Time Series Clustering for Enhanced Dynamic Allocation in A/B Testing

Emmanuelle Claeys, Myriam Maumy-Bertrand, Pierre Gançarski

Time Series Clustering for Enhanced Dynamic Allocation in A/B Testing

Emmanuelle Claeys[1] (✉), Myriam Maumy-Bertrand[2], and Pierre Gançarski[3]

[1] IRIT, University of Toulouse III, 31400, Toulouse France
emmanuelle.claeys@irit.fr
[2] University of Technology of Troyes, 10430 Rosières-prés-Troyes, France
[3] ICUBE, University of Strasbourg, 67411 Illkirch-Graffenstaden, France

Abstract. An A/B-Test is a method for evaluating online experiments on target items and observing which A/B/C/... variations are better through log reports and statistical analysis of the rewards earned by each variation. Recent advancements in A/B-Tests through reinforcement learning encompass dynamic allocation employing multiarmed bandits (MAB). MABs provides A/B-Tests with fast identification of the best variation (A or B) and helps limit the loss of the test i.e. the cost of exploring low-reward variation. When partial information is available before assigning variations, dynamic allocation is extended to the contextual multiarmed bandit problem (CMAB). Current state-of-the-art approaches for empirically estimating the context-dependent reward function for each variation demonstrate strong performance in limiting test loss and personalized tests. However, few studies have addressed this problem in the context of variable-sized time series. This paper presents a new reinforcement learning methodology to handle A/B-Tests with variable-sized time series as context information. We provide two new methods that obtain a minimization of the cumulative regret with a soft computational cost. This paper also provides numerical results on real A/B-Test datasets, in addition to public data, to demonstrate an improvement over traditional methods.

Keywords: A/B-Test · Mutliarmed bandit · Time series.

1 Introduction

In many domains, experimental evaluation is necessary for assessing the relevance of modifications made to an existing entity according to one or more objectives. For instance, an e-marketing team can look for the best modification of a web page design to increase sales [6]. The original variant (A) and its variations (B/C/...) are compared in parallel in a real environment. This leads to the exploration-exploitation dilemma which opposes the cost of learning the best variation (exploration phase) and the benefit obtained by using it in the future (exploitation phase). To tackle this dilemma, novel A/B-Test-*based* approaches have emerged especially for real-world problems [7, 21] that involve sequential decision-making, such as selecting a variation A/B/... The decision-making here

consists of assigning the *items* (patients, visitors, recommendations, ...) to the different variations (A/B/...) in order to evaluate the performance of each one (survival rate, average basket, click rate ...). During this *exploration*, it is assumed that the result, called *reward*, of each *assignment*, can be observed after a fixed period, to evaluate the performance of each variation. At the end of this exploration, the *user* can decide which variation should be implemented (i.e., during *production*) based on their performance. In [11], the authors highlighted conducting the A/B Testing in a sequential manner and without a random stopping rule to determine the completion of the experiment using a reinforcement learning policy (also known as a multiarmed bandit, MAB). The MAB is often formulated as the following problem: given a set of bandit "arms" (variation), each associated with a fixed but unknown reward probability distribution [12], an agent selects an arm to play at each iteration (when a visitor comes to the webpage), and receives a random reward variable (click, purchase, ...), sampled according to the selected arm's distribution, independently of the previous actions. More formally, $\mathcal{A}$ is the space of actions (finite), $\mathcal{X}$ a set of rewards, at each iteration $t \in \mathbb{N}^+$, an agent select an arm $a_t \in \mathcal{A}$ and receive a reward $r_t \in \mathcal{X}$ where ξ_t is a noise centered ($\mathbb{E}[\xi] = 0$) and further conditionally sub-Gaussian such as:

$$r_t = \underbrace{f(a_t)}_{\text{reward funct.}} + \underbrace{\xi_t}_{\text{noise}}. \tag{1}$$

Note that f is initially unknown and can be the average (stationary or not) of the chosen arm reward a_t. If the agent chooses, at iteration t, a suboptimal variation, it suffers *simple regret* equal to the difference between the reward from the optimal variation $a_\star$ and the reward from the chosen variation a_t at iteration t. The goal of the agent is to minimize the cumulative sum of regret R_t at $t = T$, i.e. the end of the A/B-Test : $R_T = T\mu_{a^\star} - \sum_{a\in\mathcal{A}} N_a(T) \times \mu_a$ where $a_\star = \text{argmax}_{a\in\mathcal{A}}\, \mu_a$ (and $\mu_{a^\star}$ the average of best arm) and $N_a(T)$ is the number of plays of an arm a at the end T of the A/B-Tests. Thus, an efficient bandit policy must have an average regret less than the average regret of a random policy when $T \to \infty$. A decrease in regret implies that the agent selects arms that maximize gains, leading to an increase in average gain (e.g., average CTR) by the end of the test. Thus, one can study both regret (to minimize) and average gain (to maximize), the choice depending on the ability to observe rewards across all variations. A particularly useful version of MAB is the contextual multiarmed bandit (CMAB) [14], where at each iteration, before choosing an arm, the agent observes a d-dimensional context feature vector $c_t \in \mathbb{R}^d$ sampled from some unknown distribution. In that case the best arm is $a_\star = \text{argmax}_{a\in\mathcal{A}}\langle\theta_a, c_t\rangle$ with $\theta \in \mathbb{R}^d$ as the parameter of an arm. Then a reward becomes : $r_t = f(a_t, c_t) + \xi_t$. The c_t context vector encapsulates essential features of an item, such as age, origin, and gender, revealed before allocation choices. However, conventional CMAB approaches encounter challenges when c_t contains time series data. This inability to use time series to describe an item makes it impossible to employ CMABs in the context of A/B-Testing related to e-commerce, where time series are commonly used to describe website visitors. However, deploying suboptimal

variations into production without knowing a stopping criterion in advance significantly hinders users from conducting A/B tests on their websites. A CMAB that utilizes time series as context would allow both the consideration of the evolving nature of website visitors and the encouragement of users to take risks to test variations. We propose in this paper a new approach by the inclusion of temporal data in c_t. Our methodology performs time series clustering based on the evolving features of visitors who have interacted with the original production page (Version A) in the past before conducting the test. This new approach not only recognizes the significance of time series but also introduces a nuanced preprocessing step, departing from traditional methodologies. By incorporating time series data and employing advanced clustering techniques, our method increases the average gain at the end of A/B-Test (or decreases the regret). In addition, our innovative approach improves A/B testing practices by facilitating a deeper understanding of consumers behavior. Through the identification of distinct patterns in marketing, our methodology insights into tailoring experiments to specific audience segments. This contributes to a more nuanced interpretation of test. According to the user's needs, we propose two algorithms: DBA-Ctree-Ucb and DBA-LinUCB, with significant improvements in terms of regret. Section 2 provides an overview of the state-of-the-art in CMAB methods, both with and without presegmentation. It also introduces the technique employed to address the temporal aspect of CMABs, and section 3 presents our two novel algorithms. Furthermore, Section 4 presents the experimental results obtained from various datasets. Finally, Section 5 concludes with a discussion.

2 State of the art

2.1 Classical MAB problem

The goal of the agent is to use knowledge from past observations to maximize long-term rewards : $a_{t+1} = F_t(a_1, r_1, \ldots, a_t, r_t)$. To achieve this, the agent must determine and select systematically the arm with the highest average reward $a_\star = \text{argmax}_{a \in \mathcal{A}} \mu_a$ as soon as possible, thus striking a balance between exploration (testing different arms) and exploitation (selecting the arm with the highest expected reward). Rather than focusing on cumulative rewards, which have no theoretical guarantee, theoretical analyses of bandit models focus on cumulative regret [12]. Observing convergent cumulative regret implies that the agent consistently makes the correct choices systematically after a time $t \in [1; T]$. As such, cumulative regret leads to theoretical bounds that are presented for many bandit algorithms in the literature. An in-depth technical analysis of the classical MAB was given in [13], where policies assuming only one best arm regardless of the item features (c_t) asymptotically reach a regret of $\mathcal{O}(\log T)$. However, to establish a theoretical bound on regret, it is essential to compare the performance against a reference model, often referred to as an oracle, which can learn from all the available data. Among the MABs classically used is Ucb algorithm [12] which, for each arm $a \in \mathcal{A}$, constructs an adaptive upper confidence interval on the mean: $UCB(a,t) = \hat{\mu}_a(t) + \alpha_{\text{ucb}} \sqrt{\frac{2\log(t)}{N_a(t)}}$ with $\alpha_{\text{ucb}} \in \mathbb{R}_+^*$ a positive

parameter and $N_a(t)$ is the number of selections of the arm a up to round t. At t, the U.C.B algorithm chooses $\text{argmax}_{a \in \mathcal{A}} UCB(a,t)$. Other methods, such as Ucb variants, Thompson Sampling, KL-UCB or EXP3 algorithms [20], can be used as alternatives to Ucb. However, these algorithms have a regret bound that is strongly dependent on the joint distribution of the arms. The learning process takes longer when the means of the arms are close. One possible solution to maximize this difference is to leverage the partial information available before selecting an arm, which is defined as the *context*.

2.2 Contextual Multi Armed Bandit

In the CMAB problem, the agent receives partial information such as item features, referred to as context, before making a decision at each iteration. More formally, at each iteration t, the agent observes a particular item described by a d-dimensional feature vector $c_t \in \mathbb{R}^d$. The agent chooses an arm $a_t \in \mathcal{A}$ to apply to this item based on past contexts and rewards observed in previous iterations. Like in classical MAB, the agent cannot observe rewards from arms other than a_t. The context in the CMAB can be defined differently depending on how contexts are revealed and how assumptions are made about the reward function. In our case, each different context $\mathbf{c_t}$ is no longer a vector but a d time series of maximal size m, represented by a matrix $d \times m$. We distinguish the vector representation c_t from its matrix representation $\mathbf{c_t}$ which results in $r_t = f(a_t, \mathbf{c_t}) + \xi_t$ where $\mathbf{c_t} \in \mathbb{R}^{d \times m}$. This problem of context dimension was first studied in The Query-Ad-Algorithm [3]. The Query-Ad-Clustering algorithm achieves a regret of $\mathcal{O}(T^{1-\frac{1}{2+|\mathcal{C}|}+\epsilon})$, where $\mathcal{C}$ is the set of possible contexts and an ϵ positive constant. In the Query-Ad-Clustering algorithm, the reward estimates are accurate as long as the context partitions are similar to each other. However, when the context dimension is large, the regret bound becomes almost linear. This issue is addressed in [16], where the arm rewards are assumed to depend on an unknown subset of the context. It is demonstrated that the regret in this case depends only on the number of relevant groups [8] and requires a learning prestep; however, no details are given on how to achieve this group segmentation. Two approaches are possible in AB testing to define these groups: learning them during preprocessing (data collected before conducting the test) or online with contextual bandit.

Preprocessing approaches Preprocessing approaches assume the existence of natural groups, each having a Gaussian reward distribution [16], which can be determined online [15] or before the MAB allocation. The approach involves learning a set of groups $\mathcal{G}$ and a mapping group function association g before the A/B-Test. Let's $g : \mathbb{R}^d \to \{0;1\}^{|\mathcal{G}|}$ with $|\mathcal{G}| \in \mathbb{N}$ the number of possible groups. When a new visitor described by c_t is submitted to the agent, the function $g(c_t)$ classifies it into one of these groups ($g(c_t) = \widetilde{g}_t$ with $\widetilde{g}_t \in \mathcal{G}$) and then a non-contextual bandit strategy assigns it to an arm according to previous rewards. So partitioning between several groups limits regret when divergence is maximized. In [6] the authors propose the use of the Ctree-Ucb algorithm

to construct the association function g using a conditional inference tree [9] (see Alg. 1 in Appendix) before applying several UCB bandits. Their partitioning involves collecting data from the original variation A and defining groups based on past visitors' features. Each of these groups has a reward distribution that can be modeled by a Gaussian distribution. Group detection here is a type of recursive partitioning method that involves the following basic steps:

- Step 1. Select the feature predictor that best separates different values of the reward distribution with a statistical p-value. The p-value error was corrected with a Bonferroni correction (a method to counter the problem of multiple comparisons).
- Step 2. This variable is split, and the data are divided into two datasets.
- Repeat steps 1 and 2 recursively until no further splits can be made based on predefined p-value rejection (according to the α_{CTREE} risk).
- A tree-like group partitioning model g is produced (see Fig 1).

To train the inference tree, CTREE-UCB uses the data present before starting the test (denoted as $\mathcal{L}$, collected on variation A). While the set of data from variation B is not observed, the authors in [6] show that in many cases of A/B-TESTS the groups observed on A are also observed in B. During the test, all new items are matched to a predefined group (from the tree). Each group can be supported by a noncontextual bandit. When a new item is presented to the agent, it is automatically assigned to a group based on its features, and an arm is assigned to it through the associated MAB, ensuring a satisfactory response time (millisecond time scale). This allows for a lower cumulative regret than other online learning methods, an interpretability due to the inference tree, and does not slow down the user experience (< 200 milliseconds), since the group learning step is performed only once and only non-contextual MABs are used afterwards. The alternative approach to the preprocessing step is learning the context function online, which is described in the following section.

Combinatory function Another possible approach is to learn the reward context function without any information before the test. A popular framework for contextual bandits is LinUCB proposed in [14], and variants [5] which assume a linear combination between the context and the d-parameters of each arm. LINUCB estimates the expected reward of each arm a as a linear regression of the context vector c_t, where $\theta_a \in \mathbb{R}^d$ is the regression coefficient of an arm reward function to be learned. We denote by α_{LINUCB} the parameter for the importance of exploration as $\alpha_{\text{LINUCB}} = 1 + \sqrt{(\log 2/\delta_{CI})/2}$ where $1 - \delta_{CI}$ is the confidence interval. We assume that M is an invertible matrix; and denote by $M^{-1} \in \mathbb{R}^{d\times d}$ the updated weight, which can be interpreted as the covariance of the coefficient θ_a. Hence, LINUCB considers the upper confidence bound as $\hat{\theta_a}^T c_t + \alpha_{\text{LINUCB}}\sqrt{c_t^\top M^{-1} c_t}$. The arm a_t selected is the one maximizing the upper confidence bound: $a_t = \text{argmax}_{a\in\mathcal{A}}(\theta_a^\top c_t + \alpha_{\text{LINUCB}}\sqrt{c_t^\top M^{-1} c_t})$. LINUCB gives a regret in $\mathcal{O}(\sqrt{Td})$. Modified versions of this algorithm, such as SUPLINUCB with kernel functions, are studied in [13] where the regret is $\mathcal{O}(\sqrt{Td})$. When

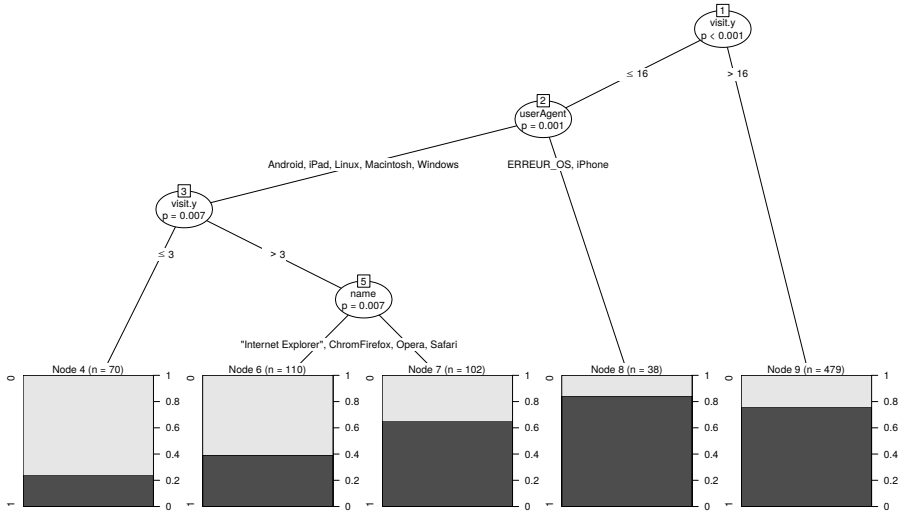

Fig. 1: Example of a conditional inference tree generated from the original page (AB Test dataset [6]) There are 5 groups with different CTR, determined by factors such as the number of past visits on the site (`visit.y`), user agent, and browser name. A separate UCB MAB will be employed for each identified group.

d is very large, inverting the matrix M can become computationally expensive and require a significant number of items. To handle nonlinear reward functions, recent works such as kernelized stochastic bandits [24] or deep neural networks [23], have considered the past selected arms and received rewards as training samples. [22] computes a gradient-based upper confidence bound with respect to the trained neural network strategy to select arms. However, when the context is a large vector, the aforementioned approaches need to work on batch style or become computationally expensive, which leads to a significant amount of regret.

2.3 Large-scale context

The algorithms detailed in the previous section have strong restrictions on the dimension of the context, where c_t must be a vector of numerical non correlated values. These restrictions limit the use of bandit algorithms when the features to be considered are time series ($\mathbf{c_t} \in \mathbb{R}^{d \times m}$). One solution to this problem is to flatten the time series of each temporal feature. One of the several problems with this approach is that if the size of feature vectors is large, it introduces significant variability in the bandit's performance, leading to a large amount of data to be collected. To address the problem of size context vectors, [4] proposed empirically building a LASSO-type statistical model and integrating it into a bandit problem. However, this approach requires that the features be independent of

each other, which is not the case when the features come from the same time series. In [18], the use of LSTM for handling time series data has the drawback of requiring a large number of visitors and frequent model retraining. This results in significant computational costs associated with deep learning methods and slows down the display of the test page, making it impractical for e-commerce applications. Considering temporal features is a difficult task without prior knowledge of the data. Additionally, we found no information in literature about A/B-TESTing contextual bandit algorithms that can adapt to data of variable size. The following section presents our contribution to address this issue.

3 Contribution

3.1 Illustrative dataset

We detail our approach using the AB tasty dataset 1 (owned by AB TASTY©). AB Tasty Dataset 1 comes from an AB testing platform that compares two versions of the same web page for an e-commerce site. The allocation of A or B to a visitor is randomized with a static allocation. Each visitor is assigned to a single variation until the end of the test (conducted over 15 days). If the visitor has made a purchase from the test page, a reward of 1 is assigned; otherwise, the value is 0. The data contains the history of visitor sessions generated before arriving on the test page: for each visitor, $\mathbf{c_t}$ includes the following information from the visitor's first visit day on the site until the day he/she arrives on the tested page.

- `presence_time_serie`: series of binary values indicating for each day whether the visitor visited the site or not.
- `connexion_time_serie`: series of hours of connections (when the visitor arrived at the site).
- `time_spend_time_serie`: series of visitor's session duration (milliseconds).

There are $n = 5156$ visitors, the shortest session is 2 days, and the longest session is 14 days.

3.2 Two new algorithms DBA-Ctree-Ucb and DBA-LinUCB for handling times series for A/B-Tests

Currently, there is no existing state-of-the-art method that effectively handles an evolving item context. However, the data presented prior to the beginning of the test could enable the transformation of this context, making it usable by a traditional context bandit method. This preprocessing step has already been employed in CTREE-UCB [6] to create an item segmentation that maximizes the difference between arm means and has demonstrated a significant decrease in cumulative regret. Our contribution proposes a model for replacing a visitor's time series with clusters and demonstrates how this improves dynamic allocation compared to transforming series into averages. We introduce two novel extensions, DBA-CTREE-UCB and DBA-LINUCB, which complement the search for

context-dependent optimal variations with preprocessing utilizing DBA-DTW-kmeans. As we said in the introduction, domains such as e-marketing, and the recognition of evolving customer profiles are crucial for effectively targeting consumers, personalizing offers, and improving the user experience in an A/B test. Time series clustering can handle complex and multidimensional data, taking into account various variables such as purchase history, online interactions, geographical data, etc. By utilizing this method, marketers obtain a holistic view of consumer profiles and identify specific subgroups within the population. The clustering parameters can be adjusted based on the study objectives, such as the desired number of clusters or sensitivity to changes. To include time series clustering in a dynamic allocation algorithm, we propose two algorithms: DBA-Ctree-Ucb and DBA-LinUCB. These algorithms improves the Ctree-Ucb and LinUcb algorithms, respectively, which do not handle temporal aspects. It is important to note that the user is expected to obtain an original variation (such as web page A) and the historical data (such as logs) of the items that have undergone this original variation. These data, collected before their arrival on the test page, form learning clusters required for the DBA-LinUCB and DBA-Ctree-Ucb algorithms. Each temporal series in the known items is replaced as a categorical feature: a cluster. The following section details our choice for time series clustering.

Choice of time Series Clustering Since the context of a user is represented by d time series, a solution could be to transform these series by d clusters. In our case, we decided to use a partitioning approach based on the similarity between items. Partitioning methods are computationally efficient and can handle large A/B-Test datasets with ease. To implement such a method, we needed to set a similarity measure. Indeed, in time series clustering methods, if one wants to compare series with irregular sampling or of different sizes, particular attention must be given to the choice of similarity measure. There are many methods for measuring similarity between time series. The most well-known method is Euclidean distance, which involves calculating the sum of the squared distances between the corresponding elements of the considered sequences at each time step. If this distance is commonly admitted, it cannot be used in our case because it requires that the series be the same length. Length resampling would be problematic for a marketing application as it would risk losing the evolutionary aspect of the customer path. However, in our work, we place ourselves in a case where the observation of the context variables can be irregular. For example, one visitor may visit the site more frequently than another. The similarity measure DTW (dynamic time warping) is a metric between two series of different sizes widely recognized as relevant for many application domains. The DTW method matches the elements of the sequences by aligning them in a way that respects the total order of the sequence of values, without crossing the associations (see Alg. 2 in Appendix). DTW can be sensitive to noise, but this is not a problem in our web application where the data is not noisy. A warping path is constructed by computing the minimum cumulative distance between all possible pairs of points in the two-time series. It is usually calculated using the Euclidean dis-

tance, but other distance metrics can also be used. Once the warping path is obtained, the DTW distance is computed as the sum of the distances along the path. The objective function optimally solved by DTW corresponds to the minimization of the sum of the costs of the different associations. Fig 8 in Appendix illustrates the concept of the similarity measure DTW.

After defining the similarity metric used in our approach, we propose to use an averaging method based on the barycentre (Dynamic Barycenter Average, see Alg. 3 in Appendix) [17]. Once the series have been aligned using DTW and represented by their average series using DBA, the K-means divide the data into K clusters by similarity, where K is a predefined number (see Alg. 4 in Appendix). It assigns each series to the cluster whose average series is closest to it in terms of DTW similarity. Our number of clusters is based on the Silhouette score: it measures how similar a series is to each other. The Silhouette score, denoted as $S(i)$, is computed using the formula :

$$S(i) = \frac{b(i) - a(i)}{\max\{a(i); b(i)\}}$$

where $a(i)$ is the average DTW distance from the ith series to others in the same cluster, and $b(i)$ is the average distance from the ith series to series in the nearest cluster to which it does not belong. The optimal number of clusters is the one that maximizes the average Silhouette score. A higher Silhouette score indicates better-defined clusters. Our experiments have demonstrated that this is the most reliable indicator for determining the number of clusters (see Experimentation section). By combining these three components, the DBA-DTW-kmeans method identifies clusters of similar profiles, taking into account temporal variations and using a more flexible similarity measure. The Fig 2 shows an example with centroids obtained from DBA-DTW-Kmeans on the preprocessing step (774 visitors) on AB Tasty Dataset 1. We apply DBA-DTW-kmeans to `presence_time_serie`, `connection_time_serie` and `time_spend_time_serie` that describe visitors to AB Tasty dataset 1.

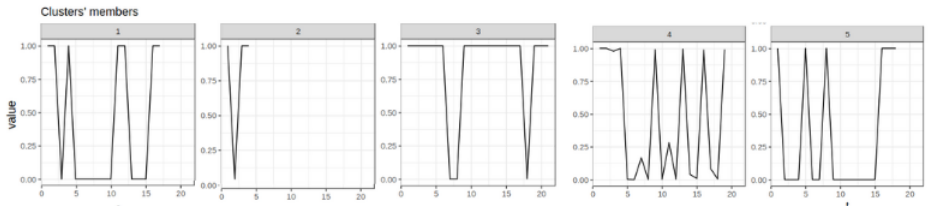

Fig. 2: The series represents 5 centroïds based on the presence/non-presence per day series associated with visitors (from the first day of their visit to the site to the day they appeared on the page being tested). If the visitor does not log on during the day, the series is set to 0.

Dynamic allocation Each time series is replaced by the centroid to which it is closest. For DBA-Ctree-Ucb the tree model g is learned from $\mathcal{L}$, where each time series has been substituted with the number of clusters to which it belongs. The second step of these two algorithms corresponds to the online A/B-Test step involving Ctree-Ucb and LinUcb. Cluster replacement can then be directly utilized by a contextual bandit model. Consequently, in the DBA-LinUCB/DBA-Ctree-Ucb version, the allocation is down according to a LinUcb/Ctree-Ucb modelling (see Alg. 5 and Alg. 6 in Appendix). The global schema of our idea is drawn in Fig 9 in Appendix. The offline part is common to both methods: the past log is used to generate the clusters. An additional step for DBA-Ctree-Ucb also creates the segmentation tree that will be used in the online part. The online part corresponds to the start of the test. The time series of new items are replaced by the nearest cluster, and the allocation of variation depends on the bandit strategy used (according to the segmentation tree g for DBA-Ctree-Ucb and according to the upper bound of a linear regression for DBA-LinUCB). The DBA-LinUCB algorithm uses a linear approach to estimate the parameters between clusters and potential rewards. This approach assumes that this reward function can be approximated by a linear function. The DBA-Ctree-Ucb divides the cluster sets into distinct groups, enabling the capture of more complex and nonlinear function links between the context and rewards. It evaluates each group and identify optimal variation according to each group by the Ucb strategy. This approach facilitates faster exploration when one or more groups demonstrate sensitivity to the test while avoiding unnecessary slowdown in cases where the test would yield no changes for some items (e.g. Fig 3).

4 Experiments

Here, we present the results of the DBA-Ctree-Ucb and DBA-LinUCB models on real-world datasets. We compare our algorithms with Ctree-Ucb [6] and LinUcb [1] which require transforming each context time series by taking the average of its values. These experiments observe whether clustering (which is defined by a partial observation of the data) significantly reduces regret. The choice of the number of clusters is first evaluated using the Silhouette index [19]. All experiments are reproducible in our R package from our GitHub repository[4].

AB Tasty Dataset 1 We present the results on the dataset introduced in section 3.1. The available payoffs are those associated with the pages presented in the case of a static allocation, so observing the regret requires a k-nearest neighbors replacement technique : we replaced the missing values by sampling from the visitors at a minimal distance (<10% of the sequence for each series). If no similar visitor profile exists, a reward is randomly drawn from the alternative page dataset. The "k-nearest neighbors" method, employed to replace missing data, keeps the correlation between the time series and the corresponding generated

[4] https://github.com/manuclaeys/banditWithR

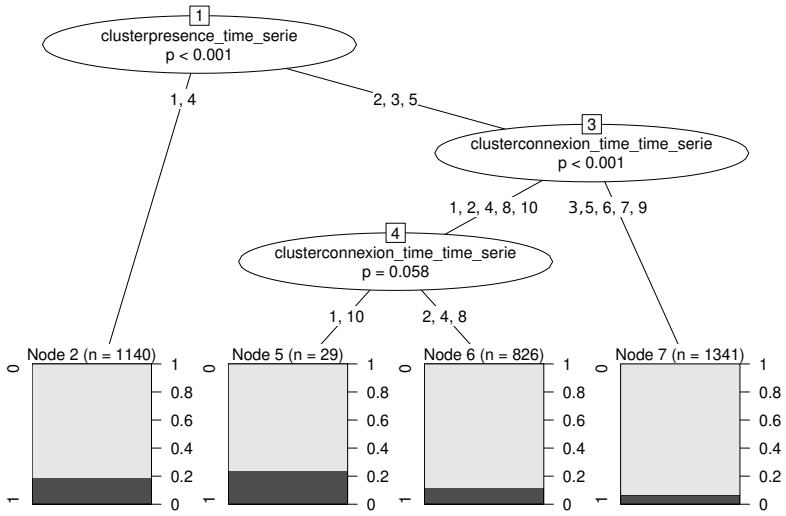

Fig. 3: Preprocessing tree obtained by DBA-Ctree-Ucb on AB TASTY dataset 1. Each leaf represents a click rate. 4 groups are identified from the pre-processing dataset.

rewards. Unlike predefined theoretical replacement models, this method does not rely on such models, which may introduce a strong dependency between missing values and their replacements in small datasets; however, this is not the case in our experiment.

Number of clusters and regret The cumulative regret of DBA-Ctree-Ucb and DBA-LinUCB is influenced by the choice of number of clusters, as depicted in the three graphs of Fig 4. Our method chooses to parameterize the number of clusters based on the Silhouette index (here 5, 5, 10 for the three series). When employing DBA-Ctree-Ucb, the use of an inference tree to partition visitors into subgroups introduces the possibility of grouping clusters with no significant differences (statistically) if the number of clusters is set too large (see Fig 3). On the other hand, for DBA-LinUCB, the learning time and regret increase linearly with the number of clusters. Consequently, choosing an inappropriate value will lead to a longer learning process and result in a greater level of regret. The centroids generated by DBA-Ctree-Ucb and DBA-LinUCB must be interpretable from a marketing perspective by the user. Those presented in Fig. 2 represent, for example, visitor patterns such as: the regular visitor (3), those prospecting for a product (1, 4, and 5) with varying intervals between visits, and those who do not return (2). By coupling with other types of series DBA-Ctree-Ucb identifies "high-potential" or "low-potential" visitors. In the generated tree (see Fig. 3), DBA-Ctree-Ucb identifies 4 visitor groups, each with a varying CTR. New visitors placed in these groups had an independent allocation by Ucb

policy, and the user was able to identify at the end of the test which variation was better for each group. It should be noted that generally, groups with very low CTR are usually less sensitive to the test. For example, visitors in group 7 (see Fig. 3) showed no preference between variations. This visitor group was highly represented in the traffic, and its 'isolation' helped accelerate learning for the other groups.

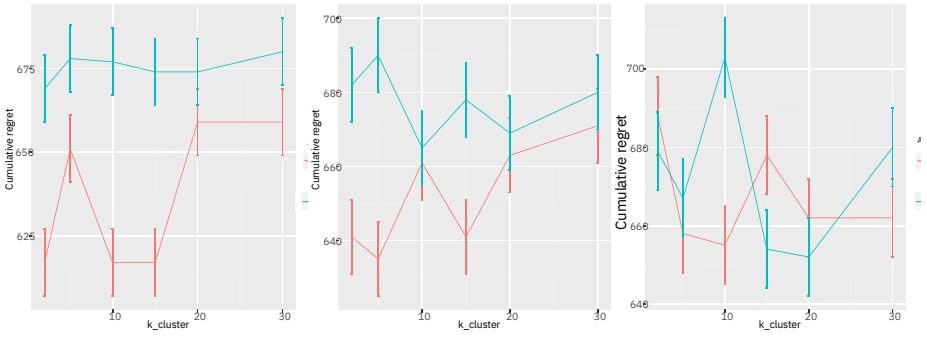

Fig. 4: Cumulative regret obtained by DBA-Ctree-Ucb (red) and DBA-LinUCB (light blue) over number of clusters for AB Tasty dataset 1 on `presence_time` (left), `connexion_time` (middle) and `time_spend` (right). For each graph, the algorithms are trained using only one of the three series as feature context information. In comparison, Silhouette's indexes suggest the following settings (`presence_time` =5,`connexion_time` = 5,`time_spend` = 10)

The evolution of cumulative regret according to time for all methods is presented in Fig. 5. It is observed that the gap between DBA-Ctree-Ucb and the other methods widens during learning. The cumulative regret (to minimize) according to clusters setting for DBA-Ctree-Ucb and DBA-LinUCB is shown in Fig. 4 and confirms that the silhouette index is a good indicator for choosing the number of clusters before starting the test. The average click-through rates (to maximize) as a function of the number of clusters are also referenced in Tab 1 for comparing all methods with different settings (number of clusters, web page used for preprocessing). The lower performance of DBA-LinUCB compared to DBA-Ctree-Ucb can be explained by the fact that if the number of clusters is large, DBA-LinUCB will require more data than DBA-Ctree-Ucb. The differences in terms of regret may seem small, but as the objective of the test is a click to buy, the probability of success is relatively low, regardless of the variation displayed. The average click rate at the end of A/B-Test was 14.14% for DBA-Ctree-Ucb, 12.79% for Ctree-Ucb and $< 12\%$ for DBA-LinUCB, LinUcb and UNIFORM (see Table 1).

Complementary experiment : AB Tasty 2 We provide a complementary experiment on another AB Tast dataset (AB Tasty dataset 2). These data have the

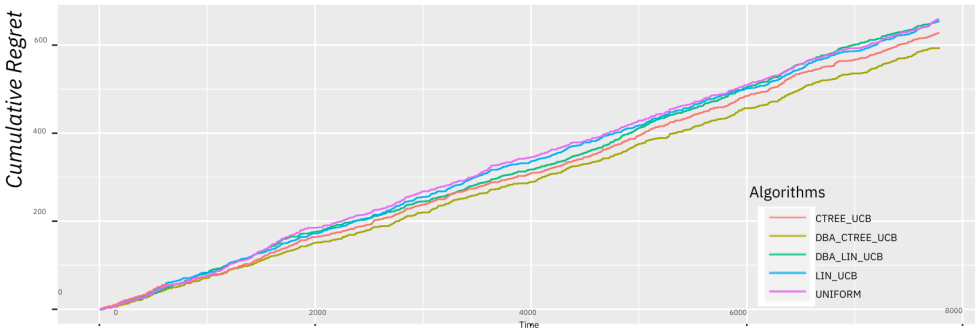

Fig. 5: Cumulative Regret UCB, LINUCB, CTREE-UCB, DBA-CTREE-UCB and DBA-LINUCB on AB TASTY dataset 1

same type of time series and come from a video streaming website. The objective of the AB Test is to increase the click-through rate of the tested page. To learn the clusters and groups, we test several configurations. One uses 30% of the visitors (1591), and the other uses all the visitors (5306). The reader can see the results in the table 1, in particular the interest of the two new approach proposed to increase the average click rate at the end of the experiment. DBA-CTREE-UCB identified 2 visitor groups (see Fig 6) where CTREE-UCB did not identify any groups. The Silouhette index suggested setting 10,5 and 10.

Localization Data for Posture Reconstruction In this experiment, we assess the performance of DBA-CTREE-UCB and DBA-LINUCB in a non-e-commerce setting using the 'Localization Data for Posture Reconstruction' dataset [10]. The dataset comprises 164860 positional measurements from five patients, each represented by three time series $(V1, V2, V3)$ capturing displacements along the x, y, and z axes. The objective is to differentiate between 'sitting' and 'sitting on the floor' activities based on these measurements. Our A/B test categorizes activities as A ('sitting on the floor') and B ('sitting'), with rewards assigned for successful detection. We frame this as a classification problem, aiming to identify patient activities accurately without imputation, unlike traditional e-commerce datasets. The study involves 1000 items and various bandit configurations. The results are on Tab 2. The parameter settings are according to the max Silhouette index: $K_{V1} = 10$, $K_{V2} = 10$, $K_{V3} = 10$. The regret comparison is shown in Figure 7 provides additional evidence to support the effectiveness of the method effectiveness for signal-type series. For this dataset, DBA-LINUCB achieves the best performance. The lower performance of DBA-CTREE-UCB can be explained by the fact that the sensors are not strongly correlated with each other, and the choice of sensor used for learning strongly influences the algorithm's performance.

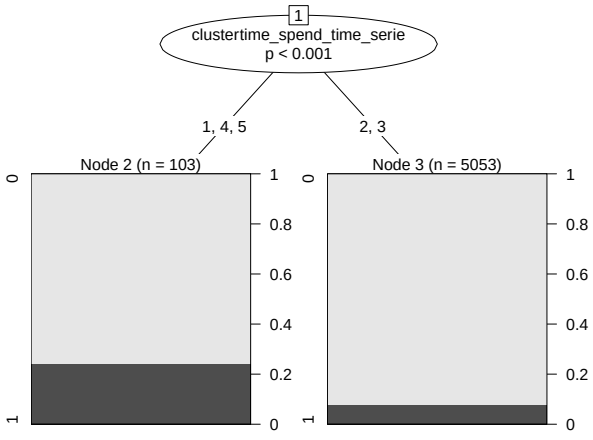

Fig. 6: Preprocessing tree obtained by DBA-CTREE-UCB on AB TASTY dataset 2. Each leaf represents a click rate. 2 groups are identified from the pre-processing dataset.

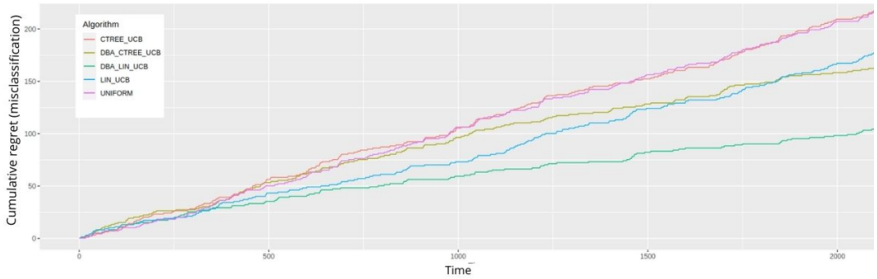

Fig. 7: Cumulative regret UCB, LINUCB, CTREE-UCB, DBA-CTREE-UCB and DBA-LINUCB for Position Dataset

5 Discussion of results and conclusion

The choice of cluster number K, based on the Silhouette index, indicates that the algorithm is suitable for parameter selection. We tested 6 other quality indices, none of which yielded such results. One crucial aspect of our method is the reduction in context size, which leads to a significant decrease in computation time. Since the cluster identification component is executed offline, it does not decrease the efficiency of the online A/B-TEST test step. The DBA-CTREE-UCB method enables the grouping of previously learned clusters if the reward distributions (on variation "A") are statistically identical. Therefore, setting the number of clusters too large has minimal impact on the regret, as clusters with identical

	AB Tasty Dataset 1		AB Tasty Dataset 2	
	$\text{Conf}_{30,70}$			
DBA-Ctree-Ucb Nb clusters	$\mathcal{L} = P_1$	$\mathcal{L} = P_2$	$\mathcal{L} = P_1$	$\mathcal{L} = P_2$
3;5;5	**13.80±1.1%**	12.56±1.1%	**9.53±1%**	**9.53±1%**
5;5;10	**14,14±1.2%**	**13.47±1.2%**	**9.53±1%**	**9.53±1%**
10;5;10	**13.67±1.2%**	**12.82±1%**	**9.92±1%**	8.66±1%
DBA-LinUCB Nb cluster				
3;5;5	11,61±1.1%		8.39±1%	
5;5;10	11,89±1.2%		8.51±1%	
10;5;10	11.81±1.2%		9.19%	
Ctree-Ucb	12.79%±1%	11.47%±1%	9.07%	9.09%
LinUcb	11.56%±1%		8.57±1%	
UNIFORM	11.49%±1%		8.26%±1%	

Table 1: Average click rate at the end of the test according to different settings for AB Tasty dataset 1 and 2

	Localization Dataset		
	$\text{Conf}_{30,70}$		
DBA-Ctree-Ucb Nb cluster	$\mathcal{L} = V1$	$\mathcal{L} = V2$	$\mathcal{L} = V3$
5 ;5 ;10	68.9% ±3%	44.1% ±4%	43,6% ±3%
10 ;10 ;10	69,8% ±3%	58.6% ±4%	66,8% ±3%
10 ;15 ;15	68,8%±4%	57.3%±4%	35,6%±4%
DBA-LinUCB Nb cluster			
5 ;5 ;10	**76% ±3%**		
10 ;10 ;10	**81,7% ±3%**		
10 ;15 ;15	**74,6% ±3%**		
LinUcb	66,2% ±2%		
Ctree-Ucb	33.7%±3%	32.2±3%	31.5±4%
UNIFORM	32,5% ±1%		

Table 2: Average classification rate at the end of the test according to different settings for Localization Dataset

reward distributions are grouped together. Our experiments also demonstrated that DBA-Ctree-Ucb and DBA-LinUCB facilitate business interpretation of clusters. For instance, separating a "perfect prospect" from a "visitor who arrived by mistake" can be challenging. Their visits are nearly short before they reach the test page. By combining different types of clusters (one based on presence and the other on time spent on the site) and predicting their click probabilities, we can differentiate between these two profiles and determine the most suitable variation. DBA-Ctree-Ucb/DBA-LinUCB appears to be a more advantageous method than Ctree-Ucb/LinUcb in terms of regrets. Clusters allow for the construction of more homogeneous groups in terms of reward distribution rather than relying solely on series averages. Separating groups based on series mean values makes learning highly sensitive to extreme values, which the clustering model avoids. However, DBA-Ctree-Ucb requires a correlation between the earnings of different variations: the earnings distribution of a group, whether on variation A or B, follows the same distribution (with the same variance), but the means may differ. In a further work we will show how this pre-processing step has helped the user to create more personalized variation B by generative DNN.

Bibliography

[1] Agarwal, A., Dudík, M., Kale, S., Langford, J., Schapire, R.E.: Contextual Bandit Learning with Predictable Rewards. ArXiv e-prints (feb 2012)

[2] Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning **47**(2), 235–256 (May 2002)

[3] Avadhanula, V., Colini-Baldeschi, R., Leonardi, S., Sankararaman, K.A., Schrijvers, O.: Stochastic bandits for multi-platform budget optimization in online advertising. CoRR (2021)

[4] Bastani, H., Bayati, M.: Online decision-making with high-dimensional covariates. SSRN Electronic Journal (01 2015)
[5] Bietti, A., Agarwal, A., Langford, J.: A contextual bandit bake-off. JLMR (2021)
[6] E. Claeys, P. Gancarski, M. Maumy-Bertrand, and H. Wassner: Dynamic allocation optimization in a/b-tests using classification-based preprocessing. IEEE TKDE **35**(1), 335–349 (2021)
[7] Fabijan, A., Dmitriev, P., Arai, B., Drake, A., Kohlmeier, S., Kwong, A.: A/b integrations:7 lessons learned from enabling a/b testing (2023)
[8] Gentile, C., Li, S., Kar, P., Karatzoglou, A., Zappella, G., Etrue, E.: On context-dependent clustering of bandits. In: Proceedings of the 34th International Conference on Machine Learning (2017)
[9] Hothorn, T., Hornik, K., Zeileis, A.: Unbiased recursive partitioning: A conditional inference framework. Journal of Comput and Graphical Statistics **15**, 651–674 (09 2006)
[10] Kaluza, B., Mirchevska, V., E. Dovgan, M.L., Gams, M.: UCI machine learning repository, an agent-based approach to care in indep. living (2010)
[11] Kaufmann, E., Cappé, O., Garivier, A.: On the Complexity of A/B Testing. ArXiv e-prints (may 2014)
[12] Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics **6**(1), 4 – 22 (1985)
[13] Lattimore, T., Szepesvári, C.: Bandit algorithms (2020)
[14] Li, L., Chu, W., Langford, J., Schapire, R.E.: A contextual-bandit approach to personalized news article recommendation pp. 661–670 (2010)
[15] Mahadik, K., Wu, Q., Li, S., Sabne, A.: Fast distributed bandits for online recommendation systems. Proceedings of the 34th ACM International Conference on Supercomputing (2020)
[16] Maillard, O.A., Mannor, S.: Latent Bandits. (Jan 2014), extended version of the paper accepted to ICML 2014.
[17] Petitjean, F., Ketterlin, A., Gancarski, P.: A global averaging method for dtw, with applications to clustering. Pattern Recognition **44**, 678– (03 2011)
[18] Rotman, M., Wolf, L.: Energy regularized rnns for solving non-stationary bandit problems (2023)
[19] Rousseeuw, P.: Rousseeuw, p.j.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. JCAM (1987)
[20] Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika (1933)
[21] Xu, D., Yang, B.: On the advances and challenges of adaptive online testing. WSDM'22 (2022)
[22] Zhang, Z., Yang, J., Ji, X., Du, S.S.: Variance-aware confidence set: Variance-dependent bound for linear bandits and horizon-free bound for linear mixture MDP. CoRR (2021)
[23] Zhou, D., Li, L., Gu, Q.: Neural contextual bandits with upper confidence bound-based exploration. CoRR (2019)
[24] Zhou, X., Ji, B.: On kernelized multi-armed bandits with constraints. In: Advances in Neural Information Processing Systems. Curran Associates, Inc. (2022)

Appendix

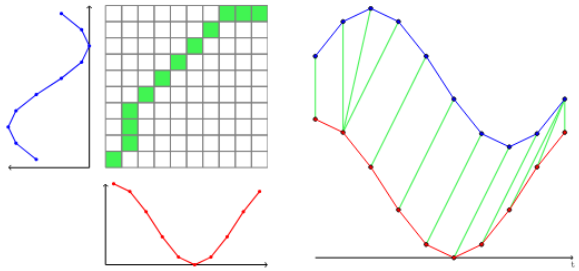

Fig. 8: Similarity of two series by DTW. (left) Matrix and optimal alignment path calculated by DTW. (right) Resulting alignment of the two sequences

Algorithm 1 CTREE algorithm

Require:

– An Alpha risk hyperparameter $\alpha_{\text{CTREE}} \in]0, 1[$

– A dataset of features $c_t \in \mathcal{C}$ and response $r_t \in \mathcal{X}$.

Calculate the the test statistics $stat_{j_0}$ for the observed data

Permute the observation in the node

Calculate $stat$ for all permutations

Calculate the p-values (from the number of test statistics $stat$, where $|stat| > |stat_0|$)

Correct p-values for multiple testing according a Bonferroni correction.

if H_0 (no statistical difference between distributions) are not rejected (p-value $> \alpha_{\text{CTREE}}$ for all X_j) **then return**

Select feature X_j^* with the strongest association (smallest p-value)

Search for the best split of X_j^* (maximize test statistic $stat_0$) and partition data

Apply CTREE to both of the new partitions

Output: A hierarchical partitioning g function

Algorithm 2 Algorithm DTW

Require:

– A sequence $S1 = \langle s1_1, \dots s1_{|S1|} \rangle$.
– A sequence $S2 = \langle s2_1, \dots s2_{|S2|} \rangle$.
– A cost matrix **costmat** of size $|S1| \times |S2|$ initialized to 0.

costmat$[1,1] \leftarrow \delta(s1_1, s2_1)$
for $i \in \{2, \dots, |S1|\}$ **do**
    costmat$[i,1] =$ **costmat**$[i-1,1] + \delta(s1_i, s2_1)$
for $j \in \{2, \dots, |S2|\}$ **do**
    costmat$[1,j] =$ **costmat**$[1,j-1] + \delta(s1_1, s2_j)$
for $i \in \{2, \dots, |S1|\}$ **do**
    for $j \in \{2, \dots, |S2|\}$ **do**

$$\textbf{costmat}[i,j] = \delta(s1_i, s2_j) + \min \left\{ \begin{array}{c} \textbf{costmat}[i-1,j] \\ \textbf{costmat}[i,j-1] \\ \textbf{costmat}[i-1,j-1] \end{array} \right\}$$

Output: costmat

Algorithm 3 Algorithm D.B.A

Require:

– A learning set $\mathcal{L}$.
– $S_{\text{aver}} = \langle s_{\text{aver},1}, \dots, s_{\text{aver},|S_{\text{aver}}|} \rangle$ the initial average sequence.
– $S1 = \langle s1_1, \dots, s1_{|S1|} \rangle$ the first sequence of the set $\mathcal{L}$.
– ...
– $Sn = \langle sn_1, \dots, sn_{|Sn|} \rangle$ the last sequence of the set $\mathcal{L}$.

Let m the maximal size of all sequences in $\mathcal{L}$.
Let $assocTab$ an empty table of size $|S_{\text{aver}}|$ contain in each cell l all the elements associated with l in S_{aver}.
Let **costmat** $\in \mathbb{R}^{m \times m}$ the cost matrix for DTW (See Alg 2).
for $Si \in \mathcal{L}$ **do**
    costmat $\leftarrow DTW(S_{\text{aver}}, Si)$
    $l \leftarrow |Si_{\text{aver}}|$
    $j \leftarrow m$
    while $l \geqslant 1$ or $j \geqslant 1$ **do**
        $assocTab[l] \leftarrow assocTab[l] \cup si_j$
        $(l; j) \leftarrow$ second(**costmat**$[l; j]$)
for $l \in 1 \dots |Sn|$ **do**
    $S^+_{\text{aver}} =$ barycenter$(assocTab[l])$

Output: S^+_{aver}

Note that D.B.A. utilizes a matrix that returns for each cell (access with $first()$) function all elements associated with this cell (access with $second()$).

Algorithm 4 Pseudo code of K-means D.B.A DTW

Require:

- A number of desired cluster K
- A learning set $\mathcal{L} = \{S1, \dots Sn\}$ of n sequence.
- $Iter$ maximal number of iteration

Randomly choose a cluster for each time series
while $Iter$ **do**
    Calculate the mean for each cluster according to D.B.A (See Alg. 3)
    Assign each sequence to the closest centroid according do DTW (See Algo. 2)

Output: A set of centroid

Algorithm 5 Algorithme DBA-CTREE-UCB

Require:

- A learning set $\mathcal{L}_{\text{pre-test\_A}}$
- A set of T items $\mathcal{L}_{\text{test\_A\_B}}$ to be tested.
- $vectClust$ a d dimensional vector of desired number of clusters for each d type of series.
- An accepted risk $\alpha_{\text{CTREE}} \in [0, 1]$.
- A positive parameter $\alpha \in \mathbb{R}^+$.

Init **matrixClusters** an empty matrix of all centroides
for Each time series of type $j \in d$ in $\mathcal{L}_{\text{pre-test\_A}}$ **do**
    Subset j-type series in $\mathcal{L}_{\text{pre-test\_A},j}$
    matrixClusters ←K-means-DBA-DTW($\mathcal{L}_{\text{pre-test\_A},j}$), add $vectClust[j]$ centroïds
Generate a $k()$ function that return a d-vector of the closest clusters according to **matrixClusters**
for Each item $\mathbf{c_t} \in \mathcal{L}_{\text{pre-test\_A}}$ **do**
    $c_t \leftarrow k(\mathbf{c_t})$
Generate a $g()$ function that return a leaf group association according to CTREE($\mathcal{L}_{\text{pre-test\_A}}, \epsilon$)
while $\mathbf{c_t} \sim \mathcal{L}_{\text{test\_A\_B}}$ **do**
    Receive an item $\mathbf{c_t}$ from $\mathcal{L}_{\text{test\_A\_B}}$
    Assign $\mathbf{c_t}$ to a group according $g \circ k(\mathbf{c_t})$
    if $N_{a,g \circ k(\mathbf{c_t})}(t) = 0$ **then**
        $a_t = a$
    $a_t = \text{argmax}_{a \in \mathcal{A}} \{\hat{\mu}_{a,g \circ k(\mathbf{c_t}),t} + \alpha \sqrt{\frac{2 * log(\sum_{a \in \mathcal{A}} N_{a,g \circ k(\mathbf{c_t})}(t))}{N_{a,g \circ k(\mathbf{c_t})}(t)}}\}$
    Assign the item $\mathbf{c_t}$ to arm a_t
    Receive a reward $r_{\mathbf{c_t},a_t}$
    Update $\hat{\mu}_{a,g \circ k(\mathbf{c_t}),t}$ and $N_{a,g \circ k(\mathbf{c_t})}(t)$
    Remove $\mathbf{c_t}$ from $\mathcal{L}_{\text{test\_A\_B}}$

Output: A sequence of T arm choices and rewards

Algorithm 6 Algorithm DBA-LinUCB

Require:

– A learning set $\mathcal{L}_{\text{pre-test\_A}}$
– A set of T items $\mathcal{L}_{\text{test\_A\_B}}$ to be tested.
– *vectClust* a d dimensional vector of desired number of clusters for each d type of series.
– A positive parameter $\alpha \in \mathbb{R}^+$.

Init **matrixClusters** an empty matrix of all centroides
for Each time series of type $j \in d$ in $\mathcal{L}_{\text{pre-test\_A}}$ **do**
    Subset j-type series in $\mathcal{L}_{\text{pre-test\_A},j}$
    matrixClusters ←K-means-DBA-DTW($\mathcal{L}_{\text{pre-test\_A},j}$), add $vectClust[j]$ centroïds
Generate a $k()$ function that return a d-vector of the closest clusters according to **matrixClusters**
Define $D = |\mathcal{K}|$ as the number of possible cluster (hot-one encording)
for $a \in \mathcal{A}$ **do**
Initialize $A_a = I_D$, the identity matrix of size D by D of arm a
Initialize $b_a = 0_D$, a null vector of size D
while $\mathbf{c_t} \sim \mathcal{L}_{\text{test\_A\_B}}$ **do**
    Receive an item $\mathbf{c_t}$ from $\mathcal{L}_{\text{test\_A\_B}}$
    for $a \in \mathcal{A}$ **do**
        Update parameters $\hat{\theta}_{t,a} = A_a^{-1} b_a$
    end for
    Choose $a_t = \text{argmax}_{a \in \mathcal{A}} \hat{\mu}_{a,t} = k(\mathbf{c_t})^T \hat{\theta}_{a,t} + \alpha \sqrt{k(\mathbf{c_t})^T A_a^{-1} k(\mathbf{c_t})}$
    Receive a reward $r_{k(\mathbf{c_t}),a_t}$
    Update $A_{a_t} = A_{a_t} + k(\mathbf{c_t}) k(\mathbf{c_t})^T$
    Update $b_{a_t} = b_{a_t} + r_{\mathbf{c_t},a_t} k(\mathbf{c_t})$

Output: A sequence of arm choices and rewards

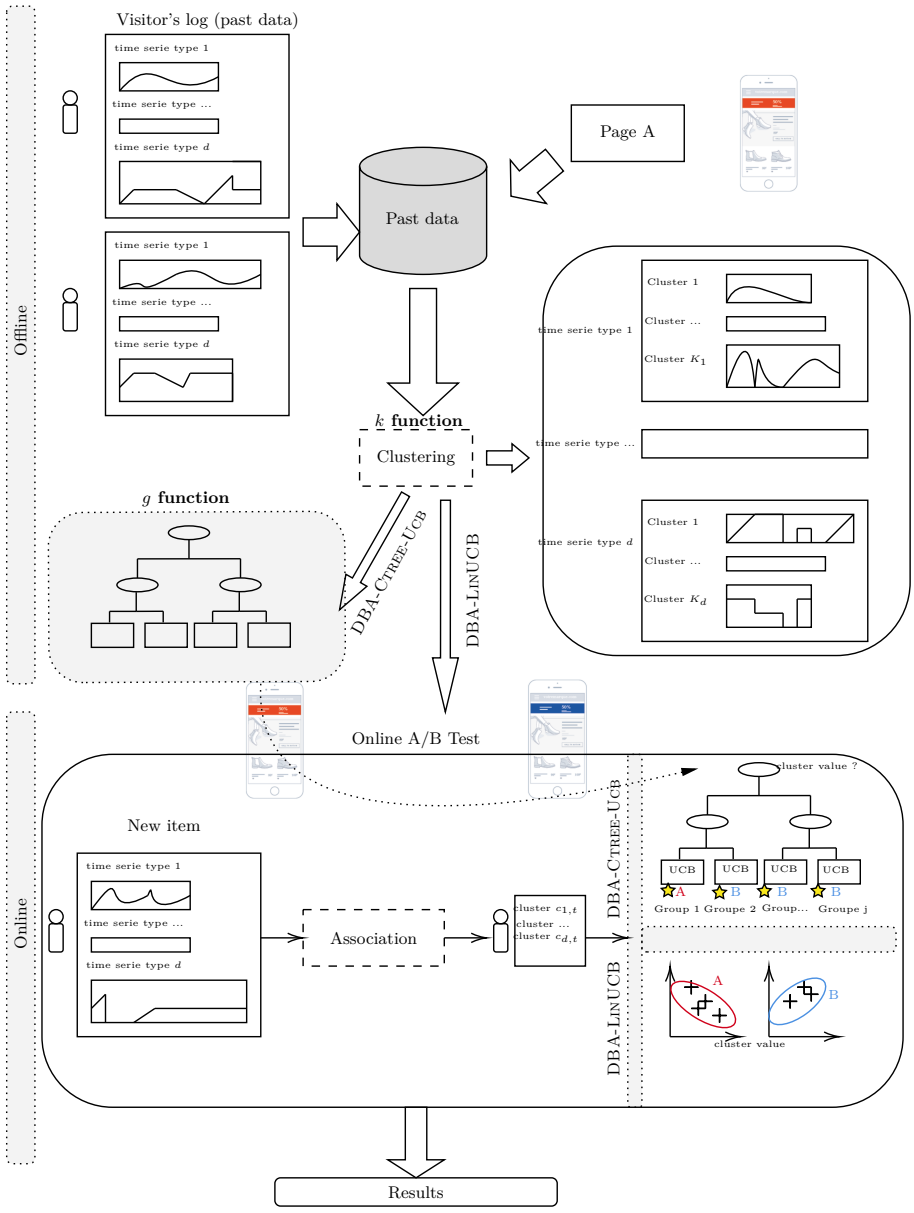

Fig. 9: Offline/online approach to integrate time series in A/B tests

DBA-Ctree-Ucb convergence

If a group of items $\widetilde{g}$ have a $\sigma_{\widetilde{g}}$-subgaussian distribution of reward for a s number of trials with $s \leq n$ and a given $1-\delta$ probability, the real average reward of an arm a is bounded by:

$$\frac{1}{s}\sum_{t=1}^{s} X_{a,t,\widetilde{g}} \leq \mu_{a,\widetilde{g}} + \sqrt{\frac{2\sigma^2_{a,\widetilde{g}} \log\frac{1}{\delta}}{s}} \tag{2}$$

with traditionally $\delta = \frac{1}{n^2}$ [2].

We assume the case where a prediction function assign items to a group $\widetilde{g}$ where it's reward follows a $\sigma_{\widetilde{g}}$-gaussian distribution regardless of the arm applied. The experiences in the next section indicate that this hypothesis can be confirmed in practice. Define a reward at iteration t:$X_t = X_{A_t,\widetilde{g}}$ and $N_{A_t,\widetilde{g}}(t)$ the number of time when the arm A_t in group $\widetilde{g}$ have been played. By definition of regret:

$$R_{n,\widetilde{g}} = \sum_{a \in \mathcal{A}} \Delta_{a,\widetilde{g}} \mathbb{E}[N_{a,\widetilde{g}}(n)] \tag{3}$$

.

With $\Delta_{a,\widetilde{g}} = \mu_{a^\star,\widetilde{g}} - \mu_{a\neq a^\star,\widetilde{g}}$. The regret increase of $\Delta_{a,\widetilde{g}}$ when

(a) $\text{U.C.B}_{a\neq a^*}(\widetilde{g},t) > \mu_{a^*,\widetilde{g}}$
(b) $\text{U.C.B}_{a=a^*}(\widetilde{g},t) < \mu_{a^*,\widetilde{g}}$

Let's $G_{a,\widetilde{g}}$ the event of (a) or (b) define by a constant $u_{a,y}$:

$$G_{a,\widetilde{g}} = \{\mu_{a^*,\widetilde{g}} < \min_{t\in[n]} \text{U.C.B}_{a=a^*}(\widetilde{g},t)\} \cap \{\hat{\mu}_{a,\widetilde{g},u_i} + \sigma_{\widetilde{g}}\sqrt{\frac{1}{u_a}\log\frac{1}{\delta}} < \mu_{a^*,\widetilde{g}}\}$$

where if $G_{a,\widetilde{g}}$ happens, the regret converge to a finite value and so if $G_{a,\widetilde{g}}$ occur, then $N_{a,\widetilde{g}}(n) \leq u_{i,\widetilde{g}}$. Let $G^C_{a,\widetilde{g}}$ the complementary event.

$$\begin{aligned}\mathbb{E}[N_{a,\widetilde{g}}(n)] &= \mathbb{E}[\mathbb{I}\{G_{a,\widetilde{g}}\}N_{a,\widetilde{g}}(n)] + \mathbb{E}[\mathbb{I}\{G^C_{a,\widetilde{g}}\}N_{a,\widetilde{g}}(n)] \\ &\leq u_{a,\widetilde{g}} + \mathbb{P}[G^C_{a,\widetilde{g}}]n\end{aligned}$$

Bound of complementary event $G^C_{a,\widetilde{g}}$ is done in two parts:

$$G^C_{a,\widetilde{g}} = \Big\{\mu_{a^*,\widetilde{g}} \geqslant \min_{t\in[n]} \text{U.C.B}_{a=a^*}(\widetilde{g},t)\} \cup \{\hat{\mu}_{a,\widetilde{g},u_a} + \sigma_{\widetilde{g}}\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \mu_{a^*,\widetilde{g}}\Big\}$$

$$\begin{aligned}\{\mu_{a^*,\widetilde{g}} \geqslant \min_{t\in[n]} \text{U.C.B}(\widetilde{g},t)\} \subset \cup_{s\in[n]}\{\mu_{a^*,\widetilde{g}} \geqslant \hat{\mu}_{a^*,\widetilde{g},s} \\ + \sigma_{\widetilde{g}}\sqrt{\frac{2}{s}\log\frac{1}{\delta}}\} \\ \mathbb{P}[\mu_{a^*,\widetilde{g}} \geqslant \min_{t\in[n]} \text{U.C.B}_{a=a^*}(\widetilde{g},t)] \leq \sum_{s=1}^{n}\mathbb{P}[\mu_{a^*,\widetilde{g}} \geqslant \hat{\mu}_{a^*,\widetilde{g},s} \\ + \sigma_{\widetilde{g}}\sqrt{\frac{2}{s}\log\frac{1}{\delta}}] \\ \leq n\delta\end{aligned}$$

And so $\mathbb{P}[G^C_{a,\widetilde{g}}] \leq n\delta + \mathbb{P}[\hat{\mu}_{a,\widetilde{g},u_a} + \sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \mu_{a^*,\widetilde{g}}]$. We remind that $\Delta_{a,\widetilde{g}} = \mu_{a^*,\widetilde{g}} - \mu_{a,\widetilde{g}}$ Assume a positive constant c where $\Delta_{a,\widetilde{g}} - \sigma_{\widetilde{g}}\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant c\Delta_{a,\widetilde{g}}$:

$$\begin{aligned}\mathbb{P}[\hat{\mu}_{a,\widetilde{g},u_a} + \sigma_{\widetilde{g}}\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \mu_{a^*,\widetilde{g}}] &= \mathbb{P}[\hat{\mu}_{a,\widetilde{g},u_a} \\ &+ \sigma_{\widetilde{g}}\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} \geqslant \Delta_{a,\widetilde{g}} \\ &+ \mu_{a,\widetilde{g}}] \\ &= \mathbb{P}[\hat{\mu}_{a,\widetilde{g},u_a} - \mu_{a,\widetilde{g}} \geqslant \Delta_{a,\widetilde{g}} \\ &- \sigma_{\widetilde{g}}\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}}] \\ &\leq \mathbb{P}[\hat{\mu}_{a,\widetilde{g},u_a} - \mu_{a,\widetilde{g}} \geqslant c\Delta_{a,\widetilde{g}}] \\ &\leq \exp\left(-\frac{u_a c^2 \Delta^2_{a,\widetilde{g}}}{2\sigma^2_{\widetilde{g}}}\right)\end{aligned} \tag{4}$$

The last line of the equation 4 refers to a bounded error for a random variable following a Gaussian σ distribution. So $\mathbb{P}[G^C_a] \leq n\delta + \exp\left(-\frac{u_a c^2 \Delta^2_{a,\widetilde{g}}}{2\sigma^2_{\widetilde{g}}}\right)$. We looking for u_a that satisfy:

$$\begin{aligned}\Delta_{a,\widetilde{g}} - \sigma_y\sqrt{\frac{2}{u_a}\log\frac{1}{\delta}} &\geqslant c\Delta_{a,\widetilde{g}} \\ u_a &\geqslant \frac{2\sigma^2_{\widetilde{g}}\log\frac{1}{\delta}}{\Delta^2_{a,\widetilde{g}}(1-c)^2}\end{aligned}$$

If $\delta = \frac{1}{n^2}$

$$\mathbb{E}[N_{a,\widetilde{g}}(n)] \leq \frac{2\sigma^2_{\widetilde{g}} \log \frac{1}{\delta}}{\Delta^2_{a,\widetilde{g}}(1-c)^2} + 1 + n \exp\left(-\frac{u_a c^2 \Delta^2_{a,\widetilde{g}}}{2\sigma^2_{\widetilde{g}}}\right) \tag{5}$$
$$\leq \frac{2\sigma^2_{\widetilde{g}} \log \frac{1}{\delta}}{\Delta^2_{a,\widetilde{g}}(1-c)^2} + 1 + n^{-1+\frac{c^2}{(1+c)^2}}$$

The function $-1+\frac{c^2}{(1+c)^2}$ is strictly negative and for any $c < \frac{1}{2}$. The equation 5 can be re written as :

$$\mathbb{E}[N_{a,\widetilde{g}}(n)] \leq \frac{2\sigma^2_{\widetilde{g}} \log n^2}{\Delta^2_{a,\widetilde{g}\frac{1}{2}}} + 2$$

For any group $\widetilde{g}$ the regret is bounded by:

$$R_{n,\widetilde{g}} = \sum_{a \in \mathcal{A}} \Delta_{a,\widetilde{g}} \mathbb{E}[N_{a,\widetilde{g}}(n)]$$
$$= \sum_{a \in \mathcal{A}} \frac{16\sigma^2_{\widetilde{g}} \log n^2}{\Delta_{a,\widetilde{g}}} + 2 \sum_{a \in \mathcal{A}} \Delta_{a,\widetilde{g}}$$

This finally define the cumulative regret for any group $\widetilde{g}$ based on past data from $a = A$ (the original variation to improve). If the conditional inference tree define a set of $\mathcal{G}$ admissible groups where $\forall \widetilde{g} \in \mathcal{G}$, $\mathcal{N}(\nu_{\widetilde{g},a=A}, \sigma_{\widetilde{g},a=A})$ is statistically different (with an accepted risk of ϵ) and $|\mathcal{G}|$ the number of possible groups, the cumulative regret during the A/B-Test is

$$R_n^{\text{DBA-Ctree-Ucb}} = \sum_{\widetilde{g} \in \mathcal{G}} R_{n,\widetilde{g}} = \sum_{\widetilde{g} \in \mathcal{G}} \left(\sum_{a \in \mathcal{A}} \frac{16\sigma^2_{\widetilde{g}} \log n^2}{\Delta_{a,\widetilde{g}}} + 2 \sum_{a \in \mathcal{A}} \Delta_{a,\widetilde{g}} \right)$$
$$\leq |\mathcal{G}| \max_{\widetilde{g} \in \mathcal{G}} [\sigma^2_{\widetilde{g}}] \left(\sum_{a \in \mathcal{A}} \frac{16 \log n^2}{\Delta_{a,\widetilde{g}}} + 2 \sum_{a \in \mathcal{A}} \Delta_{a,\widetilde{g}} \right)$$