Carousel Ads Optimization in Yahoo Gemini Native

Michal Aharon, and Oren Somekh
Yahoo Research, Haifa, Israel
(michala,orens)@verizonmedia.com

Avi Shahar, Assaf Singer, Baruch Trayvas, Hadas Vogel, and Dobri Dobrev
Tech Yahoo, Tel Aviv, Israel
(avis,assafs,btrayvas,hadas,dobri)@verizonmedia.com

ABSTRACT

Yahoo's native advertising (also known as Gemini native) serves billions of ad impressions daily, reaching a yearly run-rate of many hundred of millions USD. Driving Gemini native models for predicting both click probability (pCTR) and conversion probability (pCONV) is Offset - a feature enhanced collaborative-filtering (CF) based event prediction algorithm. The predicted pCTRs are then used in Gemini native auctions to determine which ads to present for each serving event. A fast growing segment of Gemini native is Carousel ads that include several cards (or assets) which are used to populate several slots within the ad. Since Carousel ad slots are not symmetrical and some are more conspicuous than others, it is beneficial to render assets to slots in a way that maximizes revenue. In this work we present a post-auction successive elimination based approach for ranking assets according to their click trough rate (CTR) and render the carousel accordingly, placing higher CTR assets in more conspicuous slots. After a successful online bucket showing 8.6% CTR and 4.3% CPM (or revenue) lifts over a control bucket that uses predefined advertisers assets-to-slots mapping, the carousel asset optimization (CAO) system was pushed to production and is serving all Gemini native traffic since. A few months after CAO deployment, we have already measured an almost 40% increase in carousel ads revenue. Moreover, the entire revenue growth is related to CAO traffic increase due to additional advertiser demand, which demonstrates a high advertisers' satisfaction of the product.

CCS CONCEPTS

• **Information systems** → **Computational advertising**; **Online advertising**;

KEYWORDS

Computational advertising, Gemini native, reinforcement learning, explore exploit, successive elimination, carousel ads optimization

ACM Reference Format:
Michal Aharon, and Oren Somekh and Avi Shahar, Assaf Singer, Baruch Trayvas, Hadas Vogel, and Dobri Dobrev. 2019. Carousel Ads Optimization in Yahoo Gemini Native. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330740

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330740

Figure 1: A typical Gemini native ad captured from Yahoo home-page stream. The ad consists of a title, an image, a description, and a sponsorship notification.

1 INTRODUCTION

Yahoo Gemini native[1] serves users with ads that are rendered to resemble the surrounding native content in *owned and operated* (O&O) and Syndication[2] properties (see Figure 1 for a typical Gemini native ad). In contrast to the search-ads marketplace, users' intent during page visits are unknown in general. Launched five years ago and operating with a yearly run-rate of several hundred million USD[3], Gemini native is one of Yahoo's fastest growing and main businesses. With more than two billion impressions daily, and an inventory of a few hundred thousand active ads at any given time, this marketplace performs real-time *generalized second price* (GSP) auctions that take into account ad targeting, and budget considerations, with SLA (or latency) of less than 80 ms more than 99% of the time.

In order to rank the native ads for an incoming user and her specific context according to the *cost per click* (CPC) price type, a score (or expected revenue) is calculated by multiplying the advertiser's bid and the *predicted click probability* (pCTR) for each active ad. The pCTR is calculated using models that are periodically updated by Offset - a feature enhanced *collaborative-filtering* (CF) based event-prediction algorithm [3][4]. Offset is a one-pass algorithm that updates its latent factor model for every new mini-batch of logged data using a *stochastic gradient descent* (SGD) based learning approach.

Carousel ads are a fast growing segment of Gemini native demand. Carousels are variants of native ads which include several cards (also referred to as assets) used to populate several slots (see Figure 2 for a carousel ad on mobile and desktop devices). Since slots are not symmetrical and some are more conspicuous than others (e.g., the first slot is much bigger on desktops while the first slot is the only one completely visible without scrolling on mobile), it is beneficial to optimize the mapping of assets to slots in terms

[1] https://gemini.yahoo.com/advertiser/home
[2] Where Yahoo presents its ads on a third party site and shares the revenues with the site owner.
[3] Due to obvious commercial confidentiality matters, we provide rough estimations or relative numbers regarding traffic sizes and revenues, and present relative performance lifts and rescaled CTRs only.

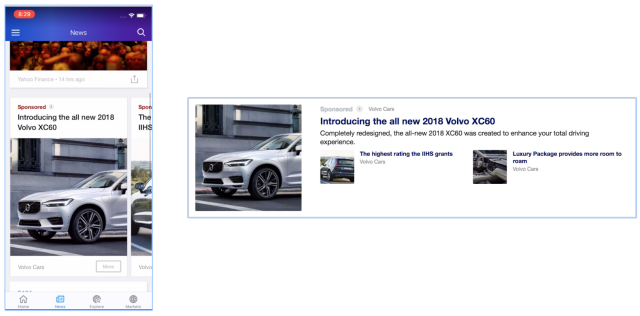


Figure 2: A typical Gemini native Carousel ad on mobile (left) and desktop (right) devices.

of CTR and revenue. The Gemini native marketplace user-interface (UI) enables advertisers to map assets to slots for carousel ads, and also to choose whether the carousel is to be presented using this mapping or to let the system optimize the mapping automatically.

In this work we present the way carousels are rendered in Gemini native to increase CTR and revenue. To keep system resources (i.e., OFFSET model size and system latency) at their current levels, we adopt a two stage solution. During the first stage, the serving system ranks the ads as usual and in case a carousel ad wins the auction, the second stage is invoked and the assets are rendered to slots according to some asset distribution. The asset distribution, calculated for each carousel, is updated every training period by our *carousel assets successive elimination* (CASE) algorithm. Roughly speaking, we start with a uniform distribution, measure the CTR of each asset on slot 1 (the most conspicuous slot), and "eliminate" [4] all assets that have lower CTR than the best asset with a predefined confidence probability (e.g., 90%). We continue this until one asset remains and then declare this carousel as "resolved". The CASE algorithm is a practical and robust version of the celebrated successive elimination approach (e.g., see [11][5]), that is designed to satisfy both system requirements (such as model size and system delay), and inherent system behaviour (such as time varying CTRs).

After demonstrating impressive 8.6% CTR and 4.3% CPM lifts when compared to a control bucket operating without *carousel asset optimization* (CAO)[5], the proposed CASE based CAO system was pushed to production and is serving all Gemini native traffic since. Statistics gathered a few months after deployment, reveals great advertiser satisfaction of this new product, with carousel revenue increase of almost 40%. Moreover, the entire revenue increment stems from additional CAO traffic due to additional advertiser demand.

The main contributions of this work are:

- We introduce a practical web scale optimization problem of rendering assets to slots for maximizing Gemini native carousel ads revenue, under real world constraints.
- We present CASE, a post-auction successive elimination algorithm which is based on asset CTR measurements. The algorithm is simple, practical, robust, and tailored to the existing Gemini native ad ranking system.
- We test our solution in web scale, serving real Gemini native users, and report impressive lifts in terms of CTR and revenue in comparison to human based asset ordering.
- The CASE algorithm based CAO system, is currently fully deployed, serving all Gemini native traffic.

The rest of the paper is organized as follows. In Section 2, we provide relevant background, and discuss related work in Section 3. We state the problem in Section 4 and elaborate on our approach in Section 5. Performance evaluation and the impact of our solution is presented in Section 6 with some statistics. We conclude and consider future work in Section 7. The Appendix includes additional statistics.

2 BACKGROUND

2.1 Gemini Native

Gemini native is one of Yahoo's major businesses, reaching a yearly run-rate of a few hundred million USD in revenue. Gemini native serves a daily average of more than two billion impressions world wide, with SLA (or latency) of less than 80 ms for more than 99% of the queries, and a native ad inventory of several hundred thousand active ads on average. Native ads resemble the surrounding page items, are considered less intrusive to the users, and provide a better user experience in general (see Figure 1 for a typical Gemini native ad on Yahoo home-page stream).

The online serving system is comprised of a massive Vespa[6] deployment, augmented by ads, budget, and model training pipelines. The Vespa index is updated continuously with ad and budget changes, and periodically (e.g., every 15 minutes) with model updates. The Gemini native marketplace serves several ad price-types including CPC (cost-per-click), oCPC (optimizing for conversions), CPM (cost-per-thousand impressions), and also supports RTB (real-time bidding) in its auctions.

2.2 The OFFSET ad click prediction algorithm

The algorithm driving Gemini native models is OFFSET (One-pass Factorization of Feature Sets): a feature-enhanced collaborative-filtering (CF) ad click-prediction algorithm [3][4]. The predicted click-probability or click-through-rate (pCTR) of a given user u and ad a according to OFFSET is given by[7]

$$pCTR(u,a) = \frac{1}{1+\exp^{-(b+v_u^T v_a)}} \in [0,1]\,, \tag{1}$$

where $v_u, v_a \in \mathbb{R}^D$ denote the user and ad latent factor vectors respectively, and $b \in \mathbb{R}$ denotes the model bias. The product $v_u^T v_a$ denotes the tendency score of user u towards ad a, where a higher score translates into higher pCTR. Note that $\Theta = \{v_u, v_a, b\}$ are the model parameters learned from the logged data.

Both ad and user vectors are constructed using their features, which enables dealing with data sparsity issues. For ads, a simple sum between the dimension D vectors of the unique creative id, campaign id, and advertiser id may be used. The combination

[4] "Elimination" means allocating minuscule probabilities to eliminated assets.
[5] Rendering carousels using advertisers' predefined assets-to-slots mapping (or ordering).
[6] Vespa is Yahoo's elastic search engine solution.
[7] In this work we focus on ad click prediction models. However, OFFSET is used for other event prediction models such as conversion prediction.

between the different user and user context feature vectors[8] is more elaborate and allow non-linear dependencies between feature pairs (for more details see [4]). The advantage of this principle (where users are presented by their features) over the standard CF approach, is that the model includes only a few thousands latent factor vectors instead of hundreds of millions of unique user latent factor vectors.

To learn the model parameters Θ, Offset minimizes the logistic loss (or LogLoss) of the training data set (i.e., past impressions and clicks) using one-pass *stochastic gradient descent* (SGD). As mentioned earlier, Offset uses an incremental approach where it continuously updates its model parameters with each new batch of training events (e.g., every 15 minutes for the click model). A more elaborate description, including details on AdaGrad usage [9], multi-value features, and regularization can be found in [3]. Offset also includes an automatic hyper-parameter online tuning mechanism [4]. This mechanism takes advantage of the parallel map-reduce architecture and strives to tune Offset hyper-parameters (e.g., SGD initial step size, regularization parameter, AdaGrag parameters, etc.) to match the varying temporal and trending effects of the marketplace.

2.3 Serving

When a user arrives at a Yahoo O&O or Syndication site, and a Gemini native slot should be populated, an auction takes place. Initially, serving generates a list of eligible active native ads for the user along with the ads' scores. Roughly speaking, an ad's eligibility to a certain user in a certain context is determined by ad targeting which is beyond the scope of this paper.

Auction. The score is a measure that attempts to rank the ads according to their potential revenue with respect to the arriving user and her context (e.g., day, hour, site, device type, etc.). In general, an ad's score is defined as

$$\text{rankingScore}(u, a) = \text{bid}(a) \cdot \text{pCTR}(u, a) \,,$$

where pCTR is provided by an Offset model (see Eq. (1)), and $\text{bid}(a)$ is the price the advertiser is willing to pay for a click on ad a.

To encourage advertisers' truthfulness, the cost incurred on the winner of the auction is determined according to *generalized second price* (GSP) [10], which is defined as

$$\text{gsp} = \frac{\text{rankingScore}_2}{\text{rankingScore}_1} \cdot \text{bid}_1 = \frac{\text{pCTR}_2}{\text{pCTR}_1} \cdot \text{bid}_2 \,,$$

where indices 1 and 2 correspond to the winner of the auction and the runner up, respectively. Note that by definition $\text{gsp} \leq \text{bid}_1$, which means the winner will pay no more than her bid. In particular, if both ads have the same pCTR, the winner will pay the bid of the runner-up (i.e., bid_2).

2.4 Native Carousel Ads

Native ads are designed to resemble the surrounding content of the page and are considered less intrusive than other ad types. In general, Gemini native CPC ads have a fixed format which includes a title, image, and a short description (see Figure 1 for a typical Gemini native ad).

Recently, the native ad marketplace introduced a variant of the "traditional" native ads referred to as carousel ads. A Gemini native carousel ad includes several assets (which are native-like ads with a title, image and also a description in certain cases) and several slots to which the different assets may be rendered into (see Figure 2 for a schematic example of Gemini native carousel ad on mobile and desktop devices). The slots are not symmetrical and some are more conspicuous than others. In particular, on desktops there are 3 slots where the left one (slot 1) is much bigger than the other two and also has a short description. On mobile there may be 5 scrollable slots while only slot 1 is completely visible at first. While the two layouts are quite different, the advantage of slot 1 is apparent in both.

In case a carousel ad wins the auction and is about to be impressed, the available assets are mapped to the slots. Gemini native UI provides the advertiser two alternatives to map assets to slots: (a) allow an automatic mapping of assets to slots (default option in the UI); and (b) mapping assets to slots according to the advertiser's predefined ordering.

3 RELATED WORK

The problem of carousel asset optimization (CAO), or finding "good" assets-to-slots mappings in terms of CTR and revenue, can be formulated as a reinforcement learning problem, where agents take actions to maximize some notion of cumulative reward [19]. In particular, our problem is related to the rich literature of the multi-armed bandit (MAB) problem, which comes in several flavors, such as regret minimization [7], and best arm identification [11][5][14]. It also studied in many settings, including restless bandits [18], mortal bandits [8], distributed bandits [12], bandits with feedback delay, and bandits with non-stationary rewards [6]. We note that our setting is somewhat similar to that of [6] with assets CTRs varying over time. However, in contrast to [6] we assume that the assets still preserve their ranking in terms of CTR when presented in slot 1. Our problem is also closely related to the contextual bandit problem [16], and to Thompson sampling [2]. Other works that also adopt a holistic approach (as oppose to our post-auction approach) are [13] and [17]. Explore-exploit techniques are widely used to optimize web content. For example, [1] describes an explore-exploit based system for selecting news items for Yahoo Homepage. There are also many commercial tools that conduct various ad related optimizations on social platforms (see [15] and references therein).

As we shell demonstrate in the sequel, due to system requirements and constraints, our approach is based on a post auction approach, which separates ad ranking and carousel asset rendering (as oppose to works that adopt a holistic approach, e.g., [16][2][13] and [17]). In addition, since CTRs and impression rates are varying over time we need a practical and robust way to compare assets CTR measurements. This is why we chose a successive elimination based approach (inspired by [11][5]) that ensures all surviving assets have the same impression rates at any given time, and therefore their measured CTRs are comparable.

[8] User and user context features may include: age, gender, geo, section id, device type, yahoo category taxonomy (YCT), time, day, etc.

4 PROBLEM STATEMENT

Assume we have a carousel ad with m assets, $n \leq m$ slots, and a predefined advertiser asset ordering. Our goal is to use the additional degree-of-freedom of mapping assets to slots, to increase the carousel CTR and revenue when compared to the same carousel rendered using the advertiser asset ordering.

Next, we discuss the main constraints, requirements, and setting conditions that dramatically affect the selected approach.

4.1 Requirements, Constraints and Conditions

Since the CAO system should be incorporated into the existing Gemini native architecture, there are several system requirements and inherent constraints that considerably affect the nature of the selected approach. Starting with the inherent constraints: (a) *Delay* – the time it takes for the system to update the model, index it, serve users, and log the users' actions. In the current production environment the average delay is more than an hour; (b) *Incremental mode* – data processing is currently done in chunks of a few minutes worth of data. This means that approaches that require policy changes after each action can not be considered.

Turning to system requirements: (a) *Model size* – Offset model size cannot be considerably increased. Since most of the model consists of ad vectors, considering different combinations of assets-slots as new ads is impossible; (b) *Query-per-second (QPS) and SLA* – QPS cannot be considerably decreased (or alternatively SLA cannot be considerably increased). This means we cannot have a Vespa query per assets combination (e.g., using the slot number as an ad feature); (c) *Serving complexity* – computation complexity of the serving system should be limited to simple processing per ad (e.g., sigmoid function of the vectors' inner product); and (d) for legacy reasons the interface to the serving system is limited to asset distributions per carousel ad. The serving system then uses the distributions in a predefined manner to map assets to carousel slots.

Finally, we mention the setting conditions that challenges the algorithm selection and design: (a) *varying impression rate* – impression rate of each carousel is dictated mainly by its popularity, bid, and budget availability. Those factors determine the ad delivery rate (the rate it wins auctions and gets impressions) and may change over time; (b) *varying assets CTRs* – CTR in general is known to vary over time (day over night, week days over weekends, etc.). In addition, popularity trends also causes ad (and assets within ads) CTR variations. Measurements performed on assets CTRs (not presented here) enable us to relax the latter condition, and assume that while assets CTRs are varying over time, they still maintain the same ranking during the carousel ad lifespan[9]. Combining the two aforementioned setting conditions makes generating comparable asset CTRs measurements, a challenging task.

5 OUR APPROACH

5.1 Overview

To avoid the need to increase Offset model size or decrease the QPS of the serving system (see Sec. 7 for more details), we chose to separate the processes of ad auction and carousel asset optimization (CAO) and adopt a post auction approach. Hence, for each incoming impression we let the serving system use Offset model to select the winning ad and only then try to "optimize" the best asset for the current impression in case a carousel ad wins the auction. Recalling that the number of possible assets is rather small (e.g., 3-5), and since asset CTR changes over time (see Section 4.1), we decided to use a *successive elimination* based solution (see [11][5]) for the CAO. Since Gemini native system works incrementally (we can only update models every few minutes), the proposed solution is to update a distribution over the assets (based on assets' CTR measurements) per carousel for every training period. Then, to send the distributions (packed up in a carousel model file) periodically to the serving system, and allow it to draw the selected assets according to the relevant asset distribution before rendering the winning carousel ad.

Note that an asset distribution is used to convey the algorithm artifacts to the serving system. Although deterministic lists would suffice in this case, we are banned to use distributions instead, due to the dictated interface of the serving system (see Section 4.1).

5.2 Rendering Carousels in Serving time

Assume the auction winning ad is some active carousel ad C with m assets $\mathcal{A}_C = \{A_j\}_{j=1}^m$ and a respective asset distribution $P_C = \{p(A_j)\}_{j=1}^m$. In the following we describe how the serving system uses P_C to render the assets (or map assets to $n \leq m$ slots) in C.

To select the asset for slot 1, draw an asset according to P_C. Assuming $A_i \in \mathcal{A}_C$ is picked, update the remaining asset set $\mathcal{A}_C \longleftarrow \mathcal{A}_C \backslash A_i$ and the remaining asset distribution as a conditional asset distribution given that A_i was selected, $\forall A_j \in \mathcal{A}_C$; $p(A_j) \longleftarrow p(A_j)/(1 - p(A_i))$. And so on and so forth until all n slots are populated.

5.3 Carousel Assets Successive Elimination Algorithm (CASE)

In this section we focus on the CASE algorithm that is applied to each active carousel ad independently. As mentioned earlier, we adopt a *successive elimination* approach (e.g., see [11][5]) where we gradually eliminate assets with lower CTR than that of the "best" asset, and continue to explore the "surviving" assets with uniform distribution until only one remains.

We assume that while assets' CTRs are varying over time, they maintain the same ranking over the carousel entire lifespan (see Appendix A for a specific example). This assumption may not hold for certain cases where assets strike a temporal trend and asset ranking is changed. We may relax this assumption and assume that CTR ranking changes "slowly" over time. In this case we can retrain each carousel periodically to capture the new trend (planned for future work).

General description. For an arbitrary active carousel C we get its asset lists (usually 3-5 assets), initialize a uniform distribution over the assets P_C and put all assets in the "surviving" asset set $\mathcal{S}_C$. Then, during every training period (e.g., 15 minutes) we do the following: (a) update the number of impressions and clicks for each asset when it was presented in slot 1; (b) find the "best" asset in terms of slot 1 CTR of all "surviving" assets; (c) update the "surviving" asset set

[9] If a carousel ad has three assets with $CTR_1 > CTR_2 > CTR_3$, where CTR_i corresponds to asset i respectively, this ranking will remain throughout the carousel lifespan.

$\mathcal{S}_C$ by deleting all assets that have slot 1 CTR lower than that of the "best" asset with confidence (see Section 5.4); and (d) update the asset distribution P_C accordingly. We stop updating the assets' distribution and "surviving" set when the latter consists of a single asset. See Algorithm 1 for a formal description of the *carousel asset successive elimination* (CASE) algorithm.

Assets distribution. Assume a carousel with 3 assets A_1, A_2, A_3 and their slot 1 CTRs $CTR(A_1) > CTR(A_2) > CTR(A_3)$, respectively. Then if the CASE algorithm is working properly the final asset distribution should be $p(A_1) = 1 - \epsilon^2 - \epsilon^3$, $p(A_2) = \epsilon^2$, $p(A_3) = \epsilon^3$, and for $\epsilon = 0.01$, we have $p(A_1) = 0.999899$, $p(A_2) = 0.0001$, $p(A_3) = 0.000001$. We note that selecting this particular distribution is arbitrary and it is meant to impose the correct order with high probability. For example, if we follow the carousel rendering (or asset selection) procedure described in Section 5.2, we get that the probability for selecting A_1, A_2, A_3 to slots 1, 2, 3 respectively is $Pr(correct) = \frac{1-\epsilon^2-\epsilon^3}{1+\epsilon} \approx 0.9998$.

The following actions are omitted from Algorithm 1 for clarity and mentioned here instead.

- *Omitting inactive carousels* To reduce the carousel model size, we omit carousels that were inactive (i.e., had no traffic) for a predefined time period (e.g., a week) and treat it as a new carousel ad if it reappears.
- *Entering Idle state* In case a carousel is being explored for more than a predefined time period (e.g., 30 days) without having at least one asset resolved, it enters an *Idle* state, rendering assets according the advertiser predefined ordering.

5.4 When CTR Difference is Significant

We use a degenerated version of the one-tailed *Welch's t-test* (see [20][22]) to eliminate assets with predefined level of confidence. Since we deal with CTR comparison the Welch's t-test score reduces to $\mathcal{Z}_W(A_j)$ given in line 16 of Algorithm 1. Usually, this test requires also the degrees-of-freedom df associated with the setting, which in our case can be lower bounded by the minimum number of impressions of both, the best asset and the tested asset minus 1

$$df \geq \min\{N_{imprs}(A_b) - 1, N_{imprs}(A_j) - 1\} \ .$$

However, since in our setting at any stage $N_{imprs}(A_b), N_{imprs}(A_j) > N^{min}_{click} >> 1$, for any practical use we can assume that $df = \infty$ and that the *t-distribution* reduces to a Normal distribution [21]. Hence, the one-tailed Welch's t-test based elimination criterion used here, provides a confidence level larger than Pr_c (an input parameter of CASE – see Algorithm 1) in case

$$\mathcal{Z}_W(A_j) > \mathcal{Z} = Q^{-1}(1 - Pr_c) \ ,$$

where $Q^{-1}(\cdot)$ is the inverse Q function of standard Normal distribution.

5.5 Why CASE?

In previous sections we justified various design decisions in an ad hoc manner. Here, we would like to summarized those in one place for clarification matters.

- The two stage post-auction based approach is selected for reduced system complexity, and not increasing Offset model size or not reducing the serving QPS (see Section 4.1). This

Algorithm 1 Carousel asset successive elimination (CASE)

Input:
C - arbitrary active carousel ad
$\mathcal{A}_C = \{A_i\}_{i=1}^M$ - advertiser asset list for C
Pr_c - confidence probability (e.g., $Pr_c = 0.9$)
ϵ - assets' distribution parameter (e.g., $\epsilon = 0.01$)
N^{min}_{click} - minimum number of clicks (e.g., $N^{min}_{click} = 10$)
α - smoothing factor for CTR calculations (e.g., $\alpha = 30$)
Output (updated after each training period):
$\mathcal{S}_C \subset \mathcal{A}_C$ - surviving asset set
$P_C = \{p(A_i)\}_{i=1}^M$ - asset distribution

1: initialize the surviving set $\mathcal{S}_C = \mathcal{A}_C$
2: initialize P_C with uniform distribution
$p(A_i) = 1/M \ ; \ i = 1, \ldots, M$
3: initialize the assets' impressions and clicks counters
$N_{impr}(A_i) = 0, \ N_{click}(A_i) = 0 \ ; \ i = 1, \ldots, M$
4: calculate the confidence constant (see Section 5.4)
$\mathcal{Z} = Q^{-1}(\frac{1-Pr_c}{2})$
5: **for** each training period τ_t **do**
6: **for** each surviving asset $A_j \in \mathcal{S}_C$ **do**
7: extract the number of impressions and clicks occurred in slot 1 during τ_t
$N^c_{impr}(A_j), \ N^c_{click}(A_j)$
8: update the impressions and clicks counters
$N_{imprs}(A_j) \longleftarrow N_{imprs}(A_j) + N^c_{impr}(A_j)$
$N_{clicks}(A_j) \longleftarrow N_{clicks}(A_j) + N^c_{clicks}(A_j)$
9: calculate asset slot 1 CTR
$CTR(A_j) = \frac{N_{clicks}(A_j)}{N_{imprs}(A_j)+\alpha}$
10: **end for**
11: **if** $\exists A_j \in \mathcal{S}_C : \ s.t. \ N_{clicks}(A_j) < N^{min}_{click}$ **then**
12: **CONTINUE**
13: **end if**
14: find "best" survivor asset
$A_b = \underset{A_j \in \mathcal{S}_C}{\text{argmax}} \ CTR(A_j)$
15: **for** each surviving asset $A_j \in \mathcal{S}_C$ **do**
16: calculate asset A_j *Welch's t-test* score (see Section 5.4)
$\mathcal{Z}_W(A_j) = \frac{CTR(A_b)-CTR(A_j)}{\sqrt{\frac{CTR(A_b)(1-CTR(A_b))}{N_{imprs}(A_b)} + \frac{CTR(A_j)(1-CTR(A_j))}{N_{imprs}(A_j)}}}$
17: **if** $\mathcal{Z}_W(A_j) > \mathcal{Z}$ **then**
18: eliminate asset A_j from surviving set
$\mathcal{S}_C \longleftarrow \mathcal{S}_C \setminus A_j$
19: update asset A_j probability
$p(A_j) \longleftarrow \epsilon^{M-\|\mathcal{S}_C\|+1}$
20: update the surviving list assets' probabilities
$\forall A_k \in \mathcal{S}_C \ ; \ p(A_k) \longleftarrow \frac{\left(1-\sum_{i=M-\|\mathcal{S}_C\|+1}^{M} \epsilon^i\right)}{\|\mathcal{S}_C\|}$
21: **end if**
22: **end for**
23: **if** surviving set includes one asset $\|\mathcal{S}_C\| = 1$ **then**
24: **BREAK**
25: **end if**
26: **end for**

is since the asset selection is done after the auction and no query is needed to calculate the ordering. Instead, only the simple draw based ordering procedure done according to the carousel asset distribution is used.

- To combat the time varying asset CTRs and impression rate, we select a successive elimination based approach (inspired by [11][5]). Therefore, at any stage all surviving assets has the same probability and has the same chance to be presented in slot 1. This makes asset CTR measurements as described in Algorithm 1 comparable.
- As required (see Section 4.1), the only interface between the CAO system and the serving system is via asset carousel distributions. In addition, CASE works incrementally and can tolerate feedback delays.
- CASE is robust and practical. Since we are less sensitive to regrets (or revenue losses due to the time it takes for carousels to be resolved) and more focused on identifying the "best" asset in terms of slot 1 CTR, we wait until we gather N_{click}^{min} clicks before we start eliminating assets, we use a "smooth" CTR measurement using the α parameter (an input parameter of CASE – see Algorithm 1), and eliminate assets that have lower CTR compared to the "best" asset with predefined confidence level. We use a statistical test to eliminate assets since unlike some theoretical works we do not know in advance what is the expected asset CTR difference.
- Not knowing the assets CTR difference in advance is another reason for selecting a successive elimination based approach. Assume we have identical assets, thus, no matter how long we wait we would never be able to tell them apart. According to CASE we won't have to, since both assets will get the same probability causing no revenue regret although the carousel will never be actually resolved.

We end this section by saying that we do not claim to be optimal, but as we demonstrate in the sequel, we do manage to use the additional degree-of-freedom of automatic asset ordering to increase carousel ads CTRs and revenue.

6 PERFORMANCE EVALUATION

In this section we report the performance of the CASE algorithm based CAO system. We describe the buckets settings, provide bucket statistics, define the performance metrics and baseline, and present the results.

It is noted that since proprietary logged data is used for evaluating our model, it is obvious that reproducing the results by others is impossible. This caveat is common in papers describing commercial systems and we hope it doesn't undermine the overall contribution of this work. In addition, due to commercial confidentially matters we report lifts and rescaled CTRs only. However, this should not change the trends and observations we provide.

6.1 Carousel Traffic Statistics

A few months after deploying the CAO in production, we gathered statistics during a fortnight period. In the following we only consider traffic of carousel ads with asset optimization option **enabled** (i.e., CAO traffic).

During the aforementioned time period we encountered a few thousands unique carousels. On average the model includes 86.7% carousels in exploration state (or unresolved carousels), 10.9% in exploitation state (or resolved carousels), and 2.5% in idle state[10]. We note that while only 10.9% of the carousels are in exploitation state, their traffic consists of 92.2% of the total CAO carousel traffic. Out of the exploring carousels, 14.0% have at least one eliminated asset (partially resolved carousels). Out of the exploiting carousels, 62.5% have declared a "best" [11] asset different than the one denoted by the advertiser. Assuming our algorithm provides ground truth, advertisers identify their carousels' "best" assets in advance, only 37.5% of the time.

Next, we focus on traffic of carousels that were terminated during the fortnight aggregation period (a total of hundreds of millions of impressions). Out of the churning carousels, we measure an average lifespan of 33.3 days. The average exploration period for carousels that were finally resolved (entered exploitation state) is 7.5 days. The average exploration period to average lifespan ratio is 24.7%, which means that carousels that were resolved during their lifespan, spend on average 75.3% of their lifespan serving users with the "best" asset placed in slot 1. Interestingly, the ratio of the average number of exploration impressions to the average number of lifespan impressions is 17.5%, which is much lower than the exploration period to lifespan period ratio (24.7%). This may be explained (putting budget constraints aside) assuming the exploiting carousels get more impressions, due to their higher CTRs which is learned by Offset and translates to higher pCTRs and therefore more auction wins. Both, explore time and explore impressions ratios distributions are depicted in Fig. 3, where the shift towards shorter exploration in terms of impressions (orange curve) is clearly notable.

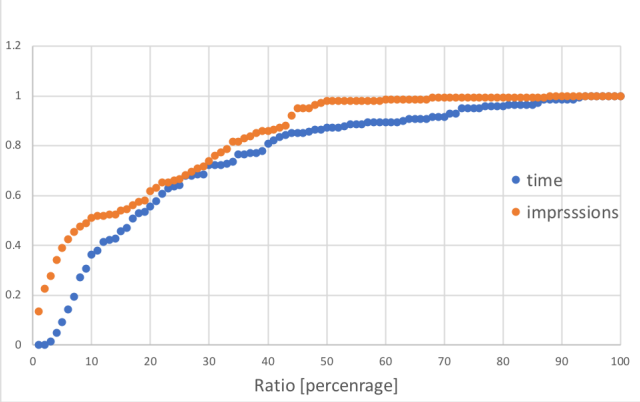


Figure 3: Carousel ad distributions of exploration period to lifespan ratio (blue curve), and exploration impressions to lifespan impressions ratio (orange curve).

6.2 Online Evaluation

6.2.1 Settings. After a phase of offline simulations, the CAO system was launched over a 0.7% traffic bucket in the *continuous deployment*

[10] Carousels with no eliminated assets after a predefined period, e.g., 30 days.
[11] The asset with highest slot 1 CTR.

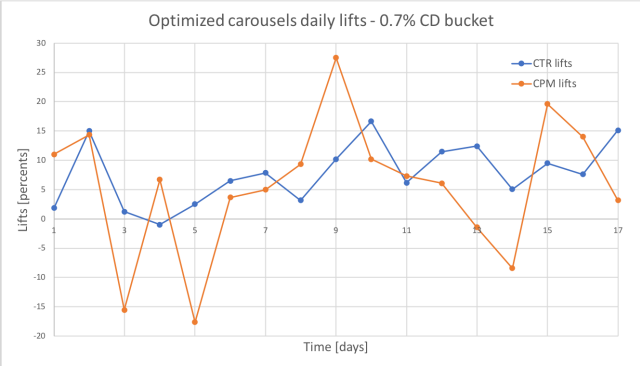


Figure 4: Carousel asset optimized (CAO) traffic CTR and CPM lifts in CD 0.7% online bucket (17 days).

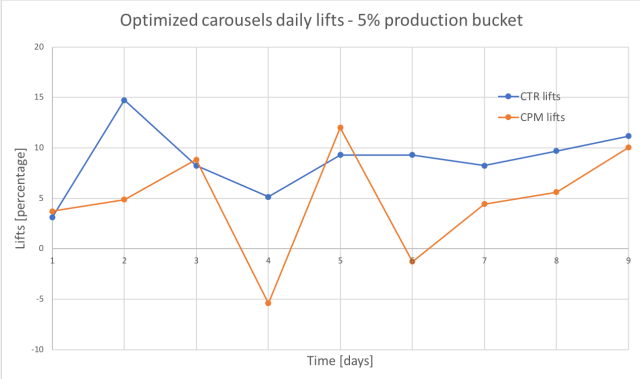


Figure 5: Carousel asset optimized (CAO) traffic CTR and CPM lifts in production 5.0% online bucket (9 days).

(CD) science testing environment. The online tests demonstrated the potential of the proposed solution and were also used for tuning the algorithm parameters. Finally, we set the confidence probability $Pr_c = 0.82$, the minimum number of clicks $N_{click}^{min} = 30$, CTR smoothing factor $\alpha = 5$, and asset distribution parameter $\epsilon = 0.01$. In addition, we set a carousel to Idle state if it has no resolved assets after a week of exploration and omit it from the model if it is inactive for a week. We chose a rather high number of minimal clicks to reduce the "noise" of CTR measurements. The aforementioned parameters set demonstrated the best CTR and revenue lifts we experienced in the limited grid search we performed during our online CD experiments.

After demonstrating good online performance (see Section 6.2.2) the CAO system was pushed to production in several incremental ramp-up steps.

Baseline. At any stage we compared the CAO buckets to a **control bucket** (in production or in CD) operating with no CAO (CASE algorithm is disabled), rendering carousels using their predefined advertisers' asset ordering for all carousels (also for those with CAO enabled).

During the testing and ramp-up phases we used the following metrics to evaluate performance.

Performance metrics. The performance metrics we use to evaluate our buckets are CTR and CPM (revenue per thousand impressions). The CTR and CPM are calculated for the entire carousel ad and not for the individual assets. Recall that the CAO system does not directly affect the Gemini native auction and it only affects the carousel CTR. Therefore, higher post-auction CTRs should translate to higher revenue (neglecting secondary factors such as budget constraints).

6.2.2 Results. The daily average CTR and CPM lifts [12] of the CAO test buckets traffic compared the that of the control buckets traffic are plotted for the 0.7% CD bucket and the 5% production bucket in Figures 4 and 5, respectively. Examining the figures we observe that the CAO system achieved its goal and consistently improved the CTR over the relevant traffic gaining positive CTR lifts on all days (except one day in the CD bucket). In addition, it is observed that the positive CTR lifts are translated to positive revenue gains on most days.

The average results of the two buckets are summarized in Table 1. In addition to the CAO traffic, we also report the overall bucket lifts. For the CAO traffic of the 5% production bucket we report an impressive 8.6% lift in CTR [13] and 4.3% lift in CPM. We note that in the current CAO daily spend, such a CPM improvement roughly translates to a few million USD increase in Gemini native yearly revenue. The table also reveals that the 5% production bucket performance on all traffic is flat positive. Since CAO traffic consists of a small portion of the overall traffic at the time, getting a flat positive performance lift is a good sanity check that the CAO system is not harmful. Nevertheless, it cannot be a good predictor of the CAO performance lift on the relevant small traffic portion.

It is worth mentioning that the results above are averaged over all traffic, not just resolved carousels in exploitation state, which includes many millions of impressions. However, performance lifts for individual resolved carousels (see Appendix A for one specific carousel) may be much higher.

6.3 Impact

In this section we report the impact of the new CAO product on the carousel market segment. This impact goes beyond the obvious revenue gain due to improved CTR reported in the previous section. To show that, we have monitored the carousel traffic for 9 days just before CAO launch and for 14 days five months after the launch. It appears that the carousel revenue (both CAO and non-CAO) increased by almost 40% during this five months period due to advertiser demand increase. Moreover, the entire carousel traffic increase is related to CAO traffic. This means that since launch, most advertisers chose their carousel assets to be optimize by CAO (set as default in advertiser UI) rather than use their own asset ordering. In addition, we also report that the daily average number of active CAO ads has more than quadrupled in this period. We cannot claim that the whole revenue growth is related to the CAO deployment only, but it certainly had a positive impact and definitely reflects advertiser satisfaction of the product.

[12] Since higher is better for both CPM and CTR, the lifts are calculated by $(CTR_{CASE}/CTR_{control} - 1) \cdot 100$ and $(CPM_{CASE}/CPM_{control} - 1) \cdot 100$.
[13] The measured CTR average values for the 5% production buckets are accurate up to ±1% of their true values with more than 95% confidence.

Description	Bucket period	Bucket size	CTR lift	CPM lift
CD bucket – CAO traffic	17 days	0.7%	7.7350%	4.3437%
CD bucket – all traffic	17 days	0.7%	-0.2229%	-0.1289%
Production bucket – CAO traffic	9 days	5.0%	8.6383%	4.3431%
Production bucket – all traffic	9 days	5.0%	0.0226%	0.0021%

Table 1: Carousel ads buckets results summary.

7 CONCLUDING REMARKS

In this work we presented a simple and practical two stage solution to the carousel asset optimization problem, or in other words an algorithm for identifying the "best" rendering of assets to slots in terms of asset CTR in the most conspicuous slot. During the first stage of serving, a regular Gemini native auction is performed and if a carousel ad wins the auction, the serving system uses the asset distribution generated by our CASE algorithm to map assets to slots. Since its deployment, after showing good performance in several online buckets, carousel revenue have increased by almost 40% over a period of five months, where the entire lift is related to additional CAO traffic.

We note that different approaches to CAO were considered as well. The simplest approach would have been to consider each combination of asset ordering as a new ad and add it to the inventory. This however, could increase in theory the ad inventory and Offset model size by a factor of the average number of asset combinations (e.g., 6 assuming 3 assets and 3 slots). Another approach considered is treating the different combinations as ad features and incorporating those into Offset. This approach would reduce the QPR of the serving system since we need to calculate pCTRs for each asset combination per carousel before conducting the auction, thereby increasing the SLA by a factor of the average number of combinations. The proposed solution, although requiring modification of the serving system and not claimed optimal, does not increase Offset model size and only slightly impacts the serving system SLA. Moreover, it is demonstrated to use the additional degree-of-freedom provided by the automatic asset ordering to increase both, carousel CTR and revenue.

Future work may include an improved CASE algorithm that considers clicks made on all slots and not just those of slot 1. This may expedite exploration and increase revenue. Another direction may be to perform CASE on smaller traffic segments composed of ad cross user features (e.g., ad×gender or ad×device). This may improve personalization but may require longer exploration periods due to sparsity issues.

REFERENCES

[1] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, Nitin Motgi, Seung-Taek Park, Raghu Ramakrishnan, Scott Roy, and Joe Zachariah. Online models for content optimization. In *Advances in Neural Information Processing Systems*, pages 17–24, 2009.

[2] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on Learning Theory*, pages 39–1, 2012.

[3] Michal Aharon, Natalie Aizenberg, Edward Bortnikov, Ronny Lempel, Roi Adadi, Tomer Benyamini, Liron Levin, Ran Roth, and Ohad Serfaty. Off-set: one-pass factorization of feature sets for online recommendation in persistent cold start settings. In *Proc. RecSys'2013*, pages 375–378, 2013.

[4] Michal Aharon, Amit Kagian, and Oren Somekh. Adaptive online hyperparameters tuning for ad event-prediction models. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 672–679. International World Wide Web Conferences Steering Committee, 2017.

[5] Jean-Yves Audibert and Sébastien Bubeck. Best arm identification in multi-armed bandits. In *COLT-23th Conference on Learning Theory-2010*, pages 13–p, 2010.

[6] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In *Advances in neural information processing systems*, pages 199–207, 2014.

[7] Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends® in Machine Learning*, 5(1):1–122, 2012.

[8] Deepayan Chakrabarti, Ravi Kumar, Filip Radlinski, and Eli Upfal. Mortal multi-armed bandits. In *Advances in neural information processing systems*, pages 273–280, 2009.

[9] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159, 2011.

[10] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *The American economic review*, 97(1):242–259, 2007.

[11] Eyal Even-Dar, Shie Mannor, and Yishay Mansour. Pac bounds for multi-armed bandit and markov decision processes. In *International Conference on Computational Learning Theory*, pages 255–270. Springer, 2002.

[12] Eshcar Hillel, Zohar S Karnin, Tomer Koren, Ronny Lempel, and Oren Somekh. Distributed exploration in multi-armed bandits. In *Advances in Neural Information Processing Systems*, pages 854–862, 2013.

[13] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

[14] Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pages 1238–1246, 2013.

[15] Ronny Lempel, Ronen Barenboim, Edward Bortnikov, Nadav Golbandi, Amit Kagian, Liran Katzir, Hayim Makabee, Scott Roy, and Oren Somekh. Hierarchical composable optimization of web pages. In *Proceedings of the 21st International Conference on World Wide Web*, pages 53–62. ACM, 2012.

[16] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[17] Aleksandrs Slivkins, Filip Radlinski, and Sreenivas Gollapudi. Ranked bandits in metric spaces: learning diverse rankings over large document collections. *Journal of Machine Learning Research*, 14(Feb):399–436, 2013.

[18] Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the brownian restless bandits. In *COLT*, pages 343–354, 2008.

[19] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

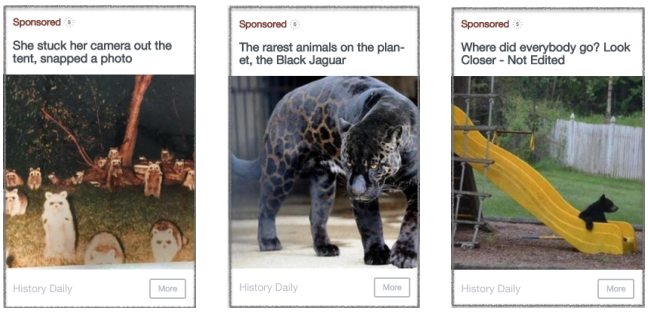
[20] Bernard L Welch. The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35, 1947.

[21] Wikipedia contributors. Student's t-distribution — Wikipedia, the free encyclopedia, 2019. [Online; accessed 4-February-2019].

[22] Wikipedia contributors. Welch's t-test — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Welch%27s_t-test&oldid=879631631, 2019. [Online; accessed 2-February-2019].

A THE TALE OF ONE CAROUSEL

The inspected carousel xxx374 was first seen on Jun. 30 last year, and had 308.5K impressions before becoming inactive on Jul. 5 (lifespan of 5.4 days). It had 3 assets that were ordered by the advertiser as follows: xxx23, xxx24, and xxx22. After exploring its assets for 0.8 days (14.8% of its lifespan) using 30K impressions (9.7% of its total impressions), the CASE algorithm identified that the correct ranking in terms of slot 1 CTR is actually xxx24, xx23, and xxx22.



((a))

((b))

Figure 7: (a) Carousel xxx374 assets (from left to right) xxx23, xxx24, and xxx22; (b) Carousel xxx374 desktop version rendered with advertiser ordering.

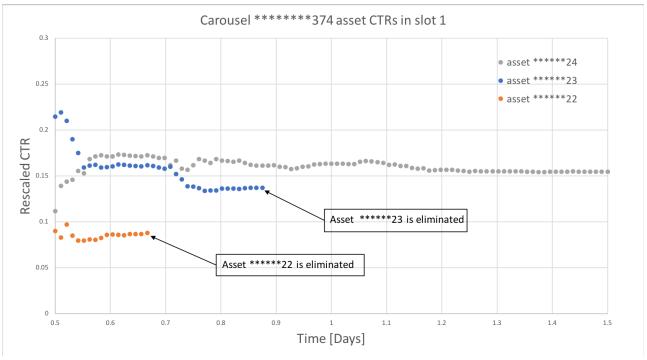


Figure 6: Carousel xxx374 life cycle slot 1 assets rescaled CTRs vs. time.

The assets' slot 1 normalized CTRs are plotted vs. time in Figure 6. The figure shows the period after all assets had at least 30 clicks and until activity turned sparse (i.e., 0.5 days to 1.5 days after first seen). It can be seen that after a short period of uncertainty, the ranking of the accumulated slot 1 CTR is established among the assets. Then, about 16 hours after first seen, asset xxx22 is eliminated and about 5 hours later, asset xxx23 is also eliminated, announcing asset xxx24 as the asset with highest CTR at slot 1. Unfortunately, we can not evaluate the performance lift of this carousel because CAO is deployed in 100% production traffic and we do not hold a control bucket running without CAO anymore.

Lastly, we present the assets of carousel xxx374 and its desktop version, rendered with the advertiser ordering in Figure 7. Recall that our users who were exposed to this carousel preferred the middle asset over the left one.