Dynamic allocation optimization in A/B tests using classification-based preprocessing

Emmanuelle Claeys, Pierre Gancarski, Myriam Maumy-Bertrand, Hubert Wassner

Dynamic allocation optimization in A/B-Tests using classification-based preprocessing

Emmanuelle Claeys, Pierre Gançarski, Myriam Maumy-Bertrand and Hubert Wassner

Abstract—An A/B-Test evaluates the impact of a new technology by running it in a real production environment and testing its performance on a set of items. Recently, promising new methods are optimising A/B-Tests with dynamic allocation. They allow quicker determination of which variation (A or B) is best, saving money for the user. However, dynamic allocation by traditional methods requires certain assumptions, which are not always valid in reality. This is often due to the fact that the populations being tested are not homogeneous. This article reports on a new reinforcement learning methodology which has been deployed by the commercial A/B-Test platform AB Tasty. We provide a new method that not only builds homogeneous groups of users, but also allows the best variation for these groups to be found in a short period of time. This article provides numerical results on AB Tasty's data, in addition to public datasets, to demonstrate an improvement over traditional methods.

Index Terms—A/B-Test, Bandit strategies, UCB strategies, Conditional inference tree, Non linear bandit, Regret minimisation.

1 Introduction

In a lot of economic, industrial, and even social fields it can be interesting to evaluate the relevance of modifications to an existing entity according to one or more objectives by directly and concretely comparing the different variations resulting from the modifications. For instance, an e-marketing team can look for the best modification to apply to a given web page to increase sales. A medical laboratory may want to find the best drug composition modification to save more patients. A company may want to define the best modification to an industrial process to increase product quality. Such a task requires a mechanism to evaluate each variation in order to make the optimal choice according to a defined objective in the given context. *A/B-Test* based approaches have been proposed to respond to this problem [1] and have recently generated renewed interest, particularly from their use in e-marketing. An *A/B-Test* consists of affecting the *items* (patients, visitors, goods to be produced . . .) to the different variations in order to evaluate the relative performance of each. During this *exploration phase*, it is assumed that the result, called *reward*, of each *affectation* can be observed and used by the *A/B-Test* algorithm to evaluate each variation's performance. At the end of this exploration phase the *user* can better decide which variation will replace the current entity to be used in the future (i.e., in *production phase*) according to their relative performance.

An important characteristic of such methods is that the decision to assign an item to a variation is irrevocable. For instance, for the entire duration of the test, a visitor will always see the same web page on each of their visits, regardless of the number of visits. Thus, it is impossible to know what the visitor would have done if they had been assigned to another variation. Consequently, the population that has been affected to a variation is distinct from those affected to any other. Finally, it is assumed that items are unaware of their participation in a test and thus the existence of different variations.

A classical approach to the exploration phase is referred to as the frequential approach and consists of assigning items to the different variations according to explicit predefined ratios (*static allocation*) for a predefined period of time. If the ratios are balanced for each variation, the duration is unfortunately difficult to define *a priori*. Experiments have shown that a user tends to overestimate the duration, causing an inferior variation to have a large detrimental effect on the result for a long period of time. In this case, the obtained cumulative reward will be much lower than that which would have been produced by the allocation of each item to the optimal variation. This difference, called *regret*, increases with negative impact. Reducing the exploration phase may reduce regret, but may also lead to a lack of data needed to calculate performance. Therefore, in addition to determining the best option, the challenge of A/B-Test methods is to also minimize regret. Nevertheless, it is important to note that regret cannot be calculated during the observation phase as the optimal variation is obviously unknown *a priori*: the objective of the test is, by definition, to determine it. Finally, in most cases (e.g. time is money, people continue to die, etc.) the sooner the algorithm finds the solution (i.e. the sooner exploration can then be stopped), the better.

To address this problem, new A/B-Test methods perform dynamic allocation of items based on bandit algorithms. Bandit *dynamic allocation* consists of adapting the allocation of visitors according to the obtained rewards and thus gradually tipping the visitors towards the optimal variation. This dynamic allocation is usually achieved using the probabilistic comparison criteria of each variation's empirical reward dis-

- *E. Claeys and P. Gançarski are with ICube Laboratory, University of Strasbourg, 300 Bd. Sébastien Brant, 67400 Illkirch-Graffenstaden.*
E-mail: claeys@unistra.fr
- *E. Claeys and M. Maumy-Bertrand are with the IRMA lab, University of Strasbourg, 7 Rue René Descartes, 67084, Strasbourg.*
- *H. Wassner is with AB Tasty, 3 Impasse de la Planchette, 75003, Paris.*

Manuscript received XXX ; revised XXXX.

tribution. The idea is to maintain and update each variation's gain estimate and to allocate items according to them. It is therefore a matter of favoring the most promising variation while continuing to refine the remaining gain estimates by continuing to allocate items to potentially sub-optimal variations.

Experiments and theoretical studies have shown that dynamic allocation [2] provides better results in terms of cumulative regret, as well as being faster at determining the best variation. In this context, a lot of methods implementing dynamic allocation based on bandit algorithms have been proposed [3], [4], [5] and have proved their ability to find optimal variations in the general case. Nevertheless, experiments also show that these methods often fail when the reward obtained by an item depends on both the variation and the item itself [6]. For instance, in web marketing, visitors naturally tend to click and buy differently according to their own financial resources or their geographical localization. In medical treatment, the efficiency of a drug often depends on the age and/or gender of the patient. To address this problem, bandit algorithms have been extended to form *contextual bandits*, which take into account each visitor's *context*, i.e. their characteristics (age, origin, sex, etc.) when allocating them in order to perform more relevant allocations. Methods such as KernelUCB [7] and LinUCB [8] (see Section 3.2.1) have demonstrated not only the benefits of such an approach, but also their limitations, including in particular:

- large latency (corresponding to the time required by the algorithm to allocate an item to a variation),
- the need for a large number of items before finding the optimal variation,
- a lack of explainability of the affectations made by the algorithm.

These limits strongly reduce their practical use.

Furthermore, it is obvious that in many cases items belong to natural groups (e.g., social classes, levels of study, age classes, . . .) for which each variation's reward distribution can differ. For instance, for a given web page students may behave differently to workers or retired people and, in fact, can be differently impacted by a modification. Unfortunately, these groups are often very difficult to determine because they strongly depend on the application domain and on the test itself.

In this paper we propose an original A/B-Test method called Ctree-Ucb which, instead of using a contextual bandit, is based on the use of several non-contextual bandits, each dedicated to a particular group of items. Our proposal consists of automatically creating homogeneous in a preprocessing step using a conditional inference method. These groups are created according to the objective of the test using information (obtained rewards, item characteristics, temporal information, etc.) derived from items subjected to an existing variation in production phase before the test. Then, in the exploration phase, a non-contextual bandit is dedicated to a group and is used to find the optimal variation associated with the group. To achieve this, each new item submitted to the A/B-Test is classified into a group before being transmitted to the associated bandit for its allocation to a variation.

The remainder of this paper is organized as follows. Section 2 presents the bandit model with an illustrative example. Based on this example, Section 3 gives a comprehensive literature review of existing approaches and focuses on contextual strategies able to take into account the characteristics of the items. Section 4 details the proposed methods. Sections 5, 6 and 7 analyse and discuss the results obtained with this method on real data provided by AB Tasty [1]. Finally the conclusions of the study are drawn in Section 8.

In sake of readability, the remainder of this article focuses on A/B-Tests with only two variations but all our propositions are directly and easily extended to tests with more variations.

2 Bandit problem

2.1 The multi-armed bandit model

The first definition of the multi-armed bandit model was introduced by Lai and Robbins [9], by an analogy with casino slot machines. For a player, it is a matter of choosing from a machine with several arms the one presenting, for him, the best expectation of gain. To do that, each time the player plays an arm and reaped (or not) the gain, he/she updates the gain estimates of the arm. The player's goal is to find the best arm, called the *optimal arm*, while limiting the number of tries. As introduced in Section 1, bandit-based approaches are frequently chosen to concretely implement dynamic allocation: at each iteration t, corresponding to the arrival of an item c_t, the bandit algorithm chooses an arm a in the set of possible arms $\mathcal{A}$ according to its own strategy π. Then, the reward $X_{c_t,a=A_t}$ obtained by the affection of the item to the chosen arm a is observed. The main characteristic of strategy π is that the allocation of the items depends on the reward expectation of the variations.

2.2 The bandit paradigm as reinforcement learning

In the general case, an A/B-Test (and more specifically the bandit algorithm) can be seen as an agent with partial knowledge of the world (the different variations, the items having been subjected to these variations, and the rewards obtained so far). Knowledge of this world is very sparse at the beginning of the test but is reinforced by observing the rewards of each variation in accordance with the context of each item. When the agent has identified the best variation according to the characteristics of the visitors, the user can:

- put one of the variations (A or B) into production,
- compose another variation to be tested.

Note that in our case, there is one bandit for each group. So, in fact there are several agents.

Initially, the bandit does not know anything about the distribution of the rewards of each arm. It has to *explore* to find it by affecting items to the different arms to learn these distributions with the risk of accumulating less reward. But, at the same time, it has to *exploit* by affecting the arm which it estimates to be the most rewarding with the risk of not discovering the optimal arm. This well-known exploration-exploitation dilemma has been extensively studied through the multi-armed bandit problem in [10].

The first mention of the bandit problem appears in [1]. This paper presents a reinforcement learning problem where

1. https://www.abtasty.com

an autonomous agent must learn the actions to be taken from experience in order to optimize a quantitative reward over time (per the definition of reinforcement learning). The agent evolves in an environment and makes decisions based on its current state. In return, the environment provides a reward, which can be positive or negative. The agent seeks an optimal decision-making behaviour (called strategy or policy, which is a function associating the action to be performed with the current state) through iterative experiences in the sense that it maximizes the sum of rewards over time. As the definition of reinforcement learning: "agents ought to take actions in an environment in order to maximize some notion of cumulative reward" [11]. The focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

2.3 Cumulative regret

Before presenting the different strategies integrated into bandit algorithms, we introduce here the main criterion used to evaluate them.

Let $\mathcal{A}$ be the set of possible arms (with $|\mathcal{A}| \in \mathbb{N}^+$) and a^* the optimal arm. The *cumulative regret* is defined as the sum, over all the items, of the difference between the rewards that would have been obtained with a^* and those actually obtained with the chosen arm. Let $X_{A_t,t}$ be the reward obtained with arm A_t selected at iteration t (corresponding to the t-th item). The cumulative regret after n iterations is defined by:

$$R_n = \sum_{t=1}^{n} \max_{a \in \mathcal{A}} [X_{a,t}] - \sum_{t=1}^{n} X_{A_t,t}, \quad (1)$$

and r_t is the simple regret from the t-th iteration defined by:

$$r_t = \max_{a \in \mathcal{A}} [X_{a,t}] - X_{A_t,t}. \quad (2)$$

As the arm a^* is unknown *a priori*, the cumulative regret can only be calculated after the end of the test. Moreover, each decision to assign an item to a variation is irrevocable. Thus, this calculation can only be done if for each item that has not been affected to a^*, the reward potentially obtained with an affection to a^* can be known (or at minimum be correctly estimated). The reader can find more information about theoretical upper bound of the cumulative regret in the appendix.

3 State of the art

Three characteristics can discriminate the different strategies:

- All the strategies π are based on the strong hypothesis that the distribution of all arm rewards follow the same law (Bernoulli distribution with THOMPSON SAMPLING [1], [12], Gaussian with UCB [13]) or otherwise makes no assumptions.
- Two mechanisms to affect items can be defined. The first qualifies as non-informative as it uses no information about items (only rewards are used to make a choice), while the second qualifies as contextual as it considers item characteristics when affecting to a variation.
- Different mathematical models can be used in the choice mechanisms such that the best arm (based on previous observations) is chosen according to predefined probabilities (such as the EPSILON-GREEDY algorithm [14], [15]) or using adaptive probabilities (for example SOFTMAX EXPLORATION [16]).

3.1 Non informative strategy

A *non informative strategy* assumes that the best arm is the same for all (or at least, for the majority of) items and so arm is allocated independently of the characteristics of the item.

3.1.1 UCB *strategy*

The UCB method is a non informative method based on an optimistic Bayesian strategy (with probabilistic upper bounds of the real average). The principle of its strategy π is to use an overestimation of the empirical average $\hat{\mu}_{a,t}$ for each arm a, the total number of items, and their allocation to different arms to assign a new item to an arm. Concretely, an arm is chosen if it is promising (because its estimated average is high) or/and seldom explored (see Algo. 1 where $T_a(t)$ is the number of times arm a has been chosen[2]).

Algorithm 1 UCB algorithm

```
Require: $\alpha > 0$
Require: Assign at least one iteration to each arm $a$
1: loop
2:     $c_t \leftarrow$ a new iteration
3:     $A_t = \operatorname{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,t} + \alpha\sqrt{\frac{2 * \log(t)}{T_a(t)}}\}$
4:     Assign arm $A_t$ to $c_t$
5:     $X_{A_t,c_t} \leftarrow$ the arm $A_t$ reward
6:     Update $\hat{\mu}_{A_t}$ and $T_{A_t}(t)$
Output: A sequence of arm choices $(A_t)$ and rewards $X_{A_t,c_t}$
```

In fact, the $\hat{\mu}_{A_t}$ estimators may not be relevant at the beginning of the test, due to the small number of items considered [17]. To get around this difficulty, [13] proposes to calculate an overestimation of this average, called the *upper confidence bound*. The authors justify their proposition by a policy known as "optimistic in the face of uncertainty" and demonstrate good results.

This upper bound is the sum of the reward's empirical average obtained so far in addition to an exploration bonus (also known as the confidence interval). It depends on the number of items assigned and observed. The more observations an arm makes, the more the arm's bonus decreases. If ν_a is Gaussian for all a, this bound will always be higher than the real average. Thus, the authors define the upper bounds of each arm by:

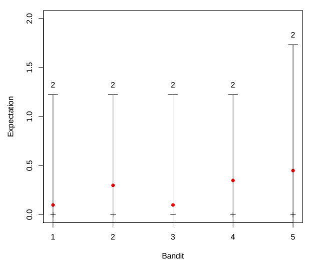
$$Upper_{\text{UCB}}(a,t) = \alpha\sqrt{\frac{2 * \log(t)}{T_a(t)}}, \quad (3)$$

where α is a positive real parameter given by the user. In the initial version of UCB, $\alpha = 1$ but in practice it has been shown that the optimal choice of this value depends on the arm distributions [18].

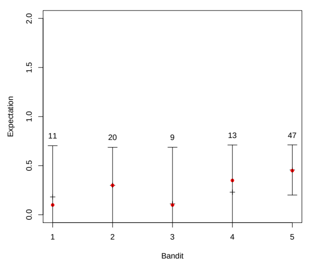
2. Note that α is different from the α risk commonly used in statistics.

The π algorithm consists of choosing the arm with the highest upper bound. After each assignment, $\hat{\mu}_{A_t}$ is updated and its bound is reduced, see Equation (3). As the confidence interval depends on $T_a(t)$ the higher $T_a(t)$, the less the overestimation: the overestimation of the chosen arm's average decreases towards its real average. The upper bounds of the unchosen arms remain unchanged. The reader can find a theoretical proof of convergence in the appendix.

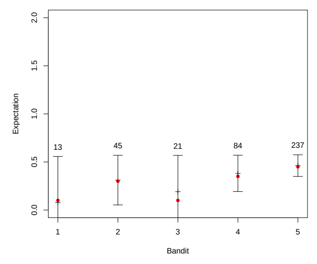
Figure 1 shows an example of the confidence bound evolution for five arms according to the number of submitted items.



(a) After 10 items tested



(b) After 100 items tested



(c) After 400 items tested

Figure 1: Confidence bound evolution according to number of times an arm has been chosen by UCB (red (black) points correspond to real (estimated) averages).

3.1.2 *Limit of* UCB *strategy*

The UCB algorithm is very efficient when the real distribution of rewards is Gaussian, an assumption that can be verified retrospectively using static allocation. Unfortunately, experiments have shown that this assumption is rarely valid. In fact, the upper bound is not reliable. Consequently, UCB requires more items to find a^* [19], [20]. Moreover, if the reward presents extreme values, the convergence can be very long nevertheless, it has been proved that the a^* will eventually be found. As such, despite these imperfections, since UCB focuses on the average reward that leads to a low complexity and since it is generally well understood by the user of an A/B-TEST, overall it responds to the problems of interpretability and limited computation time.

The remaining problem is that the identification of a^* may not be the only objective of the A/B-TEST. Rather than looking for the variation that maximises the gain on average, the A/B-TEST user can instead look for the best variation according to different sub-populations (as quickly as possible).

Indeed, suppose that there exists two sub-groups of items having different responses to the test. For instance, in the medical field, an alternative treatment may be efficient for the elderly and not for young people. In marketing, a page can be optimal only for smartphone users. In this case, ν_a is very far from a Gaussian distribution. Thus, recent approaches assume that ν_a is a mixture of Gaussians especially in contextual-based strategies. The idea of the contextual strategy (presented below) is that the reward X_{A_t,c_t} depends on both the arm assigned and an item's characteristics (features).

3.2 Contextual strategy

Contextual approaches assume that there exists sub-groups of items, each presenting a different reward distribution. Nevertheless, experiments show that it can be difficult to define such groups. Asking the user to define them is often unproductive, as they rarely have a clear vision of these different groups. To overcome this problem, it is assumed that there are *a priori* unknown links between, on one hand the *context* of the items (i.e., the characteristic vectors describing the items [21]) and the groups, and on the other hand, the groups and the averages of the rewards obtained. To fit this link, two approaches can be considered:

- In *contextual bandit*, this link is modelled by a unique regression function: the groups and their associated average rewards are directly set by the bandit during the test (see Section 3.2.1).
- In two step-based approaches, the groups are set using a pre-processing step of the A/B-TEST (see Section 3.2.2).

3.2.1 *Contextual bandits*

In contextual bandits, rewards are assumed to be generated from an unknown function depending on the item's features (characteristics) and the chosen arm. The objective is to fit this function during the test. Concretely, assumptions are made about the type of function, such as linearity. Methods such as LIN-UCB are based on this idea.

With linear regression based bandits, the arms' parameters are often calculated by matrix inversion, which can be very time consuming, depending upon the number of item features d [22]. These approaches have shown their theoretical and practical ability to reduce cumulative regrets [23]. In particular the LIN-UCB algorithm is one of the most popular form of such methods due to its performance and interpretability. Figure 2 shows the cumulative regret over t calculated with simulated data and a linear reward function. From this figure, one can see that the LIN-UCB algorithm outperforms THOMPSON SAMPLING, RANDOM, and UCB.

Statistical techniques have been proposed for cases in which the linearity assumption is not valid. In [24], the authors propose a method based on the Generalized Linear Model (GLM), called GLM-UCB. This method allows a wider class of problems to be considered, in particular cases in which the rewards are counts or binary variables using, respectively, Poisson or logistic regression. Like LIN-UCB, GLM-UCB requires a matrix inversion and can be very costly in terms of time.

Recently, a bandit algorithm based on tree regression has been proposed, called the BanditForest algorithm [25]. This algorithm uniformly assigns visitors to each arm until a tree forest models the link function. This random forest is built from the joint distribution of contexts and rewards. Thus,

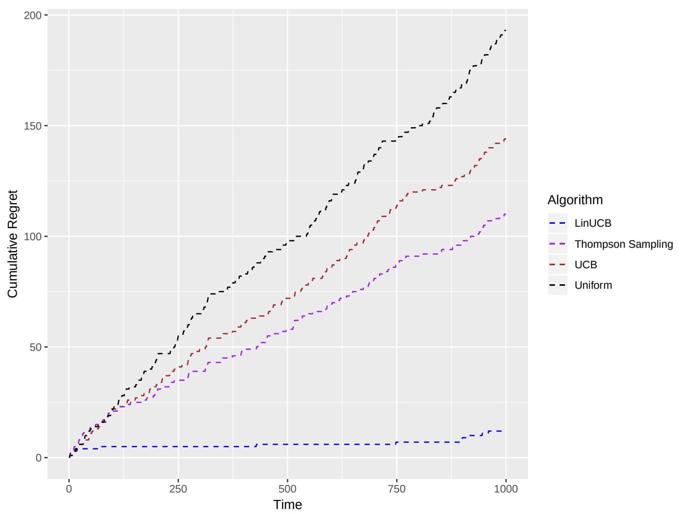


Figure 2: Cumulative regret of four linear regret-based bandit algorithms on simulated data. The lower the cumulative regret, the better the method.

all past observations (context and rewards) must be stored. The uniform assignment leads to excessive selection of sub optimal arms, causing the algorithm's performance to suffer. Moreover, the main limitation of the algorithm is that it depends on four parameters, and therefore requires strong domain expertise: two parameters directly influence the level of exploration, one controls the depth of the trees, and one determines the number of trees in the forest [26]. In [26], [27], [28] the authors propose the tree bootstrap algorithm based on a similar approach but parameterless,i.e. tree depth is automatically determined. However, these algorithms only consider binary rewards, which strongly limits their use.

The kernelised stochastic contextual bandit Kernel-Ucb [7] uses reproducing kernel Hilbert space (RKHS) to provide a non-linear model of the link reward function (like GLM-UCB) but can be slow in arriving at a decision.

In addition to making assumptions (e.g., Gaussian distribution, binary reward . . .) in order to remain understandable and implementable, the literature identifies the following drawbacks.

- In [29] the authors explain that some dynamic allocation methods do not provide more benefit than frequentist allocation when the assumptions (e.g., linear dependence between characteristics and reward, independence between items, ...) are not valid.
- The choices made by the algorithm are not explicit (black box). While understanding choices is not always necessary in a recommendation system, it is important for A/B-Tests as the user seeks to understand why and for whom one variation is better than another.
- These methods often require large a dataset.
- The CPU and/or memory requirements can be significant, particularly when the difference between versions is small.

3.2.2 *Two step-based approaches*

In two step-based approaches, it is assumed that there exists some natural groups, each having a Gaussian reward distribution and that these groups can be determined before the A/B-Test itself. The idea is to build these groups *a priori*. When a new item is submitted to the system, it is first automatically classed into a group. Then, it is assigned to an arm considering the group the item belongs to.

In [30], the authors propose the Single-K-UCB method and show that if groups are well defined, the cumulative regret converges asymptotically early in the process and the average regret falls significantly. Indeed, intuitively, the cumulative regret is in this case bounded by the sum of the cumulative gaps between the best arm and sub-optimal arms for each group (which is higher than the gap of a non informative strategy). The authors assume that the reward distributions are clustered and the clusters are determined by some latent variables. They assume that there is a surjective function f that links each item (with a context c_t) to a group k, i.e. $f(c_t) = k$, such that the reward distribution of a group k applied to an arm a, $\nu_{a,f(c)}$, is σ-Gaussian (where σ^2 is the variance). Unfortunately, they do not specify how to identify f and how to obtain the groups. They only study the problem in a context-free setting and provide a weak performance guarantee when the reward distribution is unknown in the clusters [31].

To address this problem, we propose a new method called Ctree-Ucb, which is detailed in the next section.

4 Ctree-Ucb: a contextual approach to A/B Testing

4.1 Ctree-Ucb process

Our contribution aims to address the constraints and needs experienced by users in real-world applications. Thus, we first focus on a method that can be applied in a “real time”: for instance, in e-marketing, the delay induced by the test (i.e., the dynamic allocation) must be lower than the usual display time of a webpage. Secondly, we consider that the items submitted to the system can be very heterogeneous but can be clustered according to a given criterion. Finally, we want that the results of the exploration phase are understandable by the user, and possibly reusable. In this context, we propose an approach that consists of defining groups based on item features (i.e., characteristics), each of these groups having as homogeneous population as possible. The main idea is that in such a group, the items have similar behavior relative to the proposed version and thus a group's reward distribution can be modelled by a Gaussian distribution. Each of these groups can therefore be supported by a non-contextual bandit. The general procedure is that each time a new item is presented to the system, it is automatically assigned to a group according its own features and then, through the associated bandit, a variation is assigned to it. As the complexity of this type of bandit is low, this ensure a satisfactory response time.

In summary, the proposed method Ctree-Ucb consists of two main steps:

- an offline process for creating groups based on the available data collected from the original variation,
- multiple A/B-Tests online.

The global scheme of our framework is given by Fig. 3.

We illustrate the general idea by an example: let us suppose that the test concerns the improvement of an existing web page (A). Before starting the test, behaviours of visitors who have seen page (A) are observed. We assume that the user has collected each visitor's characteristics (e.g. browser language, number of visits to the site before arriving on this page, etc.) and if a transaction was made after seeing the page. Based on these pre-collected data, our method builds a segmentation model able to classify a visitor into a group using the collected characteristics.

Next, the user constructs variation B (by modification of the original page) and start the test. In this A/B-TEST, a bandit is associated to each group and aims to determine which variation maximises the gains to the visitors classified into its group. Then each visitor arriving on the page is submitted to the test: (1) the visitor is classified into a group by applying the model on its characteristics, (2) the bandit associated to the group dynamically affects the visitor to a variation, (3) the bandit updates its statistics about the arms.

4.2 Step 1: Offline building of groups and associated classifier

To construct the groups, we suppose that there is information describing the performance of the original version (i.e., the version in production before the test). As such, a database (referred to here in as DB_{init}: $\mathcal{L}$) contains an item's context and reward (conversion, number of clicks, etc.) produced on the original version.

Thus, if the test to be carried out concerns the same type of reward and is on a variation of the original version (referred here as the arm A), using this information to build groups can only be beneficial. A model can therefore be learned using training and validation sets extracted from DB_{init}, and used to predict the group of a new item according to its context.

Many supervised methods exist to produce such a model. For instance, decision tree-based algorithms such as C4.5 [32] or C.A.R.T. [33] have shown their effectiveness in finding such homogeneous groups according a numeric/binary rewards using an entropy measurement (C4.5) or the Gini index (C.A.R.T.). Unfortunately, they present two fundamental issues: overfitting and selection bias towards continuous features [34]. Conditional inference tree based approaches have shown their robustness in comparison to these previous algorithms [35], and have a high level of stability and robustness [36], [37].

In the first step of our method, performed offline, a conditional inference tree algorithm (for instance, the one described in [38], [39], [40]) called CTREE (Algo. 2) is applied to a training dataset (herein referred to as $\mathcal{L}_n$) of n items to identify homogeneous groups. It consists of initially creating one group containing all the items. This group is associated to the root node of the tree. Then, an recursive divisive process is applied from this node. An independence hypothesis H_0 between each $j \in \{1, \ldots, d\}$ feature and the reward distribution is evaluated for all the items of the associated group, then:

- If the hypothesis can be rejected, the group is split into two subgroups using the feature that has the highest correlation with the reward, j^*, according to the value of this feature that maximises the difference between each group's distribution. The algorithm is recursively applied to the two new nodes associated to the two subgroups.
- If the hypothesis H_0 cannot be rejected at the predetermined risk level ϵ, for any feature, the recursion stops.

To verify the correlation hypothesis, statistical tests exist in the literature (for example we can cite the Bravais-Pearson test, Spearman test, Chi-squared test,... [38], [39]). At the initialisation of the CTREE-UCB scheme, such correlation tests must be defined according to the feature types (continuous, binary, categorical, ...) and reward type (continuous or binary) [41].

The conditional inference tree does not require a pruning process, which avoids overfitting. Moreover, the selection of the value upon which to split is based on the univariate p-values, thus avoiding a variable selection bias towards characteristics with many possible split values. If a statistically significant observation could have risen by "chance", because of the size of the parameter space to be searched, Bonferroni correction can be applied [42]. However, tests integrating categorical features can require very long computation time when Bonferroni correction is applied [43] nevertheless, Bonferroni correction by the Monte-Carlo method [44] can be used to reduce this time. Such a correction includes a random part, which varies the tree structure.

At the end of Step 1, groups are described by a reward average and defined by one or more features. Using this information, a predictive function f is defined which links each new item to a group. This function predicts a group k defined by an expected reward according to A. Thus, this function f can also be considered as a non-linear regression function.

Algorithm 2 CTREE algorithm

```
Require:  • $\epsilon \in ]0,1[$
          • A dataset of features $Y$ and response $X$.
          • An influence function $h$ depending on the scale of $X$.
          • An appropriate function $g_j$, which depends on the scale of the feature $Y_j$
 1: Calculate the the test statistics $s_{j0}$ for the observed data
 2: Permute the observation in the node
 3: Calculate $s$ for all permutations
 4: Calculate the $p$-values (number of test statistics $s$, where $|s| > |s_0|$)
 5: Correct $p$-values for multiple testing
 6: if $H_0$ not rejected ($p$-value $> \epsilon$ for all $Y_j$) then return
 7: else
 8:     Select feature $Y_j^*$ with the strongest association (smallest p-value)
 9:     Search for the best split of $Y_j^*$ (maximize test statistic $s$) and partition data
10:     Apply CTREE to both of the new partitions
Output: A hierarchical partitioning
```

This regression is based on the method described in [40] using the test statistic $\mathbf{T}$ which is derived from [38]. The appendix gives more details about this method. This function is used during step 2 of the A/B-TEST. As step 1 is performed offline, it does not increase the computational time of CTREE-UCB. Figure 4 shows an example of an obtained regression tree.

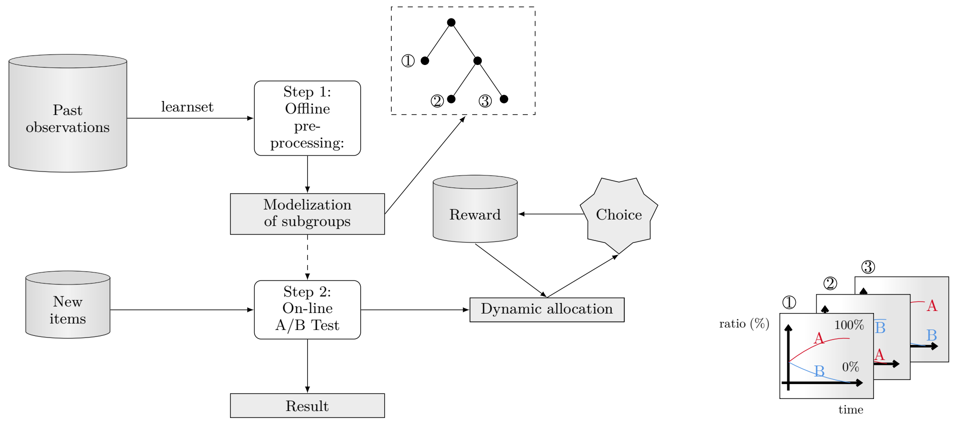


Figure 3: Ctree-Ucb: a contextual approach to A/B testing.

4.3 Step 2: Online A/B-Test

The online step corresponds to the A/B-Test. It consists of classifying each visitor into a group. The dynamic allocation is then performed by the bandit associated to the group.

As the bandits are independent, each of them can stop the exploration phase at any time and switch to the exploitation mode. The test can then end either when all the bandits are in exploitation mode, after a given number of items, or a predefined duration. Algorithm 3 defines the Ctree-Ucb method.

The reader can find more details of the theoretical guarantees of Ctree-Ucb in the appendix. The computational complexity of using CTREE to predict an average reward depends on the depth of the tree, and the depth of the tree is proportional to the (base 2) logarithm of the number of leaves [39]. The logarithm of a number grows slowly as that number gets larger, therefore even trees with a very large number of leaves will not be very deep. That makes CTREE very fast in terms of use and computation.

Algorithm 3 Ctree-Ucb algorithm

```
Require: $\alpha > 0, DB_{init}, \epsilon \in [0,1]$
 1: Generate a conditional inference tree using $DB_{init}$ and $f$
    with an accepted error $\epsilon$ using CTREE.
 2: loop
 3:     $c_t \leftarrow$ a new item with a vector $Y_t$ of features
 4:     Assign $c_t$ to group $k$ by $f(c_t) = k$
 5:     if $T_{a,k}(t) = 0$ then
 6:         $A_{t,k} = a$
 7:     else
 8:         $A_{t,k} = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,k,t} + \alpha\sqrt{\frac{2*log(\sum_{a\in\mathcal{A}} T_{a,k}(t))}{T_{a,k}(t)}}\}$
 9: Assign arm $A_{t,y}$ to $c_t$
10: $X_{c_t,A_t,k} \leftarrow$ the arm $A_{t,k}$ reward
11: Update $\hat{\mu}_{A_t,k}$ and $T_{A_t,k}(t)$
Output: A sequence of arm choices and rewards for each group $k$
```

4.4 Example on simulated data

Observations made on the data collected via A/B-Testsindicate that some of the functions linking the rewards and the feature can be modelled by a piece-wise continuous function. For example, the link between the price of a product and the quantity purchased. If the site offers one item for every 3 items purchased, linear modelling between the feature (quantity of item) and the reward no longer holds. In the medical field, if a treatment is effective for young children and elderly people, but not for adults, linear modelling also does not work. However, by using a pairwise function, such cases can be represented. In such a function, the link is linear only over an interval of values taken by the feature. When the link function between a feature and a reward is linear or piece-wise continuous, the above-mentioned traditional bandit methods have an increasing cumulative regret. To validate our method, we first propose to simulate data from a pairwise function (2000 features x and 2000 rewards) and compare the results between Ctree-Ucb, Lin-Ucb, Ucb and random. We report an example of a simulation to test the performance of Ctree-Ucb under real assumptions. For each variation (A or B), 10000 rewards are generated by the following function, related to a feature X:

$$\theta_A = (2, -1, 1.5, 0),$$
$$\theta_B = (1.5, -0.5, 1.25, 0),$$

$$X = \begin{cases} X_A = \theta_A[1], X_B = \theta_B[1], & \text{if } 1 \leq x_1 < 2 \\ X_A = \theta_A[2], X_B = \theta_B[2], & \text{if } 3 \leq x_1 < 4 \\ X_A = \theta_A[3], X_B = \theta_B[3], & \text{if } x_1 \geq 4 \\ X_A = \theta_A[4], X_B = \theta_B[4], & \text{if } x_1 < 1. \end{cases}$$

4.4.1 *Offline step*

We use 30% of the data DB_{init} for training (so the past rewards of A are the only ones observed), and the remaining 70% are used for the test. The regression tree correctly identifies groups in which the link between the feature and reward is identical (each final leaf is a group, represented by an estimated average, see Figure 4).

4.4.2 A/B-Test *(Dynamic allocation)*

During the A/B-Test itself, a dynamic allocation is made to each group. Figure 5 shows the cumulative regret over time of different algorithms. Figure 6a to 6e show the cumulative regret of Ctree-Ucb specific to each group.

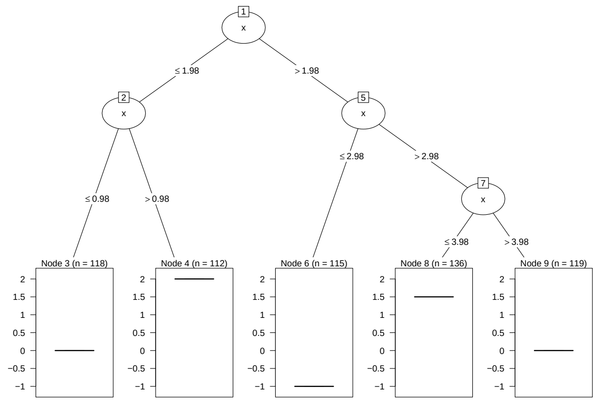


Figure 4: Groups identification on simulated data: 5 groups are identified `Node 3`, `Node 4`, `Node 6`, `Node 8`, and `Node 9`.

The following section presents the same comparison but on real datasets.

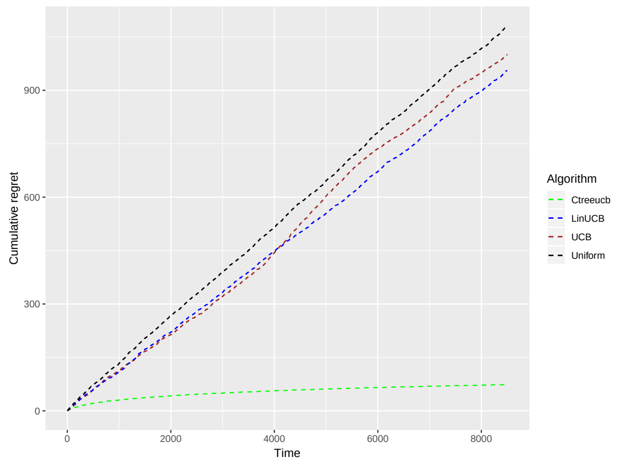
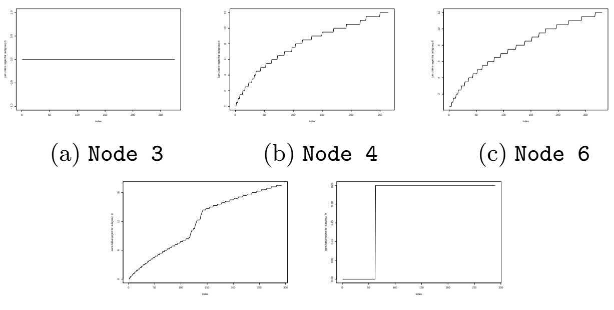


Figure 5: Cumulative regret of Ctree-Ucb, Lin-Ucb, random, and Ucb with a non linear reward function.



(a) `Node 3` (b) `Node 4` (c) `Node 6` (d) `Node 8` (e) `Node 9`

Figure 6: Cumulative regret of Ctree-Ucb for each group.

5 Materials and experimental setting

To evaluate the performance of Ctree-Ucb, we compare it to existing A/B methods: a global-based bandit (Ucb), the Lin-Ucb and Kernel-Ucb bandits, as well as a random-based algorithm that chooses variations alternatively. The main criteria for this evaluation are the cumulative and average regret. The experiments are carried out over three data sets: the first a public data set, the others are from AB Tasty and correspond to e-merchant A/B-Tests. All experiments are carried out using the R programming language on a Intel® Core™ i5-8250U CPU with 8 threads running at 1.60 GHz with 7.5 GB of RAM on 64-bit Ubuntu 17.10. All materials (including data, the conditional tree regression framework `CTREE` [41] and Ctree-Ucb) are available from: https://github.com/manuclaeys/bandit4abtest.

5.1 Data

5.1.1 *Small MovieLens dataset*

This dataset comes from the IMDB public database[3]. It contains movies described by 14 binary characteristics (Adventure, Action, Comedy, Drama, Thriller, Romance, Sci-Fi . . .) and their associated ratings (from 0 to 500) given by film reviewers. To simulate an A/B-Test using this data, we define:

- Movies as items: There are 9125 movies in the original database
- Film reviewers as the variations: Denoted by A, B, $\ldots E$ corresponding to 5 reviewers.
- Ratings as the rewards: The reward associated to a movie c_t is the rating $R_r(c_t)$ given by reviewer r associated to the variation. In case the film has not been rated by this reviewer (which may appear with recent films), the missing value is evaluated by the average of all other reviews. Therefore, $X_{.,t} = R_r(t)$ if $R_r(t)$ exists in the dataset.

The objective is to obtain the best cumulative film evaluation.

5.1.2 *AB Tasty dataset*

These A/B-Tests are comparisons of a variation of a webpage with its original state (referred here as P_1 and P_2, respectively).

These tests were performed in 2018 on AB Tasty's servers for several e-merchant clients. They consisted of using a static allocation with an equal distribution of visitors between the two pages P_1 and P_2.

For each test, visitors who navigate on the tested web page (*i.e.*, a potential customer) are identified by an ID: the first time they visit the web page, a new visitor description (see Table 1) and its associated ID are generated.

This description may vary depending on the data available to the user. Then a variation (A or B) is allocated to it. A cookie memorises the description, the ID, and the allocated version.

Each time they return to the tested page, the same variation is shown so we assume that no statistical association between visitors exist. We present the results of two A/B-Tests performed on two different websites.

All visitor actions during their visits are stored. At the end of the test (i.e., after T visitors) the reward is computed.

3. These datasets change over time, the last update was made in October 2016 but the former version used in our experiments is available from: https://github.com/manuclaeys/bandit4abtest

Table 1: Item features of the A/B-Test dataset.

Type	Features (number of possible values or domains)
Integer	Visits ($\mathbb{N}$)
Categorial	Navigator's language (27), Navigator type (6), Device (3), Operating System (7)

According to the user's objective, the reward corresponds to a visitor's purchase value during all of their visits regardless of the visit(s) the purchase(s) happened in after the affectation. It is defined by cumulative sum if the visitor t has purchased on the web page.

Notations used throughout the paper: For N variations $V_0, V_1, \ldots, V_N$, S_i is the set of items to which variation V_i has been allocated, $T_i = |S_i|$ and $T = T_0 + T_1 + \cdots + T_N$ where $|.|$ denotes the cardinality of a set.

5.2 Comparison methods

The performance of Ctree-Ucb is compared to the following four algorithms.

- Two algorithms with a non informative strategy, which do not take into account the item's context:
  - random which is parameter free.
  - The Ucb strategy described in Section 3.1.1.
- Two algorithms with a contextual strategy (see Section 3.2.1):
  - The Lin-Ucb algorithm using a linear reliability assumption.
  - The Kernel-Ucb algorithm without a linear reliability assumption, this policy estimates each variation's reward, in addition to a kernel regression of characteristics.

Each algorithm will provide a sequence of choices. These sequences are compared to a model that always chose the best variation (see Section 2.1), trained with all the data in the test set, so the cumulative and average regret at the end of the A/B-Test (iteration T) is evaluated.

Note that Lin-Ucb and Kernel-Ucb require the transformation of categorical characteristics into binary values.

5.3 Experimental protocol

5.3.1 *The* A/B-Test *parameters*

All the algorithms (except random) are derived from the standard Ucb algorithm, which requires setting the confidence interval parameter α (see Section 3.1.1). To evaluate the impact of this parameter on the results, we have carried out experiments with different values of α (from 0.25 to 2.5, as generally found in the literature).

To limit the CPU time consumed by the Kernel-Ucb algorithm, we have limited the numbers of items used in the kernel regression to 100.

5.3.2 A/B-Test *simulation*

The principle of simulation is to apply each algorithm on data sets and to compare their obtained cumulative regrets. We compare the results with a non-linear regression (CTREE) model that learns from all the data. Thus when assigning an item to a variation, regret is evaluated as the difference between the maximum prediction (between all possible variations) and the prediction of the chosen one. This assessment can be seen as a difference between conditional averages and is based on the theoretical definition presented in Section 2.

For the Ctree-Ucb method, the offline step (see Section 5.3.3) performed first and consists of learning a conditional regression tree. Then each item is evaluated by this tree in order to determinate which group it belongs to. Finally, the item is submitted to the classical Ucb algorithm associated to this group.

Since a tree is built using data from the original variation, the choice of variation A affects the rest of the process. As such, we have considered each variation as a potential original variation for each data set. So, for the MovieLens dataset, five configurations have been tested, each corresponding to a different choice of movie rating as the original variation. In the same way, each page has been tested as the original web page (A).

5.3.3 Ctree-Ucb *offline step*

The offline step of Ctree-Ucb consists of defining the item groups used in the contextual A/B-Test and is crucial as it has a large impact on the A/B-Test process. To produce these groups, we used the R conditional tree regression framework CTREE [41] with ten-fold cross validation and different maximum error risk. The displayed tree represents the groups graphically by an expected average and optionally (depending on the user's choice) the distribution's boxplot.

To assess this impact, we have carried out experiments with different configurations of the ratio between the number of items used to learn the regression tree and those used to simulate the A/B-Test.

Additional notations: $L = |\mathcal{L}|$, where $\mathcal{L}$ is the set used to learn the conditional regression tree. $T_{A/B} = |S_{A/B}|$, where $S_{A/B}$ is the set used to simulate the A/B-Test. ϵ is the error risk parameter to CTREE.

To respect the assumption that prior to the A/B-Testthe user only knows the rewards obtained by the original variation (referred here as V_A), the regression tree can only build from the set of items S_A to which variation V_A has been allocated, therefore $\mathcal{L} \subset S_A$. Two configuration have been tested:

- $\text{Conf}_{30,70}$: $\mathcal{L} = 30\%$ of V_0, $S_{A/B} = \sum_i \{70\% \text{ of } S_i\}$,
- $\text{Conf}_{100,100}$: $\mathcal{L} = V_0$, $S_{A/B} = \sum_i S_i$.

5.3.4 *Experimental configurations*

Table 2 summarises the parameters and their different potential values.

Data set	Algorithms	Parameters
MovieLens dataset	Ucb, Lin-Ucb, Kernel-Ucb	$\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$
AB Tasty dataset 1	Ctree-Ucb	$V_A \in \{V_i\}$
AB Tasty dataset 2		Config. $\in \{\text{Conf}_{30,70}, \text{Conf}_{100,100}\}$
		$\epsilon \in \{0.01, 0.05, 0.1\}$
		$\alpha \in \{0, 0.25, 0.5, 1, 1.5, 2, 2.5\}$

Table 2: Algorithm parameters.

There are 441 combinations: $3 \times 3 \times 7$ combinations for the 3 UCB-based algorithms and $(2 \times 2 \times 2 \times 3 \times 7) + (5 \times 2 \times 3 \times 7)$ for the Ctree-Ucb method. For the sake of clarity, we report only 88 combinations in our experiments.

For each experiment, a table is presented that summarises the cumulative and average regret of each algorithm. The average regret allows us to evaluate the evolution of the cumulative regret compared to the set of tested items (S). **The best performances (cumulative regret / average) appear in bold** in each table.

		CTREE-UCB					LIN-UCB	KERNEL-UCB	UCB	RANDOM
		$V_A : \mathrm{R}_1$	$V_A : \mathrm{R}_2$	$V_A : \mathrm{R}_3$	$V_A : \mathrm{R}_4$	$V_A : \mathrm{R}_5$				
Configuration		$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$	$\epsilon = 0.05$				
Config.$_{30,70}$	R_T	**19155**	21628	28458	27894	19976	22378	21239	20650	22808
	$\mathbb{E}[R_T]$	**3**	3.39	4.47	4.37	3.13	3.50	3.32	3.23	3.57
Config.$_{100,100}$	R_T	**27875**	34152	35679	35908	37679	68101	40848	31146	45643
	$\mathbb{E}[R_T]$	**3.05**	3.74	3.91	3.94	4.13	7.46	4.48	4.41	5

Table 3: Influence of segmentation parameters on cumulative regret, R_T, and average regret $\mathbb{E}[R_T]$ with the MovieLens dataset ($\alpha = 1$).

6 Experiments

6.1 MovieLens dataset

6.1.1 *Offline step*

In Figure 7, each leaf of the tree associates an average to an identified group. The leaves of the tree represent the groups identified by CTREE that will then be used in the classification model in the dynamic allocation step.

From the rewards given by reviewer1 before the test, 9 leaves were generated, which corresponds to 9 groups of items with statistically different distributions of rewards. `Node#15` is the most represented with 1133 items. The group that maximises rewards for variation A is group `Node#11`. We also note, for example, that reviewer 1 generally gives a higher rating if the film is in the "Film noir" category (`Node#11`). The lowest average reward is given to "Comedy" (`Node#8`) or "Action" (`Node#7`) movies.

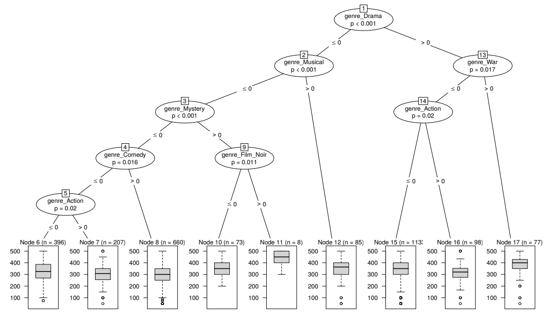


Figure 7: Conditional inference tree on MovieLens dataset (Config.$_{30,70}, V_A$: Reviewer $1, \epsilon = 0.05$).

6.1.2 A/B-TEST *(Dynamic allocation)*

In the MovieLens dataset, many films have not been seen by some reviewers and therefore, replacing missing scores (films that have not been seen by a reviewer) results in identical scores between several reviewers. There are therefore films for which the simple regret will be equal to zero regardless of the reviewer chosen. The difference between the averages Δ_a, $\forall a \neq a^*$, being very small, finding the reviewer who maximises rewards (in accordance with a context) is difficult (see Section 3.1.2). Such a situation can reduce the performance of the tested bandit algorithms and be comparable to a static allocation strategy (performed by the RANDOM algorithm). The parameter ϵ has little impact on the results in terms of regret and its impact on the results will be covered in the next experiments.For readability of Table 3, only the classic value of the accepted error risk are reported, i.e. $\epsilon = 0.05$.

CTREE-UCB has the lowest cumulative regret (in bold in the Table 3). The LIN-UCB algorithm has weak performance, one explanation could be that the linearity assumption required by this algorithm is not valid. KERNEL-UCB has a cumulative regret comparable to a static allocation (RANDOM). More data is probably needed to complete its regression. CTREE-UCB gives better results with reviewer 1 (in bold) with Config.$_{30,70}$ or Config.$_{100,100}$. Learning on reviewer 3 (on Config.$_{30.70}$) or 5 (Config.$_{100.100}$) decreases its performance. We assume that this implies an over- or under-learning depending on the configuration.

To get good results with CTREE-UCB, whatever the V_A parameter, the offline step must be performed on a population representative of the one to be tested.

Figure 8 shows the cumulative regret of each algorithm for a given configuration. The lowest regret during the test is that of CTREE-UCB (in green). The strategy UCB (in brown) comes in second position in terms of performance. The highest cumulative regret is that of RANDOM (in black). However, these results indicate a linear regret for all algorithms.

Figure 9 shows how α affects cumulative regret. In this experiment, with the exception of KERNEL-UCB, the value of α has little influence on the cumulative regret of the studied algorithms. Whatever the value α, CTREE-UCB produces the best results and guarantees their stability. Unlike other algorithms, KERNEL-UCB works differently depending on α. Its cumulative regret seems highly dependent on this parameter. Choosing a sub-optimal value for α can therefore make it less effective than RANDOM (Fig. 9).

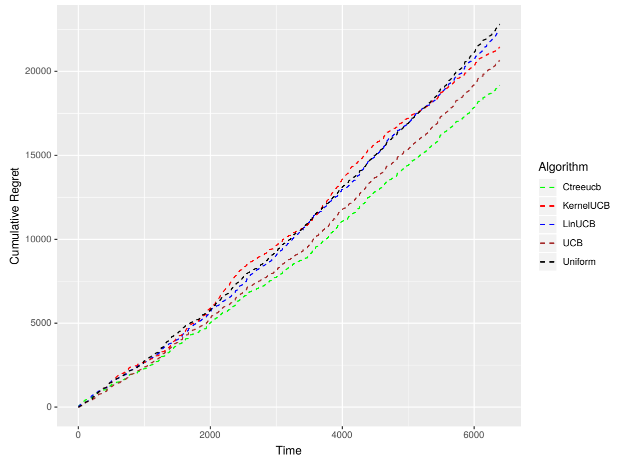


Figure 8: Cumulative regret for the MovieLens dataset (V_A : R_1, Conf$_{30,70}$, $\epsilon = 0.05$,$\alpha = 0.25$).

6.2 AB tasty database 1

6.2.1 *Offline step*

The user has a clothing sales website and its objective is to increase the value of a purchase. On Fig. 10, each leaf is associated to an expected reward (a purchase value). On Config.$_{30,70}$ 2543 visitors are dedicated to Step 1 (learning step) and 5934 visitors are tested (Step 2). On Config.$_{100,100}$ steps 1 and 2 have 8477 visitors. There are 10 groups identified by the first step. The number of past visits before seeing the tested page has the strongest correlation with the purchase values. However, purchase value can increase or decrease according to the visitor's user agent or language.

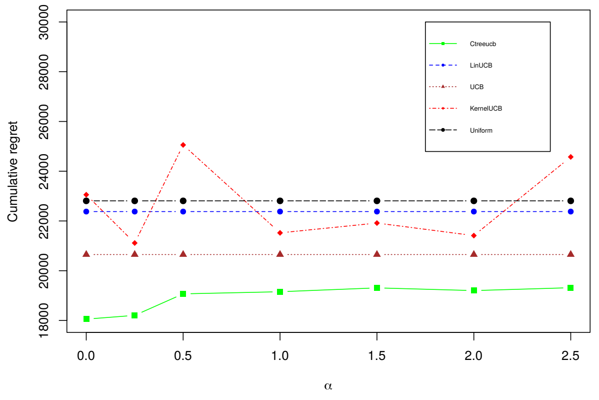


Figure 9: Cumulative regret according to α for the MovieLens dataset.

6.2.2 A/B-Test *(Dynamic allocation)*

On Config.$_{30,70}$, all results provided by Ctree-Ucb (in bold) are the best. Table 4 gives the cumulative regret according to the different parameters (ϵ and V_A) for the AB Tasty dataset 1. With all configurations the ϵ parameter doesn't modify the tree structure. However, the challenge in step 1 is to avoid overfitting (to many groups, as in Config.$_{100,100}, V\_A : P\_1$ in Table 4) or underfitting (to fewer groups, as in Config.$_{30,70}, V\_A : P\_2$ in Table 4). In fact, too few groups leads to a performance similar to that of a non-contextual strategy. On the other hand, too many groups slow down the exploration period, but Ctree-Ucb's results remain more effective than those of Ucb, Kernel-Ucb, and random.

In this test, variation A was the best of all, therefore all bandit algorithms have a logarithmic cumulative regret (see Fig. 11). It also appears that the exploration period could be stopped earlier (after 2000 visitors) with our method when compared to a frequentist approach (random) and consequently save the user money.

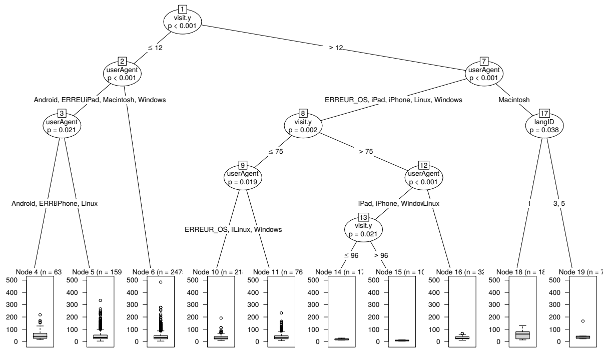


Figure 10: Tree from AB Tasty dataset 1 (Config.$_{30,70}, V\_A : P\_1, \epsilon = 0.05$).

6.3 AB Tasty database 2

6.3.1 *Offline step*

The user owns a media website, and wants to optimise purchase values. Eight-hundred visitors are assigned to step 1

		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	random
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
Conf$_{30,70}$	R_T	**100**	**100**	**100**	355	355	355	364	592	408	6610
	$\mathbb{E}[R_T]$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{1.68 * 10^{-}2}$	$\mathbf{1.68 * 10^{-2}}$	$\mathbf{5.59 * 10^{-}2}$	$\mathbf{5.59 * 10^{-}2}$	$\mathbf{5.59 * 10^{-}2}$	$\mathbf{6.11 * 10^{-}2}$	$\mathbf{9.97 * 10^{-}2}$	$6.87 * 10^{-}2$	1.11
Conf$_{100,100}$	R_T	152	152	152	67	67	67	24	448	241	8148
	$\mathbb{E}[R_T]$	$\mathbf{1.79 * 10^{-}2}$	$\mathbf{1.79 * 10^{-}2}$	$\mathbf{1.79 * 10^{-}2}$	$\mathbf{0.79 * 10^{-}2}$	$\mathbf{0.79 * 10^{-}2}$	$\mathbf{0.79 * 10^{-}2}$	$\mathbf{0.28 * 10^{-}2}$	$\mathbf{5.28 * 10^{-}2}$	$\mathbf{2.840.79 * 10^{-}2}$	0.96

Table 4: Influence of segmentation's parameters on the cumulative and average regret with AB Tasty dataset 1 ($\alpha = 1$).

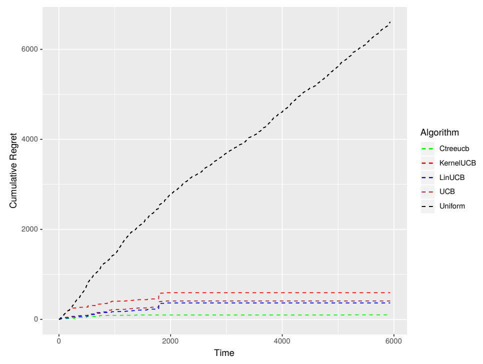


Figure 11: Cumulative regret according to α with AB Tasty dataset 1 (Conf$_{30,70}$ $V_A = P_1$, $\epsilon = 0.05$).

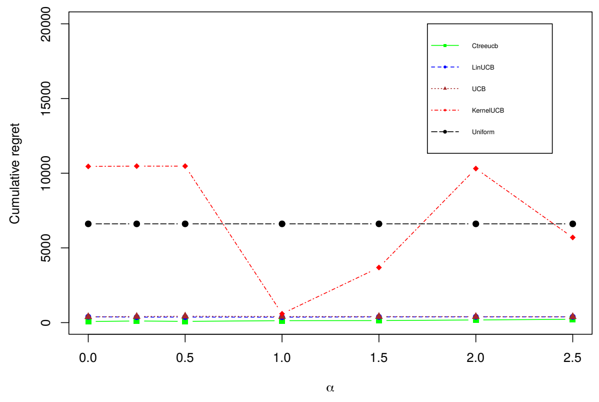


Figure 12: Cumulative regret according to α with AB Tasty dataset 1 ($V_A : P_1$, Conf$_{30,70}$, $\epsilon = 0.05$).

(learning step) and 1865 visitors are tested (step 2). As in the previous experiment, each leaf of Fig. 13 is associated to an expected reward (a purchase value). There are 7 groups discovered in the first step. We note that the characteristics present in the previous experiment (user agent, visits) also have an influence on groups in this experiment. In addition to these, the browser (called `name`) becomes relevant in this experiment. Our hypothesis is that since the sale concerns online videos, the user's browser is likely to influence the visual rendering.

6.3.2 A/B-Test *(Dynamic allocation)*

Table 5 gives the cumulative regret according to the different parameters (ϵ and V_A). On $V_A : P_1$, Conf$_{30,70}$ the tree structure is only slightly modified (occurrence or avoidance of a maximum of one/two groups). On Conf$_{100,100}$ with all configurations the ϵ parameter does not modify the tree structure, see Figure 15. However, according to $V_A = P_1$ Ctree-Ucb gives the best results whatever the configuration. Nevertheless, on $V_A = P_2$, the best results are given by Lin-

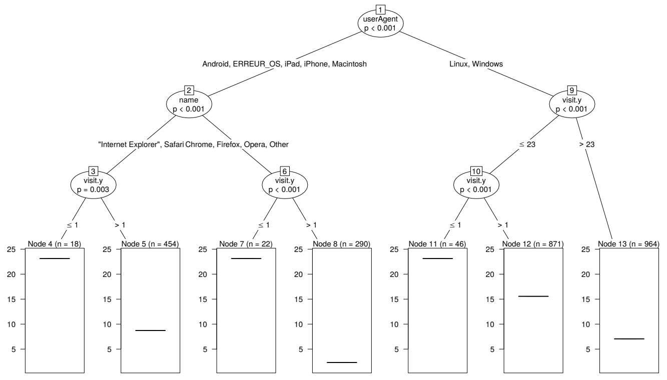


Figure 13: Tree from the AB Tasty dataset 2 (Config.$_{100,100}$, $V_A : P_1, \epsilon = 0.05$).

Ucb. Figure 14 presents the cumulative regret according to α and shows the robust performance of Ctree-Ucb.

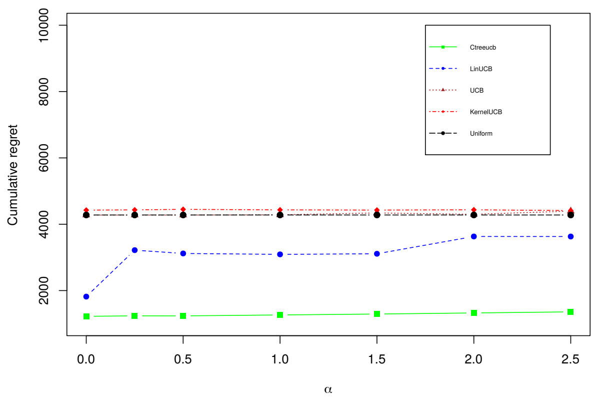


Figure 14: Cumulative regret according to α with AB Tasty dataset 2 ($V_A : P_1$, Conf$_{30,70}$, $\epsilon = 0.05$).

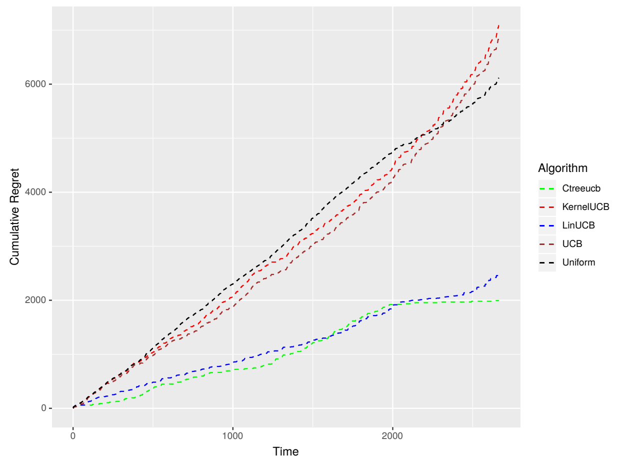


Figure 15: Cumulative regret with A/B Tasty dataset 2 (Conf$_{100,100}$ $V_A = P_1$,$\epsilon = 0.05$,$\alpha = 1$).

Group analysis:

		Ctree-Ucb						Lin-Ucb	Kernel-Ucb	Ucb	random
		$V_A : P_1$			$V_A : P_2$						
Configuration		$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$	$\epsilon = 0.01$	$\epsilon = 0.05$	$\epsilon = 0.1$				
Conf$_{30,70}$	R_T	**1251**	1263	1265	3110	3110	3110	3092	4469	4284	4279
	$\mathbb{E}[R_T]$	**0.67**	0.67	0.67	1.67	1.67	1.67	1.67	2.40	2.30	2.30
Conf$_{100,100}$	R_T	**1994**	**1994**	**1994**	3843	3843	3843	2457	7007	6114	6862
	$\mathbb{E}[R_T]$	**0.75**	**0.75**	**0.75**	1.44	1.44	1.44	0.92	2.63	2.30	2.57

Table 5: Influence of segmentation parameters on the cumulative and average regret with AB tasty dataset 2 ($\alpha = 1$).

The cumulative regret of Ctree-Ucb (see Fig. 15) is the sum of the cumulative regret of each group, we propose a more detailed analysis of Ctree-Ucb's results by observing the cumulative regret of the 7 subgroups. The parameters of Ctree-Ucb are: Conf$_{100,100}$ $V_A = P_1$, $\epsilon = 0.05$, $\alpha = 1$.

Note that the group names in Fig. 16 refer to the id of the leaf in the tree and not the n-th group.

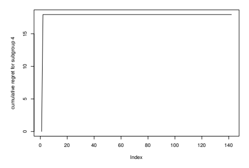
- Figures 16b, 16c, and 16e show the cumulative regret of Group #5, Group #7, and Group #11. In these figures, the cumulative regret converges asymptotically. For example in Group #7, the first half of the visitors tested produced 100% of the total cumulative regret. For the last visitors their regret is always equal to zero. This result shows that for these groups, Ctree-Ucb ends the exploitation in an optimal way. This also suggests that these groups are homogeneous (see Section 3.1.2).
- Figures 16a, 16f, and 16g show each groups cumulative regret, which is almost equal throughout the A/B-Test. Only a few visitors belonging to this group were impacted by the A/B-Test. These results show that Ctree-Ucb separates unaffected visitors correctly, see Section 5.1.2).
- Figure 16d shows a case in which the cumulative regret grows almost linearly throughout the A/B-Test. For this group, the variation chosen for exploration requires more items or is not the best for all visitors. There are different reasons that may explain this: the gap between the variation's average is very small, learning from the original page did not correctly identify all possible groups existing in the test dataset, or the characteristics used are not sufficient to give a reliable average for this group.

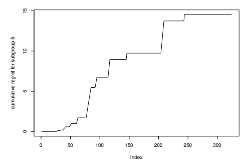
Time response:

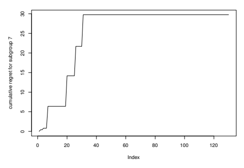
In a lot of cases, applications require allocations to be made within a very short time frame. For example, the choice between two versions of a web page must be made in real time: for each visitor, the algorithm must choose the variation in less than half a millisecond to avoid delays in displaying the page.

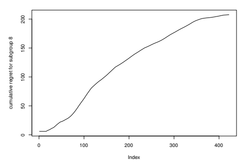
Step one of Ctree-Ucb is computed before the A/B-Test itself (offline). Thus, time allocated to step one is not considered. Only the computation time required for dynamic allocation, i.e. step 2 (online), is considered (see Table 6). We report in Tab. 7 the computation time required for Ctree-Ucb and compare it to Lin-Ucb, Kernel-Ucb, and Ucb on Config.$_{100,100}$. We also include the longest calculation time among all groups (max group). As the visitor groups are tested separately, it is possible to reduce the total computation time.

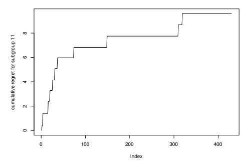
After several experiments, the response time per visitor for Ctree-Ucb is always less than one millisecond. Kernel-Ucb, on the other hand, requires the longest computation time. This is mainly due to the regression calculation of the kernel. However, Lin-Ucb and Kernel-Ucb response time

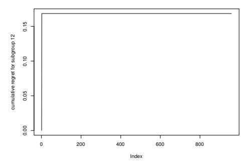

(a) Group #4


(b) Group #5


(c) Group #7


(d) Group #8


(e) Group #11


(f) Group #12

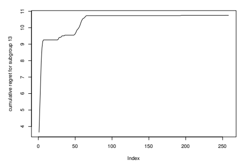

(g) Group #13

Figure 16: Cumulative regret for 7 groups with AB tasty Database 2 ($\text{Conf}_{100,100}$ $V_A = P_1$, $\epsilon = 0.05$, $\alpha = 1$).

Dataset	Nb of features	Size	Step 1 (s)	Step 2 (s)
MovieLens	19	18250	0.297	0.347
AB Tasty 1	5	8477	0.117	0.523
AB Tasty 2	4	2265	0.36	0.504

Table 6: Dataset and computation time of Ctree-Ucb in seconds.

increases with the number of features d. Ctree-Ucb has a short execution time and respects the response time required.

Experiment on one million simulated data was carried out. The behavior of Ctree-Ucb algorithm, especially on the system response time was very similar to one with the ABTatsy dataset[4].

	Ctree-Ucb	Lin-Ucb	Kernel-Ucb	Ucb
Total time execution (second)	0.504 (max group : 0.226)	0.622	16.013	0.096
Time execution by visitor (millisecond)	0.18 (max group: 0.08)	0.23	6	0.03

Table 7: Total time calculation of each algorithm ($\text{Config.}_{100,100}$) on AB Tasty dataset 2.

7 Discussion

Our experiments indicate that Ctree-Ucb responds to different A/B-Test issues. Our results include continuous or categorical characteristics, continuous reward, and a number of possible variations higher than two (A/B/C/...). It shows the performance of Ctree-Ucb for different types of tests.

For confidentiality reasons, only the datasets provided by websites that agreed to publish their data were presented. To improve performance if necessary, each group can be processed independently on a server to accelerate computation time (nevertheless it was not necessary given the good results in calculation time for AB Tasty data sets). However, Ctree-Ucb obtained good results on websites that could receive more than 2000 visitors per second.

Ctree-Ucb has three parameters: Conf., V_A, ϵ, and α. From our experiments we can conclude the following.

- A partial dataset ($\text{Conf}_{30,70}$) for step one is sufficient to obtain results comparable to the total dataset ($\text{Conf}_{100,100}$).
- By considering different original variations V_A, the results of Ctree-Ucb may be different. However, Ctree-Ucb remain good compared to results of Lin-Ucb, Kernel-Ucb, and Ucb.
- The parameter ϵ (associated with the accepted risk in the inference tree) has a low influence on the results, therefore the default value of 0.05 can be used.
- an incorrect α value can lead to a degradation of Ucb performance while Ctree-Ucb is less sensitive to the alpha parameter.

Ctree-Ucb has the following advantages.

- The model can handle both numerical and categorical values. Other techniques are often usable only with specific variable types.
- The construction of groups by a conditional inference tree simplifies their interpretation. Using Boolean logic, the user understands the characteristics that have the most impact on a group's distribution, unlike black box models such as neural networks, whose results are difficult to explain.
- Group construction, performed offline, results in a response time comparable to a non-contextual method and can be decreased with distributed computing (like one group per server). Thus, when the user wants to have a choice quickly, Ctree-Ucb can be used.

However, Ctree-Ucb requires:

- an original variation, set up before the test;
- stationary reward distributions, as Lin-Ucb, Kernel-Ucb, and Ucb;
- the population of items used to provide the groups, before starting the test, to be representative of the population of items tested.

The quality of the result obtained from Ucb and Lin-Ucb are very different according to the data type. These algorithms can be equivalent to random when their assumptions (like linearity, ...) are not verified, or when α's value is not optimal.

Kernel-Ucb is difficult to use in practice. As Cesa-Bianch *et al.* note "when the number of kernel evaluations is bounded, there are cases where no algorithm attains performance better than a trivial sub-sampling strategy, where most of the data is thrown away. Also, no algorithm can work

4. To limit the paper length, this experiment has not been reported in the paper but can be easily reproduced using data and codes available at: https://github.com/manuclaeys/bandit4abtest

well when the regularisation parameter is sufficiently small or the norm constraint is sufficiently large" [45].

On the other hand, with AB Tasty's datasets, Ctree-Ucb gets the best results. These datasets correspond to our main objective, the other is presented to provide results on public datasets.

8 Conclusion

In this paper, we present the new approach, Ctree-Ucb, for A/B-Test based on bandit models. It focuses on practical (real) applications. Experiments on synthetic and real data show that Ctree-Ucb achieves a cumulative regret comparable to the best performance of methods from the state of art. We also show that Ctree-Ucb provides good results whatever parameters are chosen and that a default setting is enough to obtain reliable results. Furthermore, the computation time required by Ctree-Ucb to make allocations allows its integration in an industrial environment where fast response times are crucial. Moreover, experiments have shown that the conditional inference tree is a powerful method for achieving group definition, leading to a decrease in cumulative/average regret.

Ctree-Ucb's high performance in identifying groups with different original variations suggests a correlation between the distribution of variations. In practice, in most A/B-Tests, the changes provided by a variation are limited. We can therefore create groups on variation A and assume that they are similar on variation B.

Our experiments on both partial data (first 30%) and complete data show that the groups are persistent over time. Moreover, group identification can help guide the user in the composition of the test itself. If the test was irrelevant for a group another, more specific, test can be performed.

Our future work will focus on Ctree-Ucb's ability to consider temporal data, such as a visitor's navigation timeline prior to their arrival on a test page.

Acknowledges

This research was supported by AB Tasty and ICube laboratory (University of Strasbourg/CNRS). We thank Jérémie Mary for his assistance with bandit techniques.

References

[1] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3-4, pp. 285–294, 1933.

[2] H. Robbins, "Some aspects of the sequential design of experiments," *Bull. Amer. Math. Soc.*, vol. 58, no. 5, pp. 527–535, 09 1952.

[3] A. J. C. Gittins and J. C. Gittins, "Bandit processes and dynamic allocation indices," *Journal of the Royal Statistical Society, Series B*, pp. 148–177, 1979.

[4] O. Nicol, J. Mary, and P. Preux, "Icml exploration and exploitation challenge: Keep it simple !" in *Journal of Machine Learning Research (JMLR)*, 2012, iJournal.

[5] E. Kaufmann, O. Cappé, and A. Garivier, "On the Complexity of A/B Testing," *ArXiv e-prints*, May 2014.

[6] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 661–670.

[7] M. Valko, N. Korda, R. Munos, I. Flaounas, and N. Cristianini, "Finite-time analysis of kernelised contextual bandits," in *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, ser. UAI'13. Arlington, Virginia, United States: AUAI Press, 2013, pp. 654–663.

[8] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 208–214.

[9] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Advances in Applied Mathematics*, vol. 6, no. 1, pp. 4 – 22, 1985.

[10] M. N. Katehakis and A. F. Veinott, "The multi-armed bandit problem: Decomposition and computation," *Mathematics of Operations Research*, vol. 12, no. 2, pp. 262–268, 1987. [Online]. Available: http://www.jstor.org/stable/3689689

[11] L. P. Kaelbling, M. L. Littman, and A. P. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996. [Online]. Available: http://people.csail.mit.edu/lpk/papers/rl-survey.ps

[12] S. L. Scott, "Multi-armed bandit experiments in the online service economy," *Appl. Stoch. Model. Bus. Ind.*, vol. 31, no. 1, pp. 37–45, Jan. 2015.

[13] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2, pp. 235–256, May 2002.

[14] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *IEEE Transactions on Neural Networks*, vol. 16, pp. 285–286, 1998.

[15] H. Bastani, M. Bayati, and K. Khosravi, "Mostly exploration-free algorithms for contextual bandits," 2017.

[16] M. Tokic and G. Palm, "Value-difference based exploration: Adaptive control between epsilon-greedy and softmax," in *KI 2011: Advances in Artificial Intelligence*, J. Bach and S. Edelkamp, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 335–346.

[17] M. N. Katehakis and H. Robbins, "Sequential choice from several populations," *Proceedings of the National Academy of Sciences*, vol. 92, no. 19, pp. 8584–8585, 1995.

[18] G. Burtini, J. Loeppky, and R. Lawrence, "A survey of online experiment design with the stochastic multi-armed bandit," *CoRR*, vol. abs/1510.00757, 2015.

[19] N. Carrara, E. Leurent, R. Laroche, T. Urvoy, O.-A. Maillard, and O. Pietquin, "Budgeted Reinforcement Learning in Continuous State Space," in *Conference on Neural Information Processing Systems*, ser. Advances in Neural Information Processing Systems, vol. 32, Vancouver, Canada, Dec. 2019. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02375727

[20] A. Pacchiano, M. Phan, Y. Abbasi-Yadkori, A. Rao, J. Zimmert, T. Lattimore, and C. Szepesvari, "Model selection in contextual stochastic bandit problems," 2020.

[21] L. Zhou, "A survey on contextual multi-armed bandits," *CoRR*, vol. abs/1508.03326, 2015.

[22] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge University Press, 2020.

[23] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *Foundations and Trends® in Machine Learning*, vol. 5, no. 1, pp. 1–122, 2012.

[24] S. Filippi, O. Cappe, A. Garivier, and C. Szepesvári, "Parametric bandits: The generalized linear case," in *Advances in Neural Information Processing Systems 23*, J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds. Curran Associates, Inc., 2010, pp. 586–594.

[25] R. Féraud, R. Allesiardo, T. Urvoy, and F. Clérot, "Random forest for the contextual bandit problem," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, A. Gretton and C. C. Robert, Eds., vol. 51. Cadiz, Spain: PMLR, 09–11 May 2016, pp. 93–101. [Online]. Available: http://proceedings.mlr.press/v51/feraud16.html

[26] A. N. Elmachtoub, R. McNellis, S. Oh, and M. Petrik, "A practical method for solving contextual bandit problems using decision trees," in *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney,*

Australia, August 11-15, 2017, 2017. [Online]. Available: http://auai.org/uai2017/proceedings/papers/171.pdf

[27] ——, "A practical method for solving contextual bandit problems using decision trees," *CoRR*, vol. abs/1706.04687, 2017. [Online]. Available: http://arxiv.org/abs/1706.04687

[28] D. J. Foster, A. Agarwal, M. Dudík, H. Luo, and R. E. Schapire, "Practical contextual bandits with regression oracles," in *ICML*, 2018.

[29] J. Vermorel and M. Mohri, "Multi-armed bandit algorithms and empirical evaluation," in *Machine Learning: ECML 2005*, J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 437–448.

[30] O.-A. Maillard and S. Mannor, "Latent Bandits." Jan. 2014, extended version of the paper accepted to ICML 2014 (paper and supplementary material).

[31] Y. Qi, Q. Wu, H. Wang, J. Tang, and M. Sun, "Bandit learning with implicit feedback," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 7276–7286. [Online]. Available: http://papers.nips.cc/paper/7958-bandit-learning-with-implicit-feedback.pdf

[32] J. R. Quinlan, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[33] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees*, ser. The Wadsworth and Brooks-Cole statistics-probability series. Taylor & Francis, 1984.

[34] Y.-S. Shih, "A note on split selection bias in classification trees," *Computational Statistics & Data Analysis*, vol. 45, no. 3, pp. 457–466, 2004.

[35] J. Mingers, "Expert systems-rule induction with statistical data," *The Journal of the Operational Research Society*, vol. 38, no. 1, pp. 39–47, 1987.

[36] C. Strobl, J. C. Malley, and G. Tutz, "An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests." *Psychological methods*, vol. 14 4, pp. 323–48, 2009.

[37] W.-Y. Loh, "Fifty years of classification and regression trees," *International Statistical Review*, vol. 82, no. 3, pp. 329–348.

[38] H. Strasser and C. Weber, "On the asymptotic theory of permutation statistics," 1999.

[39] T. Hothorn, K. Hornik, M. A. van de Wiel, and A. Zeileis, "A lego system for conditional inference," *The American Statistician*, vol. 60, no. 3, pp. 257–263, 2006.

[40] T. Hothorn, K. Hornik, and A. Zeileis, "Unbiased recursive partitioning: A conditional inference framework," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 651–674, 2006.

[41] T. Hothorn, U. München, K. Hornik, W. Wien, A. Zeileis, and W. Wien, "party: A laboratory for recursive partytioning."

[42] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, no. 293, pp. 52–64, 1961.

[43] T. Hothorn, K. Hornik, M. van de Wiel, and A. Zeileis, "Implementing a class of permutation tests: The coin package," *Journal of Statistical Software, Articles*, vol. 28, no. 8, pp. 1–23, 2008.

[44] T. Morikawa, A. Terao, and M. Iwasaki, "Power evaluation of various modified bonferroni procedures by a monte carlo study," *Journal of Biopharmaceutical Statistics*, vol. 6, no. 3, pp. 343–359, 1996, pMID: 8854237.

[45] N. Cesa-Bianchi, Y. Mansour, and O. Shamir, "On the complexity of learning with kernels," *CoRR*, vol. abs/1411.1158, 2014.

Emmanuelle Claeys Emmanuelle Claeys received an engineering degree from ESIEA (Paris, France) in 2014. She obtains her PhD from the University of Strasbourg in 2019. Her main areas of research interest are machine learning, data mining, and applied statistics. She is a temporary member of AB Tasty, charged with providing testing and personalisation solutions using web analytics.

Pierre Gançarski Pierre Gançarski received his Ph.D. in 1989 and his Habilitation in 2007 in Computer Science from the University of Strasbourg (France). He is currently Professor of Computer Science at the University of Strasbourg. His current research interests include collaborative multistrategical clustering with applications to complex data mining and remote sensing image analysis.

Myriam Maumy-Bertrand Myriam Maumy-Bertrand received her Ph.D. in Statistics in 2002 from the *Pierre et Marie Curie (Paris VI)* University (France). From September 2013 she was in a CNRS delegation for one year. She is currently full lecturer of Statistics at the University of Strasbourg. Her current research interests include non-parametric statistics, PLS regression, statistical learning, variable selection, regularisation and application in large dimensions.

Hubert Wassner Hubert Wassner received an engineering degree from ESIEA (Paris, France) in 1995. He was a research engineer at GENSET from 1997 to 2004 and teacher of computer science and algorithms at ESIEA from 2004 to 2014. He's currently the chief data scientist for the AB Tasty. His current research interests include Bayesian statistical tests, bandit algorithms, and predictive segmentation in web analysis.