellement) connues au début du test et ne peuvent être affinées qu'en allouant des visiteurs à celle-ci. L'avantage principal d'un tel algorithme est que l'allocation des visiteurs aux différentes variations est automatiquement ajustée au cours du temps en fonction des estimations afin d'affiner celles-ci (*exploration*) tout en maximisant le gain (*exploitation*). Or, de par la nature des algorithmes de bandit, la part d'exploration tend à disparaître au profil de l'exploitation, ce qui permet d'arrêter

l'exploration lorsque celle-ci n'est plus nécessaire. Lors du choix de l'algorithme de bandit, deux aspects sont à prendre en compte :

— Les caractéristiques des visiteurs sont-elles considérées ou non ? Un algorithme de bandit non contextuel cherchera à identifier la variation ayant la moyenne de gain la plus élevée. Un algorithme de bandit contextuel cherchera à identifier la variation ayant la moyenne de gains conditionnelle la plus élevée selon différentes caractéristiques des visiteurs ;

— Le temps de latence est-il une contrainte pour l'utilisateur ou non ? Dans notre contexte applicatif, l'algorithme de bandit fournit un choix qui aboutira sur l'affichage de la page web testée. Il est évident qu'un affichage ralenti pour le visiteur n'est pas acceptable (plus formellement, l'expérience visiteur ne doit pas être détériorée). Ainsi, le temps de latence, correspondant au temps nécessaire au bandit pour effectuer l'allocation, doit être inférieur à 100 millisecondes.

Nous nous plaçons ici dans le cas d'une maximisation du gain moyen final avec prise en compte des contextes visiteurs (bandit contextuel) en temps quasi réel (latence quasi nulle). Parmi les méthodes existantes, nous avons retenu deux algorithmes de bandit contextuel qui répondent à ces contraintes :

— Ctree-Ucb (Claeys et al., 2017) utilise un pré-groupement avant de démarrer l'expérience et adopte une stratégie d'allocation dynamique indépendante pour chaque groupe. Cet algorithme exploite les données de la variation originale (en production avant de démarrer le test) pour produire une fonction de classement prenant en entrée un visiteur (et ses caractéristiques) pour lui attribuer un groupe. Lors du test, chaque groupe est associé à un bandit non contextuel. Un nouveau visiteur est automatiquement classé dans un groupe puis le bandit associé au groupe lui choisit une variation.

— Lin-Ucb (Langford et Zhang, 2007) est un des algorithmes de bandit le plus utilisé. Cet algorithme repose sur une régression linéaire empirique, utilisant les contextes et les récompenses observés sur chaque variation. Lin-Ucb est un processus coûteux d'inversion d'une matrice de covariance M de taille $d \times d$ où d est le nombre de caractéristiques décrivant un visiteur.

2.3 Problématique

Les algorithmes introduits dans la section précédente présentent des restrictions fortes sur le format du contexte, celui-ci devant être présenté sous forme d'un vecteur à valeurs réelles ou catégorielles. De fait, ces restrictions limitent l'utilisation des algorithmes de bandit dans le cas où les caractéristiques à considérer portent sur des séries temporelles. Une solution pour contourner ce problème est alors d'aplatir les séries temporelles de chacune des caractéristiques temporelles. Deux approches sont alors possibles : choisir la dimension du contexte telle qu'elle puisse contenir tous les éléments différents dans toutes les séries ou concaténer la taille de toutes les séries de tous les visiteurs. Ces approches posent plusieurs problèmes : d'une part, ces contextes pourront s'avérer creux si le nombre d'éléments temporels différents est élevé ; d'autre part et surtout, le modèle du bandit contextuel repose sur une modélisation statistique empirique prenant en entrée un contexte pour fournir une probabilité de succès ou une

récompense moyenne. Or un contexte de grande taille introduit une grande variabilité sur les prédictions du modèle, ce qui nécessite de collecter beaucoup de données.

Pour répondre au problème de vecteur de contexte de très grande taille, Bastani et Bayati (2015) proposent de construire empiriquement un modèle statistique de type LASSO pour l'intégrer dans une problématique de bandit. Cependant, cette approche nécessite que les caractéristiques soient indépendantes entre elles, ce qui, de façon évidente, n'est pas le cas lorsque celles-ci sont issues d'une même série temporelle. De fait, il n'existe pas, à notre connaissance, d'algorithmes de bandits contextuels pouvant s'adapter à des données de taille variable. Une alternative possible est de réduire chaque caractéristique évolutive à une caractéristique catégorielle ou numérique représentative. C'est cette alternative qui caractérise notre approche.

3 Notre contribution : une allocation dynamique basée sur des informations temporelles

3.1 Principe

Dans notre approche, nous avons cherché à remplacer les séries temporelles, inexploitables telles quelles par un algorithme de bandit, par une caractéristique catégorielle. Idéalement, dans le cas du e-commerce par exemple, cette caractéristique pourrait représenter les profils-types d'e-marketing précédemment cités. Ainsi, une première solution pour remplacer chaque série temporelle par une caractéristique catégorielle, serait de demander à l'analyste de labeliser chaque série manuellement. Cette solution n'est malheureusement pas envisageable dès lors que le nombre de données devient important et qu'une allocation en temps réel est nécessaire. De plus, ces profils-types sont difficiles à mettre en évidence par l'expert car cela nécessiterait d'analyser manuellement les comportements passés des visiteurs afin d'en extraire des comportements caractéristiques. La deuxième solution consiste à remplacer la série par la moyenne des valeurs qu'elle contient. Or les expériences ont montré que la perte d'information était telle que les résultats s'en trouvent affectés (voir Section 4.2.1). Une troisième voie consiste à définir un mécanisme de remplacement en prétraitement du test basé sur l'identification automatique des *caractéristiques catégorielles représentatives* pour chacun des attributs temporels du contexte-visiteur à partir de données existantes (c'est-à-dire de toutes les séries temporelles présentes sur cette caractéristique) avant le test. Lors de la phase de test, ce mécanisme sera utilisé pour automatiquement remplacer les caractéristiques du visiteur testé par des caractéristiques catégorielles, ce contexte ainsi modifié étant directement utilisable par les méthodes classiques de bandits contextuels. Les *données d'apprentissage* nécessaires pour construire le mécanisme pourront par exemple, avoir été produites par une *variation originale* déjà en production. Dans la suite nous détaillons notre mécanisme de remplacement (Section 3.2) avant de décrire deux méthodes développées à partir de celui-ci (Section 3.3).

3.2 Des données temporelles à des données catégorielles

Le mécanisme de remplacement consiste de fait à résumer une information temporelle en une information catégorielle. Or ce problème est bien connu en fouille de

données. De fait, de nombreuses méthodes ont été proposées parmi lesquelles les approches basées sur le clustering qui ont prouvé leur efficacité dans ce cadre. Aussi, le mécanisme de remplacement que nous proposons se compose de deux phases :

1. une phase de clustering consistant à construire, en prétraitement, sur chacune des caractéristiques temporelles des visiteurs, un ensemble de clusters à partir de l'ensemble des séries temporelles présentes sur cette caractéristique dans les données existantes. Il est à noter que nous ne considérerons pas les récompenses obtenues car l'objectif ici n'est pas de la prédire mais d'identifier et de représenter des comportements différents.

2. une phase de catégorisation consistant à remplacer, lors du test lui-même, chaque caractéristique temporelle de chaque visiteur par une caractéristique catégorielle correspondant au label du cluster auquel appartient la série temporelle.

Concrètement, dans nos expériences, nous avons utilisé une approche de partitionnement basée sur la similarité entre les séries temporelles car elle est très fortement utilisée en fouille de données et y a fait ses preuves. Néanmoins, toute autre méthode permettant l'extraction de clusters pourrait être utilisée. Pour mettre en œuvre une telle méthode, il nous a été nécessaire de fixer une mesure de similarité (Agrawal et al., 1993). La plus connue est la distance euclidienne (Box et Jenkins, 1994) qui consiste à calculer la somme des carrés des distances des éléments constitutifs des séquences considérées à chaque pas de temps. Si cette distance est très utilisée, elle ne peut l'être dans notre cas, car elle impose que les séries aient toutes la même longueur. Or, dans nos travaux, nous nous plaçons dans le cas où l'observation des variables de contexte peut être irrégulière. Par exemple un visiteur peut se rendre plus fréquemment sur le site qu'un autre. La mesure de similarité *D.T.W. (Dynamic Time Warping)* est une métrique entre deux séries de tailles différentes largement reconnue comme pertinente pour de nombreux domaines d'application (Petitjean et al., 2011). D.T.W. construit un appariement des éléments des séquences afin d'aligner ces séquences. Cet ensemble d'associations respecte l'ordre total sur le séquencement des valeurs : autrement dit, les associations ne peuvent se croiser. Le lecteur peut trouver d'autres mesures selon les cas pratiques dans (Aghabozorgi et al., 2015).

Disposant d'une mesure de similarité, il est alors possible d'appliquer différents algorithmes de fouille de données. Parmi les méthodes existantes, K-Means est la plus utilisée. Cependant elle nécessite de disposer d'une méthode de calcul de la moyenne des observations (Niennattrakul et Ratanamahatana, 2007). La méthode de moyennage *D.B.A (Dtw Barycentric Averaging)* associée à D.T.W. définie par (Petitjean et al., 2011) a montré de bonnes performances dans ce cadre. D.B.A. a été conçue comme une méthode globale, en opposition aux stratégies par paire telles que NLAAF (Gupta et al., 1996). L'idée principale de D.B.A., inspirée de K-Means, repose sur un raffinement itératif d'une séquence moyenne initiale (initialisée arbitrairement), afin de minimiser sa distance quadratique aux séquences à moyenner. Cette méthode est aujourd'hui une des plus utilisées pour le clustering de séries temporelles basé sur D.T.W., nous l'avons donc choisie pour notre approche.

3.3 Deux nouvelles méthodes : DBA-Ctree-Ucb et DBA-LinUcb

Pour intégrer les séries temporelles dans les contextes visiteur, nous proposons deux algorithmes DBA-Ctree-Ucb et DBA-LinUcb extensions respectives de Ctree-Ucb et Lin-Ucb. Ces algorithmes imposent de disposer d'une variation originale A (par exemple la page web existante) et de données temporelles évolutives des visiteurs ayant été soumis à cette variation originale. Ces données constitueront les données d'apprentissage de DBA-LinUcb et DBA-Ctree-Ucb.

Une première étape en pré-traitement du test, consiste à effectuer le clustering de séries temporelles via K-Means basé sur D.T.W./D.B.A afin de déterminer un ensemble de clusters pour chacune des caractéristiques temporelles. Si l'algorithme sélectionné est DBA-Ctree-Ucb, une étape intermédiaire construit la fonction de classement des nouveaux visiteurs dans des groupes. La deuxième étape (en ligne) correspond à l'A/B-Test lui-même. Elle consiste à remplacer, pour chaque visiteur soumis au test, les caractéristiques temporelles par des caractéristiques catégorielles grâce au mécanisme de remplacement décrit précédemment. Le visiteur et son nouveau contexte sont alors traités par l'algorithme sélectionné.

4 Cadre expérimental

4.1 Les données

Nous utilisons dans nos expérimentations un jeu de données réelles (*base de données originale*), collectées à partir d'un A/B-Tests par allocation statique réalisé par la société AB Tasty sur une page d'un site de vente dans le prêt à porter. Les données portant sur 11 389 visiteurs, incluent des données temporelles évolutives captées lors des parcours d'achat des différents visiteurs qui ont navigué sur le site web et ont vu la page test concernée au moins une fois : de sa première visite jusqu'à son arrivée sur la page test, sa navigation est enregistrée. Lorsqu'il arrive sur la page test, une variation (A ou B) lui est attribuée (aléatoirement). Chaque fois qu'un visiteur revient sur la page testée, la même variation lui est affichée (irrévocabilité de l'allocation). Si un visiteur achète après avoir vu la page test, la récompense est de 1, 0 sinon. À chaque visiteur sont associées trois séries temporelles, de taille identique égale au nombre de jours entre sa première et sa dernière visite de la page testée :

- `presence_time_serie` : constituée de valeurs binaires. Pour chaque jour où le visiteur s'est rendu sur le site, une valeur 1 est définie, 0 sinon. Chaque série commence et finit donc par la valeur 1.
- `connexion_time_serie` : constituée de valeurs entières comprises entre 0 et 24. Pour chaque jour où le visiteur s'est rendu sur le site, une valeur égale à son heure d'arrivée sur le site est définie, 0 sinon. Si le visiteur vient plusieurs fois dans la même journée, seule la première heure est prise en compte.
- `time_spend_serie` : constituée de valeurs comprises dans $\mathbb{R}$. Pour chaque jour où le visiteur s'est rendu sur le site, une valeur égale à sa durée d'activité moyenne pendant sa visite (en microsecondes) sur le site est définie, 0 sinon.

Chaque série temporelle est issue de la séquence de *logs* journaliers associée aux différentes sessions d'un visiteur de sa première visite sur le site à la dernière qui inclue la première arrivée sur la page test. Sa longueur est donc égale au nombre de jours

entre la première visite et la date d'arrivée sur la page. De fait, les tailles des séries temporelles peuvent être différentes suivant les visiteurs.

4.2 Confrontation des méthodes

Pour vérifier si l'intégration des séries temporelles améliore notre modèle de bandit, nous comparons DBA-LinUcb et DBA-Ctree-Ucb avec d'une part, Lin-Ucb et Ctree-Ucb en réduisant chaque série temporelle à sa moyenne et d'autre part, avec uniform qui correspond à une allocation statique.

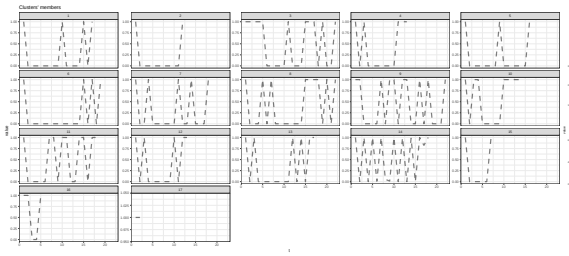
Nous avons testé avec différentes configurations (nombre de clusters, taille du jeu de données d'apprentissage...) de nos algorithmes [1] . Utiliser le regret pour comparer les résultats nécessite de connaître, pour chaque visiteur, la récompense obtenue pour toutes les variations ce qui est impossible par définition (un visiteur ne voit qu'une seule variation). Pour pallier ceci, il serait possible de remplacer les récompenses manquantes par une moyenne sur l'ensemble de visiteurs, mais cela pourrait biaiser fortement les résultats [2]. Nous avons donc décidé de comparer uniquement les gains moyens obtenus à la fin du test pour chacune des configurations. Les visiteurs de la base de données initiale ont été utilisés pour simuler chacun des tests. Pour cela, pour chaque visiteur, l'algorithme choisit une affectation. Si celle-ci correspond à celle qui a été faite dans la réalité, le visiteur, le choix et la récompense associés sont pris en compte dans l'expérience. Sinon, le visiteur est rejeté et le suivant dans le base de données est soumis à l'algorithme (*rejection sampling*).

La performance des approches se fait par rapport aux taux moyens de transactions (Nombre d'acheteurs/Nombre de visiteurs). Ces performances sont comparées selon la taille du jeux de données d'apprentissage (nécessaire pour apprendre les clusters) et les nombres de clusters fixés en amont du test. Le *rejection sampling* diminuant le nombre de visiteurs testés (certains seront ignorés par l'algorithme), nous proposons de répliquer cinq fois le jeu de données. Avant d'observer les performances des algorithmes proposés, nous avons effectué au préalable une étude sur l'interprétabilité des clusters obtenus lors de la première étape.

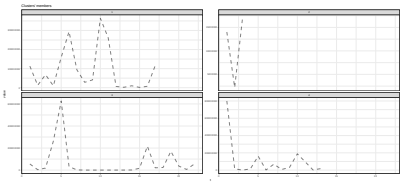
Interprétabilité des clusters Nos expériences ont été menées avec deux configurations : les clusters sont appris sur 30% ($\text{Conf}_{\text{clust30,70}}$) ou 100% ($\text{Conf}_{\text{clust100,100}}$) de la base de données [3]. Dans chacune de ces configurations, le choix des nombres de clusters a été fait en fonction de critères de qualité usuellement utilisés en clustering. Dans la Fig. 1, nous présentons graphiquement les centroïdes des clusters obtenus pour la configuration $\text{Conf}_{\text{clust30,70}}$ avec $Nb_{\mathtt{p}} = 17$, $Nb_{\mathtt{t}} = 4$ et $Nb_{\mathtt{c}} = 15$ où $Nb_{\mathtt{p}}$ (resp. $Nb_{\mathtt{t}}$ et $Nb_{\mathtt{c}}$) est le nombre de clusters pour la caractéristique `presence_time_serie` (resp. `time_spend_serie` et `connexion_time_serie`).

En accord avec les profils-type décrits dans la section 2.1, avec une configuration $\text{Conf}_{\text{clust30,70}}$, nous pouvons identifier que les comportements d'indécis sont par exemple représentés par les clusters #3, #9 ou #14 de la série `presence_time_serie`

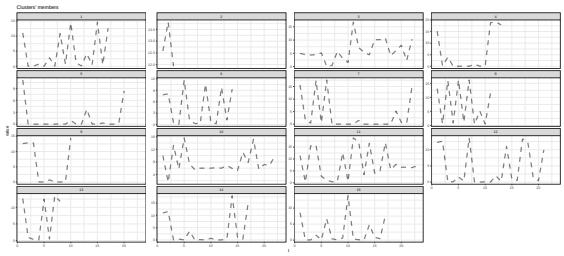
1. Code `R` et données disponibles à l'adresse `https://github.com/manuclaeys/TimeSeriesBandits`
2. un modèle permettant de remplacer des valeurs manquantes en fonction de séries temporelles de taille variable étant, comme nous l'avons vu en section 2.3, difficile à construire
3. Le choix d'un apprentissage sur 30% des données est ici motivé par (Claeys et al., 2017)



(a) `presence_time_serie` : $Nb_p = 17$



(b) `time_spend_serie` : $Nb_t = 4$



(c) `connexion_time_serie` : $Nb_c = 15$

FIG. 1: Centroïdes des clusters

(voir Fig.1a). En effet, ces groupes font des visites de plus en plus rapprochées. En revanche, nous pouvons supposer qu'avec des visites plus espacées, comme celles représentées par clusters #6, #12 ou #13 de la série `presence_time_serie` les visiteurs sont des acheteurs potentiels prenant le temps de la réflexion. A contrario, les visiteurs appartenant au groupe #17 de la série `presence_time_serie` ne sont, soit pas intéressés par le site, soit déjà engagés dans leur processus d'achat. Les clusters de la série `connexion_time_serie` permettent d'avoir une idée sur les horaires de connexion. Des visiteurs appartenant aux clusters #7 ou #8 par exemple se rend régulièrement vers 15h sur le site. Il est intéressant d'observer que les combinaisons de clusters pour chaque visiteur permettent de différencier un visiteur arrivé par erreur d'un prospect parfait. En effet les prospects parfaits et ceux qui sont arrivés là par hasard sur le site sont difficilement distinguables en observant uniquement leurs parcours dans la série `presence_time_serie`. Leurs visites sont rapides et ils ne reviennent pas forcément sur le site. En analysant le temps passé sur le site `connexion_time_serie`, il est par contre possible de distinguer ces deux profils (un visiteur arrivé par erreur quittera immédiatement le site). De la même façon, les acheteurs potentiels et les indécis peuvent revenir à la même fréquence sur le site. Pour les distinguer, l'analyste peut supposer que ceux qui reviennent à la même heure régulièrement peuvent être considérés comme ceux visitant le site pour le loisir (Roukine, 2011). À noter que nos expériences ont montré que les centroïdes obtenus avec $\text{Conf}_{\text{clust}30,70}$ étaient globalement très similaires à ceux obtenus avec $\text{Conf}_{\text{clust}100,100}$.

4.2.1 Étape d'exploration dans l'A/B-Tests (Allocation dynamique)

Pour des raisons de lisibilité et d'interprétation des résultats, nous scindons en deux le compte rendu de nos expérimentations. La première partie compare DBA-Ctree-Ucb et Ctree-Ucb, la deuxième partie compare DBA-LinUcb et Lin-Ucb. Nous comparons ici les moyennes des gains moyens (taux de transactions) des algorithmes précédemment cités, sur 200 exécutions. Nous pouvons ainsi observer l'influence des nombres de clusters Nb_p, Nb_t et Nb_c sur les résultats. Afin que les clusters soient interprétables par l'utilisateur, nous avons limité le nombre maximal de clusters pour chacun des paramètres.

DBA-Ctree-Ucb vs Ctree-Ucb : Le taux de transactions obtenu par simulation d'uniform (c'est-à-dire en allocation statique) à partir des données originales est de 11,74%. Dans les tests nous avons fait varier les nombres de clusters pour estimer la sensibilité de DBA-Ctree-Ucb à ces paramètres (Tab. 1). La Tab.1a, présente les résultats obtenus pour différentes configurations (Nb_p, Nb_t, Nb_c) sur les données initiales. On voit que les taux de transactions obtenus par DBA-Ctree-Ucb dépassent systématiquement ceux de Ctree-Ucb basé sur la moyenne des séries. Cependant, il est possible que les performances des algorithmes aient été biaisées par le *rejection sampling*. Des résultats complémentaires sont donc présentés dans la Tab.1b où les données ont été dupliquées pour représenter 5 fois le jeu de données initial. Dès lors, les performances diminuent mais le paramétrage ($Nb_p = 17$, $Nb_t = 3$ et $Nb_c = 3$) reste globalement un des meilleurs que ce soit avec ou sans duplication du jeu de données.

Tab. 1: Sensibilité au clustering DBA-Ctree-Ucb vs Ctree-Ucb (AB Tasty #7)

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
Nb_p ;Nb_t ;Nb_c				
5 ;3 ;3	13,34%	13,56%	12,94%	12,88%
5 ;4 ;15	13,96%	14,12%	11,90%	12,23%
11 ;3 ;3	14,15%	14,10%	14,62%	14,58%
11 ;4 ;15	13,85%	13,35%	13,85%	13,58%
17 ;3 ;3	14,23%	**15,46%**	**15,27%**	**15,34%**
17 ;4 ;3	**14,26%**	13,60%	14,87%	13,73%
Ctree-Ucb	11,40%	11,71%	12,24%	12,77%
uniform	11,74%	11,74%	11,74%	11,74%

(a) Données initiales

	$\text{Conf}_{30,70}$ et $\text{Conf}_{\text{clust}30,70}$		$\text{Conf}_{100,100}$ et $\text{Conf}_{\text{clust}100,100}$	
DBA-Ctree-Ucb	$V_A = P_1$	$V_A = P_2$	$V_A = P_1$	$V_A = P_2$
Nb_p ;Nb_t ;Nb_c				
5 ;3 ;3	12,76%	12,31%	12,58%	**12,59%**
5 ;4 ;15	12,53%	12,47%	**12,61%**	**12,59%**
11 ;3 ;3	12,49%	12,45%	12,55%	12,57%
11 ;4 ;15	12,23%	12,14%	12,49%	12,46%
17 ;3 ;3	**12,76%**	**12,49%**	12,53%	12,48%
17 ;4 ;15	12,46%	12,43%	12,41%	12,76
Ctree-Ucb	12,28%	12,15%	12,28%	12,14%

(b) Données dupliquées

DBA-LinUcb vs Lin-Ucb : De façon similaire aux expérimentations précédentes, la Tab. 2 présente les résultats de DBA-LinUcb et Lin-Ucb (n'utilisant que la moyenne) obtenus avec les données initiales (Tab. 2a) et avec duplication (Tab. 2b). Les taux de transactions obtenus par DBA-LinUcb dépassent Lin-Ucb si le bon nombre de clusters est choisi en amont. Cependant, les plus mauvaises performances de DBA-LinUcb restent très proches de celles obtenues par Lin-Ucb. Comme pour DBA-Ctree-Ucb le paramétrage ($Nb_{\mathtt{p}} = 17$, $Nb_{\mathtt{t}} = 3$, $Nb_{\mathtt{c}} = 3$) est optimal avec $\text{Conf}_{\text{clust30,70}}$ sans duplication. Pour les autres configurations, c'est le paramétrage ($Nb_{\mathtt{p}} = 5$, $Nb_{\mathtt{t}} = 4$, $Nb_{\mathtt{c}} = 15$) qui maximise le gain moyen. Notre hypothèse est que DBA-LinUcb est très dépendant du nombre de clusters et qu'un nombre de clusters important implique un apprentissage plus long dû à la modélisation des paramètres de la fonction linéaire.

Tab. 2: Sensibilité au clustering DBA-LinUcb vs Lin-Ucb (AB Tasty #7)

DBA-LinUcb Nb_p ;Nb_t ;Nb_c	$\text{Conf}_{\text{clust30,70}}$	$\text{Conf}_{\text{clust100,100}}$
5 ;3 ;3	12,92%	12,61%
5 ;4 ;15	12,77%	**12,76%**
11 ;3 ;3	12,70%	12,42%
11 ;4 ;15	12,93%	12,44%
17 ;3 ;3	**13,07%**	12,65%
17 ;4 ;15	12,61%	12,70%
Lin-Ucb	12,43%	12,43%

(a) Données initiales

DBA-LinUcb Nb_p ;Nb_t ;Nb_c	$\text{Conf}_{\text{clust30,70}}$	$\text{Conf}_{\text{clust100,100}}$
5 ;3 ;3	11,93%	12,01%
5 ;4 ;15	**12,77%**	**12,66%**
11 ;3 ;3	12,45%	12,06%
11 ;4 ;15	12,51%	12,07%
17 ;3 ;3	12,20%	12,59%
17 ;4 ;15	12,50%	12,17%
Lin-Ucb	12,57%	12,57%
uniform	11,74%	11,74%

(b) Données dupliquées

5 Conclusion

Nous avons présenté ici la méthode innovante DBA-Ctree-Ucb d'allocation dynamique autorisant l'utilisation de données temporelles. Les résultats sont prometteurs en termes de gain moyen par rapport aux autres méthodes. Grâce à sa pré-segmentation utilisant un arbre d'inférence, la méthode peut regrouper, dans un même groupe, les visiteurs ayant eu un même comportement sur une variation originale existante. Ces groupes sont pour l'utilisateur, une source d'informations intéressante pour extraire des profils-type de visiteurs. DBA-Ctree-Ucb s'avère relativement peu sensible aux différents paramètres. De plus, un apprentissage des clusters sur 30% de la base de données original est suffisant pour obtenir des résultats comparables à un apprentissage total. Des expériences supplémentaires sont actuellement en cours pour confirmer cette hypothèse. De même, des études approfondies sur l'influence du *rejection sampling* sur les performances sont également en cours et présentent des résultats très encourageants. Enfin, seule C.H.A ayant été comparée (et non retenue), des études complémentaires devront être entreprises afin d'évaluer d'autres algorithmes de clustering avec, éventuellement, d'autres mesures de similarité.

Références

Aghabozorgi, S., A. S. Shirkhorshidi, et T. Wah (2015). Time-series clustering - a decade review. *Information Systems 53*.

Agrawal, R., C. Faloutsos, et A. N. Swami (1993). Efficient similarity search in sequence databases. In *Proc. 4th International Conference on Foundations of Data Organization and Algorithms*, FODO '93, London, UK, UK, pp. 69–84. Springer-Verlag.

Bastani, H. et M. Bayati (2015). Online decision-making with high-dimensional covariates. *SSRN Electronic Journal*.

Box, G. E. P. et G. M. Jenkins (1994). *Time Series Analysis : Forecasting and Control* (3rd ed.). Upper Saddle River, NJ, USA : Prentice Hall PTR.

Claeys, E., P. Gançarski, M. Maumy-Bertrand, et H. Wassner (2017). Regression tree for bandits models in a/b testing. In *IDA*.

Gittins, J. et D. Jones (1974). A dynamic allocation index for the sequential design of experiments. In J. Gani (Ed.), *Progress in Statistics*, pp. 241–266. Amsterdam : North-Holland.

Gupta, L., D. Molfese, R. Tammana, et P. Simos (1996). Nonlinear alignment and averaging for estimating the evoked potential. *IEEE transactions on bio-medical engineering 43*, 348–56.

Lai, T. et H. Robbins (1985). Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics 6*(1), 4 – 22.

Langford, J. et T. Zhang (2007). The epoch-greedy algorithm for multi-armed bandits with side information. In *NIPS*.

Lattimore, T. et C. Szepesvári (2019). *Bandit Algorithms*. Cambridge University Press.

Niennattrakul, V. et C. Ratanamahatana (2007). On clustering multimedia time series data using k-means and dynamic time warping. pp. 733–738.

Petitjean, F., A. Ketterlin, et P. Gancarski (2011). A global averaging method for dynamic time warping with applications to clustering. *Pattern Recognition 44*, 678–.

Pomirleanu, N., J. Schibrowsky, J. Peltier, et A. Nill (2013). A review of internet marketing research over the past 20 years and future research direction. *Journal of Research in Interactive Marketing 7*.

Roukine, S. (2011). *Améliorer ses taux de conversion web : Vers la performance des sites web au-delà du webmarketing*.

Zhou, L. (2015). A survey on contextual multi-armed bandits. *CoRR abs/1508.03326*.

Summary

Recently promising new methods have been developed to optimize e-commerce A/B testing using dynamic allocation. They provide faster results to determine the best variation and reduce test costs. However, dynamic allocation by traditional methods of reinforcement learning is restrictive on the type of data used : time series cannot be considered as a context. In this paper, we present two new methods, based on a common approach, that allow time series to describe visitors. Our numerical results on data from real tests leads to an improvement on dynamic allocation apply to A/B testing.