A New Optimization Layer for Real-Time Bidding Advertising Campaigns

Gianluca Micchi, Saeid Soheily-Khah, Jacob Turner
Skylads, 75008 Paris, France

Abstract
While it is relatively easy to start an online advertising campaign, obtaining a high Key Performance Indicator (KPI) can be challenging. A large body of work on this subject has already been performed and platforms known as DSPs are available on the market that deal with such an optimization. From the advertiser's point of view, each DSP is a different black box, with its pros and cons, that needs to be configured. In order to take advantage of the pros of every DSP, advertisers are well-advised to use a combination of them when setting up their campaigns. In this paper, we propose an algorithm for advertisers to add an optimization layer on top of DSPs. The algorithm we introduce, called skott, maximizes the chosen KPI by optimally configuring the DSPs and putting them in competition with each other. skott is a highly specialized iterative algorithm loosely based on gradient descent that is made up of three independent sub-routines, each dealing with a different problem: partitioning the budget, setting the desired average bid, and preventing under-delivery. In particular, one of the novelties of our approach lies in our taking the perspective of the advertisers rather than the DSPs. Synthetic market data is used to evaluate the efficiency of skott against other state-of-the-art approaches adapted from similar problems. The results illustrate the benefits of our proposals, which greatly outperforms the other methods.

Keywords: Demand Side Platform (DSP), Online Advertising, Gradient Descent, Optimization, Real Time Bidding (RTB)

1. Introduction
Online advertising is a vast market, worth several billion USD per year [13]. It is easy to understand the importance of optimization in such a market: every percent of increase in efficiency has a value in the order of millions of USD. From the advertiser’s point of view, however, optimization is often very difficult due to

*Corresponding author
Email address: research.paris@skylads.com (Gianluca Micchi, Saeid Soheily-Khah, Jacob Turner)
the constraints imposed by the structure of the ecosystem of online advertising. Let us see why.

There are currently two main paradigms in the market: sponsored search and Real-Time Bidding (RTB) auctions.

Sponsored search, the main source of revenues for search engines, consists in showing relevant ads whenever a user inputs a query. For example, typing “football shoes” in the search bar will provide the user with several links to online shops selling sports apparel. In order to appear amongst the sponsored links, an advertiser must place a bid on the queries or keywords to which it wants to be connected. Typically, advertisers pay if and only if their sponsored link is clicked on; this creates a collaborative effort between search engine and advertiser to show the most promising ad. Sponsored search has been the subject of lots of research papers over the years, dealing with, for example, budget optimization \[10, 14\] and click-through rate prediction \[24, 31\].

The RTB paradigm, which is the focus of our optimization work, is inherently different. In this paradigm, advertisers participate in auctions to buy the available space on a website. The winning advertiser is allowed to display its ad (in the technical jargon, it has bought an “impression”). Unlike sponsored search, advertisers pay for all impressions they buy, even if they don’t “generate a click”, i.e., users don’t click on them to be redirected to the advertiser’s page. This changes everything as it removes any interest for the auctioneer to find an ad that is a good match to the current inventory, a task that is now completely left to the advertisers. As a consequence of these differences, the optimization results obtained in sponsored search are not directly applicable to RTB campaigns.

As we already mentioned, this optimization process is made difficult by the very structure of the market that, in its simplest approximation, looks like this (cf. Fig. 1): The central entity is the AdExchange, whose job is to run real-time bidding auctions and assign every available inventory (i.e., the screen space on which the advertising should be published) to its corresponding winning advertiser; On one side of AdExchanges there are Supply Side Platforms (SSPs), which provide the inventory and are in direct contact with the publishers (e.g., the owner of a web page); On the other side, the one that interests us the most, there are Demand Side Platforms (DSPs): they bid on the available inventory on behalf of the advertisers according to the advertisers’ necessities. Each step of this chain brings its own constraints and optimizations: for example, a DSP might work only with certain AdExchanges, effectively limiting the amount of inventory available to the advertiser, but it also might offer better performances on some particular indicators due to internal optimization algorithms.

It is easy for an advertiser to set up a campaign with a DSP and monitor its effectiveness by calculating certain Key Performance Indicators (KPIs). On the flip side, however, the use of a DSP prevents the advertiser from directly determining the bid on each individual auction, only allowing it to fix some average parameters to associate with larger sets of impressions. We call each of these abstract entities made of a set of impressions and its corresponding parameters a media object. Each media object can almost be treated as a
A separate entity with its own associated budget that can change over time, the only global constraint being that the total budget of the campaign is fixed. A single media object makes the optimization easy to handle but inefficient because it treats all impressions in the same way, bidding roughly the same amount of money for all impressions and showing the same advertising to all users. A correct choice of the parameters of the media objects can, therefore, have a huge impact on the global optimization of the campaign.

But this setup leaves advertisers with many questions: What is the most appropriate DSP to use amongst the many available on the market? How to parametrize it? How much budget to assign to each of its media objects? These questions are often answered by human experts that base their decisions on past experience, intuition, and some off-line data analysis. However, there is no guarantee that the goals set by the experts are reachable, nor that they are optimal.

In recent decades, researchers have focused mainly on market models [6, 12, 18, 21, 29] and bidding algorithms for DSPs [5, 30, 7, 11, 19, 23]. However, to the best of our knowledge, no paper tackles the optimal management of a campaign from the point of view of an advertiser that uses a DSP. The new constraints that come with such a perspective make other optimization algorithms studied in the literature difficult to compare with ours. For example, our need to partition the budget arises from the necessity to work with media objects: when an impression arrives that is a good fit to a particular media object, we need to be sure that the media object has a sufficient budget to spend. This problem does not exist when one can decide how much money to spend on a single impression basis, with the only constraint of the total campaign budget.

The main contribution of this paper is the skott algorithm, that solves this understudied problem: skott automatically handles advertising campaigns, finding the best parameters to put inside each DSP in order to maximize the performance; it also allows the contemporary use of different DSPs that are put in competition to further increase the performance. Therefore, not only it gives a recipe to fully take advantage of any single DSP, but it also adds a new layer of optimization on top of them. The algorithm reacts quickly to market variations and scales linearly with respect to the number of media objects.

The remainder of this paper is organized as follows. In Section 2 we discuss a few algorithms that deal with a similar problem and have been an inspiration

![Figure 1: A simple, approximate representation of the RTB auctions market structure. The proposed algorithm, skott, would work as an interface between advertisers and DSPs.](image-url)
for our work. In Section 3 we state the problem at hand and our goals. Sections 4, 5, and 6 deal with the three independent sub-routines that compose SKOTT. The conducted experiments and results obtained are discussed in Section 7. Conclusions are drawn in Section 8. Finally, appendices give details on the creation of the synthetic market data, the algorithms we chose for comparison of the results, and the technical part of the implementation.

2. Related work

The problem of how to spend a budget in order to maximize the profit during an advertising campaign has never been studied, to the best of our knowledge, from the point of view of an advertiser using DSPs. Nevertheless, there are many works in the literature that are relevant because they try to solve a similar problem. We will cite and discuss some of them in this section, mainly to highlight what is the difference in our approach.

In [10] the authors propose to use a randomized uniform strategy for choosing how much to bid on every keyword in a sponsored search. That could be applied to our problem of Real-Time Bidding auctions through the use of DSPs problem by analogy, associating every keyword with a different media object. (We refer to the introduction and to the problem statement for the definition of media objects.) However, their model assumes a complete knowledge of the bidding landscape, that is, the probability distribution of the winning bids for each impression. This is information that advertisers don’t have in the case of RTB auctions through DSPs. Also, the model in [10] requires the bidding landscape to be static, a hypothesis that we don’t require.

The problem of collecting the largest possible reward of an advertising campaign with a constraint on the budget can be also written in the linear programming formalism. This is done, for example, in [5]. There, however, the authors: 1. take the point of view of a DSP, and 2. try to optimize the revenue of publishers and not of advertisers. In particular, they assign each piece of inventory to a different advertising campaign assuming that the total Cost Per Click (CPC) of each campaign is fixed. This last point is clearly not the case when we look at it from the advertiser’s side because, as we said, we consider the case where advertisers pay for the inventory they buy regardless of it being clicked on. This implies that the CPC is then determined by the ratio between the average cost per impression and the click-through ratio of a media object, therefore it’s not constant, as we will thoroughly see when we analyze the SKOTT algorithm.

A similar work, explicitly based on [5], is [19], where a linear programming algorithm is proposed for DSPs wanting to maximize their revenues. Besides the different perspective, the authors assume a fixed Click-Through Rate (CTR), known in advance. This is not required in our approach, where we infer the CTR from market data in real time and the only assumption we make on its analytical form is that it varies slowly with the bid. A linear programming approach inspired by these works will be tested against SKOTT in the simulations.
A third approach to the problem is due to reinforcement learning. In this case, the bidder is considered as an agent that learns how much to bid for every individual auction. To direct its choice, it is aware of the budget constraints, the goals of the campaign, and all context information from the impression it is trying to buy. This approach is studied for example in [4] where, due to the huge size of the space of possible actions to take, the authors help the decision process by using a model-based approach. This approach is extremely interesting but ineffective in our case for two main reasons: first of all, the constraints under which we work prevent us from accessing individual auctions; and, secondly, we obtain information not in real time but in batches that come in at larger time scales (roughly an hour). Nevertheless, the reinforcement learning approach is tested against SKOTT in the simulations, adapting it to allocate the budget to the different media objects according to their results.

3. Problem statement

As stated in the introduction, advertisers can participate in RTB auctions by using a DSP. Typically, several DSPs are employed in order to increase the amount of people reached and better respond to business necessities. Each DSP needs to be configured. The details of the configuration might differ from DSP to DSP, but there is a central core of abstraction that is common to all of them. We call it a media object: it is a set of instructions given by the advertisers, some of which are qualitative and set once and for all at instantiation while others are quantitative and can be changed at any moment. An example of the former is the creative associated to the media object, i.e., the actual advertising being shown. Another example of qualitative instructions are the filters on incoming auctions that select on which impressions to place a bid depending on the user and inventory characteristics. The hourly budget and the base bid for the auctions, instead, belong to the latter. It is important to note that the media object is the most precise layer of abstraction that is accessible to advertisers. The only influence that they can have on the auctions, and therefore their only possibility for optimization, lies in the parameters of the media objects.

We consider an advertising campaign to be defined by: a total budget $B$, a start date, an end date, a desired spend profile (i.e., the amount of total money spent at any moment during the campaign), and a collection of $K$ media objects spread across different DSPs. The typical duration for a campaign is of the order of a few weeks up to a few months, while the value of $K$ depends heavily on the campaign and can vary from as few as 1 to over 100,000.

During the campaign lifetime, the advertiser receives information on the behavior of every media object from the different DSPs. Each data point contains hourly information on the impressions bought, the clicks generated, the money spent, and possibly other such quantities. Since advertisers don’t have access to the auctions individually but only through the media objects, they should only consider the average effect of their optimizations over all impressions. Therefore, for each media object we take an hourly average of the information received. In practice, we consider only the Click-Through Rate (CTR, the ratio of clicks
generated to impressions bought) and Cost-Per-Click (CPC, the ratio of money spent to clicks generated).

The main goal of our algorithm is to change the media objects parameters in such a way as to optimize a certain KPI while keeping the desired delivery over time. In order to demonstrate a practical case, we have chosen to optimize the total number of clicks generated in the campaign. This is often a valid indicator to optimize because a user that is interested to purchase something from the advertiser’s website will probably click directly on the ads, while only a small fraction of the people that click on the ads will actually make a purchase. The click is then correlated to the monetary return of the campaign while not being as rare as an actual purchase.

4. **SKOTT: Budget partitioning**

SKOTT is an iterative algorithm made up of three subroutines: budget partitioning, that rewards high-quality media objects by giving them more money; base bid setting, that controls the bid of each media object separately with the goal of increasing the media object’s quality; and pacing control, that prevents under-delivery. Figure 2 provides a schematic view of the different steps of the algorithm. The three sub-routines act independently one from another and can therefore be analyzed in any order. In this section we deal with the budget partitioning that defines which percentage of the hourly budget should be allocated to each media object.

A budget partition is a vector of $K$ weights $w$. Each element $w_i$ represents which fraction of the total budget is assigned to the corresponding media object $i$. A uniform distribution, where all media objects are assigned the same budget, is represented by the vector $u = (k^{-1}, k^{-1}, ..., k^{-1})$. A greedy distribution is when a single media object takes all the available budget and is represented by the vector $w_g^{(i)} = (0, ..., 1, ..., 0)$.

The ideal algorithm for budget partitioning should:

- return a list of non-negative weights that sum to one at every decision epoch $t$,
- optimize a specific KPI (in our case the total number of clicks),
- promptly react to changes in the market, be they sudden or slow.

A very important point to consider when devising the algorithm is that the data must be bought through winning auctions. Reducing the budget of a media object well below the expected CPC will result in no clicks being bought, thereby gaining little to no useful information to estimate the quality of the media object. An advertiser may spend some time and money to explore the market randomly, then concentrate their money on the best performing media objects. This would probably lead, in average, to an increase in the return of the campaign; but the price to pay is a high risk to get stuck on a sub-optimal
4.1. The update rule

The algorithm we propose is a variation of the exponentiated gradient descent method originally proposed in [16]. At each iteration, it updates the weight vector $\mathbf{w}$ in order to optimize the KPI. The algorithm is resumed in Figure 3. It makes use of the concept of a reward assigned to each media object at every decision epoch. The reward is a numerical way to estimate how well the media object did in the epoch. We will see explicitly what it looks like later on.

Given a vector of weights at epoch $t$, $\mathbf{w}_t = [w_{1,t}, w_{2,t}, ..., w_{K,t}]$, describing the distribution of the budget allocated to each of the $K$ media objects during the $t$-th hour of the campaign, the algorithm will return a new vector of weights.
Algorithm for budget partitioning

1: **function** budget partitioning
   \[ t: \text{epoch index} \]
   \[ C_t: \text{clicks obtained by the media objects during } t \]
   \[ B_t: \text{budgets assigned to the media objects before } t \]
   \[ \hat{C}_{t-1}: \text{cumulative discounted clicks before } t \text{ (unless } t = 0) \]
   \[ \hat{B}_{t-1}: \text{cumulative discounted budgets before } t \text{ (unless } t = 0) \]
   \[ w_t: \text{previous repartition of the budget} \]
   \[ u: \text{uniform repartition of the budgets over the media objects} \]
   \[ \lambda: \text{regularization parameter} \]
   \[ \alpha: \text{learning rate} \]
   \[ \gamma: \text{discount factor} \]

2: **procedure** update the discounted quantities
   \[ \text{if } t = 0 \text{ then} \]
   \[ \hat{C}_0 \leftarrow C_0 \]
   \[ \hat{B}_0 \leftarrow B_0 \]
   \[ \text{else} \]
   \[ \hat{C}_t \leftarrow \gamma \hat{C}_{t-1} + C_t \]
   \[ \hat{B}_t \leftarrow \gamma \hat{B}_{t-1} + B_t \]

3: **procedure** calculate gradient of loss function
   \[ Q_t \leftarrow \hat{C}_t / \hat{B}_t \]
   \[ Q_t \leftarrow Q_t / \max(Q_t) \]
   \[ \nabla L_t \leftarrow -\hat{Q}_t + \lambda_t \cdot (w_t - u) \]

4: **procedure** update weights
   \[ \text{c} \leftarrow \exp(-\alpha \cdot \nabla L_t) \]
   \[ w_{t+1} \leftarrow w_t \cdot c / \sum_i w_{t+1} c_i \quad \triangleright \text{element-wise multiplication} \]
   \[ \text{return } w_{t+1} \]

---

Figure 3: Algorithm for budget partitioning

\[ w_{t+1} \text{ that is closer to the minimum of the loss function} \]
\[ L_t(w_t) = -\sum_i R_i(w_t) + \frac{\lambda}{2} \|w_t - u\|^2. \]  

Here, \( R_i \) is the reward associated to every media object, \( \lambda > 0 \) is the regularization parameter (that can depend on the epoch \( t \)), and \( u \) is the vector of uniform distribution with all entries equal to \( 1/K \) that we introduced before. The effect of the first term of the loss function is to favor the repartitions giving larger reward. The second term, known as the regularization term, requires the repartition \( w \) to be close to the uniform distribution \( u \). In other terms, it enforces the exploration of the market, with the consequences discussed in section 3. The relative importance of the exploration is therefore given by the numerical parameter \( \lambda \) that can be set at will. The easy interpretation of this parameter and its conceptual relevance is an important feature of our algorithm. We will discuss how to choose it at the end of this section.
The update rule defined by the exponentiated gradient descent is the following:

\[ \mathbf{w}_{t+1} = \frac{\mathbf{w}_t \cdot \exp (-\alpha * \nabla \mathcal{L}_t(\mathbf{w}_t))}{\sum_i [\mathbf{w}_{i,t} \cdot \exp (-\alpha * \nabla_i \mathcal{L}_t(\mathbf{w}_t))]}, \]

where \( \alpha \) is a real positive parameter known as the learning rate and \( \nabla \) indicates the gradient of a function with respect to the vector of weights \( \mathbf{w} \).

### 4.2. Explicit calculation of the derivative of the loss function

The rest of this section is devoted to write what is the value of \( \nabla \mathcal{L}_t(\mathbf{w}_t) \) that is needed to update the weights. To do so, we need to explicitly define what is the reward and to find its gradient. In general, the reward is given by the goal of the advertising campaign. As we already mentioned, we will use the maximization of the number of clicks as an example. The reward is then simply the number of clicks that a media object obtains during an epoch: \( \mathbf{R} = \mathbf{C} \). Its gradient represents the relative change in the number of clicks that a media object would have generated if we had given it a slightly different budget. This clearly can not be obtained directly from the market. Our solution is to model the relation between clicks and budget analytically, derive the gradient, and then approximate it using the sampled results from the market. In equations, this reads:

\[ R_{i,t}(\mathbf{w}_t) = Q_{i,t}(\mathbf{w}_{i,t}) \cdot w_{i,t}; \]
\[ \nabla \sum R_{i,t} = Q_t(\mathbf{w}_t), \]

where \( Q_t(\mathbf{w}_t) \) is the (unknown) vector of the coefficients that represents conceptually the quality of the media objects. Notice that to pass from Eq. 3 to Eq. 4 we have made the assumption that \( Q_t(\mathbf{w}_t) \) varies slowly with the weight vector so that its derivative becomes negligible. This is clearly an approximation, but a useful one whose price we happily pay.

Since \( \mathbf{w}_t = \mathbf{B}_t / \sum_i \mathbf{B}_{i,t} \), where \( \mathbf{B}_t \) is the vector of budgets associated to each media object at time \( t \), the quality vector can be written as:

\[ Q_t = \frac{\mathbf{R}_t}{\mathbf{w}_t} = \sum_i B_{i,t} \frac{C_t}{\mathbf{B}_t}. \]

Let us notice that \( \mathbf{Q} \) is quite similar to the vector of inverse CPCs, the two differences being the (unimportant) global positive multiplicative factor \( \sum_i \mathbf{B}_{i,t} \) and the presence at the denominator of \( \mathbf{B}_t \) instead of \( \mathbf{S}_t \) (the budget allocated instead of the money actually spent during the epoch). This is in accordance with our intuitive identification of \( \mathbf{Q} \) with the quality of a media object because lower CPCs are desirable.

Let us also mention that, since the quality of the media objects depends on external factors, a rescaling is needed to ensure the relative importance of...
the regularization parameter (hence the uselessness of the global multiplicative factor $\sum_i B_{i,t}$). We thus use the rescaled quality $\tilde{Q}_t$ defined as:

$$\tilde{Q}_t = \frac{Q_t}{\max(Q_t)},$$

which ensures all the elements of the vector to be positive and not larger than 1.

Under these conditions, we can rewrite the derivative of Equation 1 as:

$$\nabla L_t(w_t) = -\tilde{Q}_t + \lambda_t (w_t - u).$$

4.3. Fighting the noise

Due to the stochastic nature of the data coming from the market, there are a few corrections to make to the model of the quality vector in order to improve the precision and the stability of the results.

Let us consider the quality factor as defined in Eq. 5. The problem is that clicks are extremely rare: A typical CTR is 0.1%, meaning that only one impression out of a thousand generates a click. However, it is always possible, albeit rare, that a media object buys a small amount of impressions and generates a click. This is, of course, just sampling noise due to the very nature of the quantities we are dealing with. However, if not taken into account, it would dominate the response of the algorithm and lead to very unstable situations. Even worse, it could lock the algorithm to put all its money into a single, sub-optimal media object for a long time. To deal with that we make two corrections.

First of all, we put a hard bound on the gradient between $-10/\alpha$ and $+10/\alpha$, $\alpha$ being the learning rate of the gradient descent, to avoid exploding exponentials. This is very simple and straightforward, but it successfully prevents media objects with unusually large rewards to take all the budget.

Then, we claim that a better estimation of the value of the quality of a strategy can be done using a cumulative discounted version of the clicks and budgets, i.e., a variable that takes into account not only the latest data but also past data weighted by a discount factor $\gamma$:

$$Q_t = B_t \cdot \tilde{C}_t \Rightarrow \tilde{C}_t = C_t + \gamma \tilde{C}_{t-1}, \quad \tilde{B}_t = B_t + \gamma \tilde{B}_{t-1},$$

where we call $\tilde{C}$ the vector of cumulative discounted clicks initialized with the rule $\tilde{C}_0 = C_0$ (and similarly for the vector of cumulative discounted budgets $\tilde{B}$). Here, $\gamma \in [0,1]$ controls the importance of past data in the estimation of the quality of the media object. When $\gamma = 0$ we have no memory and $\tilde{C}_t = C_t$ (same for the budgets); we are back in the situation represented by (5). On the other hand, $\gamma = 1$ implies that the data collected at time $t_0$ is considered relevant for all $t > t_0$ and is never to be forgotten. This is desirable only when the quality is guaranteed to be constant. Since this is not the case, we use a $\gamma < 1$ to slowly forget data that is no longer relevant. We can fix the exact value of $\gamma$ by choosing a time scale for our campaign. If, for example, we want
to forget data that is a week old, we can put \( n_d = 7 \) and solve the equation \( \gamma^{n_d} = 1/e \) which yields \( \gamma = e^{-1/n_d} \approx 0.87 \). A first order approximation to this result can be obtained with \( \gamma \approx 1 - 1/n_d \approx 0.86 \).

4.4. The regularization parameter

We have said that an important feature of our algorithm is the relevance of the regularization parameter, that decides the trade-off between exploration and exploitation. Here, we explain what we chose in our simulations and why.

The regularization parameter that we used is

\[
\lambda_t = \eta \cdot K \cdot \gamma_{d(t)},
\]

where \( \eta \) is a positive number that determines the exploration-exploitation trade-off (in our simulations it is set to 1), \( K \) is the number of media objects, \( \gamma_r \) is (another) discount factor that determines when should exploitation dominate over exploration, and \( d(t) \) is the number of days that have passed since the beginning of the campaign.

The interest in rescaling with \( K \) comes from the advantage of keeping the product \( \lambda_t \cdot u \) that appears in the gradient of the loss function independent from the number of media objects. (Remember that \( u \) is the uniform distribution, whose elements are all \( k^{-1} \).) This grants a comparable greediness (measured for example as the KL-divergence from the uniform distribution, see Equation 28) when running on campaigns with vastly different number of media objects.

The presence of the term \( \gamma_{d(t)} \) stems instead from the advantage of having a larger exploration at the beginning of the campaign and a larger exploitation towards the end, where we want to monetize the knowledge we have acquired. The numerical value of \( \gamma_r \) is determined in the same way as for the discount factor in the quality vector, just using a different time-scale. If, for example, we want to keep a large exploration for 20 out of the 30 days of the advertising campaign, we would fix \( \gamma_r = 1 - 1/20 = 0.95 \).

5. Skott: Base bid setting

In this section, we present the algorithm that dynamically changes the base bid of a media object.

The base bid of a media object represents a sort of default value that is adjusted by the DSP depending on how valuable it deems a certain piece of inventory for said media objects. Many DSPs make this adjustment by multiplying the base bid by a score calculated from data about a specific item of inventory. Typically, however, the base bid will represent the average bid offered during the campaign.

Clearly, a high base bid will lead to chronic overbidding. This is indicated by the fact that the average cost per impression is significantly below the bid, assuming that the inventory is priced based on the second highest bid (as is overwhelmingly the case, cf. [28]). Overbidding is very risky because it might lead to a very large expense on non-valuable impressions if another player in
the market is making the same mistake. Conversely, a low base bid can cause the inability of a media object to buy inventory deemed valuable by the DSP, leading to an under-utilized budget.

Still following the example for which our goal is to maximize the number of clicks, we write \( L_t = -C_t \). Notice that we don’t put any regularization parameter because we don’t put any constraints on the base bids, so far. Now, we need to express the number of clicks as a function of the bids and then use gradient descent to maximize the function. In practice, the function that we use is:

\[
L_t(b_t) = -C_t(b_t) = -\frac{S_t(b_t)}{\text{CPC}_t(b_t)},
\]

where \( S_t(b_t) \), and \( \text{CPC}_t(b_t) \) are, respectively, the total amount of money spent and the resulting Cost Per Click in the previous epoch as a function of the base bid, each element of the vectors representing a different media object. In line of principle, the loss function should maximize the amount of money spent while decreasing the CPC of each inventory piece. These two objectives are contrasting: to spend more money one should increase the base bid to have access to more inventory, but to buy cheap clicks one should reduce the base bid.

In the following, we analyze separately the functions that appear on the right hand side of (10), starting with the CPC. For sake of simplicity, since all the bids act independently, we will just give the solution using scalar quantities. The full result is presented at the end of the section and is also resumed in Figure 4.

5.1. The analysis of the CPC

The CPC can be rewritten as:

\[
\text{CPC}_t = \frac{\text{CPM}_t}{1000 \text{CTR}_t},
\]

where \( \text{CPM}_t \) is the average Cost Per Mille, that is, the average price to pay for a thousand impressions, and \( \text{CTR}_t \) is the Click-Through Rate that we have already introduced.

First of all, we assume the CTR to be independent of the base bid and we estimate it from the market data as:

\[
\text{CTR}_t = \frac{C_t}{N_t},
\]

where \( N_t \) is the number of impressions bought and \( C_t \) are the clicks. The assumption of independence is justified by the fact that, if the media object filters are accurately set, all elements that are accessible by a single media object should be equally valuable. Also, a correlation between CTR and bid would mean that there is a general consensus on what is the most promising impression to buy no matter the campaign advertisers are running. The truth, as usual, lies in the middle: There is a certain correlation between CTR and
Algorithm for setting the base bid

1: function base bid setting

▷ N_t: number of impressions bought
▷ B_t: allocated budget
▷ S_t: money actually spent
▷ C_t: clicks obtained
▷ b_t: base bid
▷ β: median winning bid over all the inventory
▷ τ: under-delivery threshold
▷ α: learning rate
▷ u_b: upper bound on the bid
▷ l_b: lower bound on the bid
▷ B_min: minimal tolerance for zero budget

2: procedure calculate gradient of loss function

3: if B_t < B_min then
4:   ∇L(b_t) ← 0 ▷ If no budget is set, ignore the media object.
5:  else if S_t = 0 then
6:   ∇L(b_t) ← -1/α ▷ If no money is spent, increase the bid.
7:   else
8:     ∇L_t(b_t) ← -C_t \times N \times 1000 \times \frac{\beta \theta (\tau - \frac{S_t}{B_t}) - \frac{\beta}{l_b} \left[1 - \frac{\beta}{l_b} \ln \left(1 + \frac{l_b}{\beta} \right)\right]}{1 - \prod_{i=0}^{t} \mu_i}
9:     procedure apply Nadam to update the bids (see Figure 5)
10: procedure bound the bids in the correct interval
11:   b_{t+1} ← \max (l_b, b_t + 1)
12:   b_{t+1} ← \min (b_{t+1}, u_b)
13: return b_{t+1}

Figure 4: Algorithm for setting the base bid

Nadam algorithm

1: function Nadam

Require: α_t; µ_t, µ_{t+1}; ν_t; ε; ▷ Hyperparameters
Require: m_t; n_t; ▷ first/second moment vectors from the previous iteration
Require: b_t; g_t; ▷ bids and gradient of the loss function

2: m_{t+1} ← µ_t \times m_t + (1 - µ_t) \times g_t
3: n_{t+1} ← ν_t \times n_t + (1 - ν_t) \times g_t^2
4: \hat{m} ← \frac{\mu_{t+1} \times m_t}{1 - \prod_{i=0}^{t} \mu_i} + \frac{(1 - \mu_t) \times g_t}{1 - \prod_{i=0}^{t} \mu_i}
5: n_{t+1} ← ν_t \times n_t / (1 - ν_t)
6: b_{t+1} ← b_t - \frac{\alpha_t}{\sqrt{n_t} + \epsilon} \times m_t
7: return b_{t+1}

Figure 5: One iteration of Nadam, adapted from [8]
bid, but in absence of better methods to estimate it we neglect it. Introducing it in later improvements of the method will only require adding a term to the bid loss function.

We now have to find the CPM. Following [30], let us assume that the probability of winning an auction with a base bid of $b$ is given by an expression of the form:

$$P(b) = \frac{b}{b + \beta},$$

(13)

where $\beta$ is the median winning bid over all the inventory (since bidding $\beta$ gives a 50% probability of winning).

We can define a probability density function:

$$p(b) = \frac{dP(b)}{db} = \frac{\beta}{(b + \beta)^2},$$

(14)

that gives the percentage of inventory whose winning bid is exactly $b$.

RTB auctions often employ a second-price model to enforce truthful bidding [26, 20, 9]. In such a situation, and remembering that the bid is typically expressed in total offer per thousand impressions, the CPM is given by:

$$\text{CPM}_t = \frac{\int_0^{b_t} x p(x) \, dx}{\int_0^{b_t} p(x) \, dx} = \beta_t \left[ \left( 1 + \frac{\beta_t}{b_t} \right) \ln \left( 1 + \frac{b_t}{\beta_t} \right) - 1 \right].$$

(15)

We can notice the logarithmic increase of the average CPM at infinite bids representing competitors placing extremely high bids to acquire inventory, a strategy that gets rarer and rarer with increasing bids. From Equation 15 we can estimate the value of the parameter $\beta_t$ by comparing the estimated value of the CPM with the actual CPM returned from the market during that epoch.

The CPC as a function of all the basic quantities of the problem then reads:

$$\text{CPC}_t = \frac{N_t}{1000} C_t \cdot \beta_t \left[ \left( 1 + \frac{\beta_t}{b_t} \right) \ln \left( 1 + \frac{b_t}{\beta_t} \right) - 1 \right].$$

(16)

To obtain the derivative of the CPC part of the loss function we thus need only derive the CPM in 15. Calculations lead to:

$$\frac{d\text{CPM}_t}{db_t} = \frac{\beta_t}{b_t} \left[ 1 - \frac{\beta_t}{b_t} \ln \left( 1 + \frac{b_t}{\beta_t} \right) \right];$$

(17)

$$\frac{d\text{CPC}_t}{db_t} = \frac{N_t}{1000} C_t \frac{\beta_t}{b_t} \left[ 1 - \frac{\beta_t}{b_t} \ln \left( 1 + \frac{b_t}{\beta_t} \right) \right].$$

(18)

5.2. The analysis of the amount of money spent

In Equation 13 we have made an assumption about the probability of winning an auction based on the base bid $b_t$ that is well evidenced. We can try to leverage this assumption to find a relationship between $b_t$ and $S_t$. Let us divide our discussion in two parts: The case of under-delivery and the case of correct delivery.
In the case of under-delivery, a media object buys the entire inventory that is available to it (because if more was available, it would buy it with the remaining money). This quantity can be estimated with

\[ N_t(b_t) = I_{tot,t} \cdot P(b_t), \tag{19} \]

where \( I_{tot} \) is the total amount of inventory that would be available with an infinite bid. The total money spent is then given exactly by:

\[ S_t = \frac{I_{tot,t}}{1000} \int_0^{b_t} p(x) x \, dx \tag{20} \]

\[ = \frac{I_{tot,t}}{1000} \beta_t \left[ \ln \left( 1 + \frac{b_t}{\beta_t} \right) - \frac{b_t}{b_t + \beta_t} \right], \]

where the factor 1000 comes from the fact that the bid are expressed in offer per thousand impressions.

The derivative of (20) with respect to the bid is given by:

\[ \frac{dS_t}{db_t} = \frac{I_{tot,t}}{1000} \frac{b_t \cdot \beta_t}{(b_t + \beta_t)^2} = \frac{N_t \cdot \beta_t}{1000 b_t + \beta_t}. \tag{21} \]

We could have found the first equality also applying the fundamental theorem of calculus to (20), while the second equality comes from the substitution \( I_{tot,t} = N_t(b_t)/P(b_t) \) (see Equation 19) which does not depend on the base bid \( b_t \) because the dependences of \( N_t \) and \( P \) cancel each other.

In case of good delivery, instead, some pieces of inventory are not bought by the media object. A change in the base bid would most probably modify the number of such pieces of inventory but won’t change the total amount of money spent. Therefore, in this case, \( S_t \) is constant with respect to \( b_t \) and its derivative is 0.

In order to discriminate between the two delivery regimes, we use a Heaviside step function \( \theta \left( \tau - \frac{S_t}{B_t} \right) \), where \( \tau \) is an under-delivery threshold, typically set to 0.95 and not to 1 because a small amount of under-delivery is inherent to the discreteness of the problem.

5.3. Proposed loss function and gradient

We can now give the gradient with respect to the bids of the loss function \( \mathcal{L}_t \) proposed in (10). It reads:

\[ \nabla \mathcal{L}_t(b_t) = -C_t \left( \frac{1}{S_t} \frac{dS_t}{db_t} - \frac{1}{CPC_t} \frac{dCPC_t}{db_t} \right), \tag{22} \]

which, with the results found so far, becomes

\[ \nabla \mathcal{L}_t(b_t) = -C_t \cdot \frac{N_t}{1000 \cdot S_t} \times \]

\[ \times \left\{ \frac{\beta_t}{b_t + \beta_t} \theta \left( \tau - \frac{S_t}{B_t} \right) - \frac{\beta_t}{b_t} \left[ 1 - \frac{\beta_t}{b_t} \ln \left( 1 + \frac{b_t}{\beta_t} \right) \right] \right\}. \tag{23} \]
We notice that this equation is always well-defined, except when \( S_t = 0 \). This can happen in two situations: either there is no budget assigned to the strategy, in which case we impose no changes to be made since they would have no effect anyway; or there is a budget assigned but the strategy doesn’t manage to spend anything, in which case we are probably seriously underbidding and we fix the value of the gradient to be negative.

With this loss function, we perform a Nadam gradient descent \([8, 15]\) and then bound the result to be in between a minimal and a maximal bid set by the client. Unlike in budget partitioning, we choose an additive gradient descent because we don’t need any normalization.

As a last remark on the base bid setting, there is currently a resurgence in first price auctions. Our method is still applicable even in this situation, provided a few changes are made to the form of the equations. In particular, \((15)\) and \((20)\) would read respectively:

\[
\text{CPM}_t = b_t; \quad (24)
\]

\[
S_t = \frac{I_{\text{tot},t}}{1000} b_t \int_0^{b_t} p(x) \, dx, \quad (25)
\]

giving rise to different, but nevertheless well-defined update rules.

Recently, another research paper that deals with the bidding algorithm was published \([23]\). While there are similarities between their approach and ours, we chose to maximize directly the number of clicks instead of defining another utility function that needs other hyperparameters such as the monetary value of each click. Moreover, the method we propose in this paper does not need to have one data point per impression (an information that we assume is not at our disposal) but only the average over a certain period of time.

6. Skott: Pacing control

The third and last sub-routine is the one that controls the delivery ratio. It checks that the total amount of money spent in the campaign so far follows the desired profile. If that’s not the case, it increases the total budget available for the next epoch. Notice that our goal is not to determine what is the best delivery profile of the campaign over time, but only to stick to it as well as possible. This sub-routine is the simplest one since it only sets a single scalar parameter, unlike the previous two who sets a vectorial one.

Typically, advertisers want to control exactly how much money they spend during the campaign. For example, the simplest delivery profile is the uniform one, where the ideal amount of money spent until \( t \) is equal to the total budget of the campaign times the fraction of the campaign that has elapsed already. However, the money that was really spent on the market doesn’t always correspond to the desired amount: unforeseen technical issues, fluctuations in the available inventory, sudden changes in the properties of the market; they all can contribute to a variation in the amount of money spent, typically resulting in under-delivery.
Algorithm for pacing control

1: function PACING CONTROL
\triangleright t: the current epoch
\triangleright T − t: epochs left
\triangleright S_t: actual spent until t
\triangleright \bar{S}_t: ideal spent until t
\triangleright \bar{B}_{t+1}: ideal budget for the next epoch
\triangleright \eta: aggressiveness parameter
2: \Delta S_t \leftarrow \bar{S}_t - S_t
3: \Delta B_{t+1} \leftarrow \eta \cdot \Delta S_t / (T - t)
4: B_{t+1} \leftarrow \bar{B}_{t+1} + \Delta B_{t+1}
5: return B_{t+1}

Figure 6: Algorithm for pacing control

Before the budget partitioning and base bid setting sub-routines can react to the under-delivery and adapt their parameters, the actual delivery of the campaign will have lost ground to the ideal one. It is desirable then to take some measures in order to catch up with the ideal spent as soon as possible.

The hourly budget $B_{t+1}$ set by the algorithm looks like this:

$$B_{t+1} = \bar{B}_{t+1} + \Delta B_{t+1}, \quad (26)$$
$$\Delta B_{t+1} = \frac{\eta}{T - t} (\bar{S}_t - S_t), \quad (27)$$

where $\bar{B}_{t+1}$ is the ideal hourly budget, $\bar{S}_t$ and $S_t$ are respectively the ideal and actual amount of money spent until epoch $t$, $T - t$ is the number of epochs left, and $1 \leq \eta \leq T - t$ is the aggressiveness parameter. If the aggressiveness is set to 1, the algorithm tries to evenly spread the correction over the rest of the campaign. Surprisingly, this is not good: the reason is that a small amount of under-delivery is very common and it won’t be contrasted fast enough, imposing a money rush toward the end of the campaign. Also, we typically want to regain the ideal spend curve at a higher speed. However, too large a value of aggressiveness is not desirable either, because it could mean a very large sudden injection of money, possibly reducing the quality of our inventory and breaking the simple assumptions we had to make to construct a model. We typically choose values between 2 and 20.

A schematic view of the algorithm is presented in Figure 6.

7. Experimental results

We tested our algorithm on a simulated environment. (More on the characteristics of the market simulator in Appendix A) We will show the results as follows: First we will compare different budget partitioning algorithms while keeping no optimization on the base bids or on the pacing. Then, we will compare base bid setting algorithms on top of the budget partitioning we presented.
in this paper. Finally, we will show the advantages of introducing the pacing control algorithm on top of the budget partitioning and base bid setting algorithm we chose.

7.1. The comparison of budget partitioning algorithms

We compared our algorithm to three other algorithms: (1) A vanilla algorithm (codenamed vnl) that does absolutely nothing. (2) A multi-armed bandit algorithm inspired by [1, 3], codenamed mab. (3) A linear optimization algorithm inspired by [5], codenamed lop, that maximizes the clicks under the constraints of the total available budget and an interval of admitted budgets for every media object. More information about these algorithms can be found in Appendix D.

Figure 7 gives the comparisons of these algorithms. We present a simulation with day parting, i.e., with a different algorithm running for each hour of the day. The total number of epochs is 30, the number of days in a month. The first thing we want to point out is that lop is quite greedy, as we expected,
while \texttt{mab} is almost like \texttt{vnl}. This is due to the fact that only one media object per epoch is updated, giving just 30 small kicks to the initial situation. Our proposed algorithm, \texttt{skt1}, seems to strike in between: It moves quickly without becoming greedy.

To quantify this result, we measure greediness by calculating the KL-divergence \cite{17} of the proposed budget repartition with respect to the uniform distribution \(u\). The KL-divergence is a widely used method: for example, in reinforcement learning, it measures the distance between two policies, i.e., two different courses of action that optimize a given reward \cite{2, 25, 22}. It is defined as:

\[
\Delta(w, u) = \sum_i w_i \log \left( \frac{w_i}{u_i} \right).
\] (28)

A value of \(\Delta(w, u) = 0\) means that the distribution of the weights is exactly the same as \(u\). On the other hand, the maximal value is obtained by a greedy distribution \(w_g^{(i)}\) where one of the elements is 1 and all the others are 0. In this case, the KL-divergence measures \(\log(K)\). The values in the lower right plot of Figure 8 are values of KL-divergence rescaled by a factor \(\log(K)\), so that they are always constrained between 0 and 1 independently of the number of media objects.

These qualitative discussions find their quantitative conclusions in Figure 8 and in Table 1. The first column of numerical values (\texttt{spt}) represents the percentage of the initial budget that was spent, the second (\texttt{clk}) the additional...
Table 2: Optimization Results for Base Bid and Pacing.

| algo                  | spt | clk  | cpc  | kld  |
|-----------------------|-----|------|------|------|
| skt1                  | 96.2 % | 100.0 % | 0.785 | 0.095 |
| skt1 + pst            | 90.2 % | 136.9 % | 0.538 | 0.071 |
| skt1 + skt2           | 84.8 % | 231.0 % | 0.300 | 0.086 |
| skt1 + skt2 + skt3    | 99.8 % | 251.6 % | 0.324 | 0.085 |

clicks in percent with respect to the \textit{vnl} algorithm, the third (\textit{cpc}) the total CPC of the campaign, and the fourth (\textit{kld}) the distance of the budget repartition from the uniform distribution.

If one considers only the total number of assigned clicks as the metric to measure the performance of the algorithms, \textit{lop} wins over \textit{skt1} by a small margin. Also, its total calculated CPC is slightly lower, meaning that every click costs less money. However, on the bottom right panel of Figure 8 we can see how greedy \textit{lop} is. This reflects on the total amount of money spent on the top left panel: if the desired media object doesn’t have enough available inventory, this algorithm can not react decisively. Nothing grants us that this situation won’t happen in real life with even more damaging results, in particular, a severe under-delivery of the budget. On the contrary, \textit{skt1} manages to spend almost all the available budget (represented by the purple line) even without an explicit optimization on the bid and the pacing. The increased adaptability makes it more resistant to the real market test and thus more valuable.

7.2. The rest of the analysis

We now choose the \textit{skt1} algorithm for the budget partitioning and study the effect of different base bid setters. This time, we compare the new \textit{skt2} algorithm to other two algorithms: again \textit{vnl} where bids are not changed, and also \textit{pst}, an algorithm that uses a predetermined set of rules. We also add in the analysis the comparison of \textit{skt2} on top of \textit{skt1} with the full \textit{skott} algorithm, which also contains the pacing control sub-routine.

Let us explain first a little bit about \textit{pst}. The predetermined set of rules analyzes the CPC first: if it is higher than a certain goal set at the beginning of the campaign, the bid is reduced by a certain fixed multiplicative constant unless the media object is under-delivering. In that case \textit{pst} will still try to slightly increase the bid. We can see many inconveniences in this approach compared to the algorithm we presented in Section 5: first of all we need an additional external parameter, that is, the goal CPC. Furthermore, it makes very little use of the data from the market, reducing the adaptability.

These limitations have an effect on the results, as can be seen from Figure 10 and from Table 2. Differently from before, the baseline for the column (\textit{clk}) is now \textit{skt1} and not \textit{vnl}. The \textit{skt2} algorithm manages to outperform both \textit{vnl} and \textit{pst} by a vast amount, obtaining a larger number of clicks while spending less money.
Figure 9: The evolution of the base bid for 10 media objects, according to the four different algorithms we test, taken over a single run of the optimization. The solid lines show a smoothed version of the dotted lines for better visualization.

Figure 10: Same as Figure 9 but for the base bid setting algorithms. The solid lines show a smoothed version of the dotted lines for better visualization.
From the top left corner of Figure 10, one can see that the amount of money spent oscillates quite a bit at the beginning of our proposed algorithm. We see this stems from a similar oscillation in the algorithm's attempts to find the appropriate bids, as can be seen from Figure 9: the peaks of money spent correspond to bids slightly higher than the optimal and vice versa. Finally, we see that adding the third and last part of the algorithm manages to spend almost the entire initial budget, obtaining a small increase in clicks at the expense of a slightly higher CPC.

It is interesting to notice that the last plot, containing the KL-divergence, shows that the budget repartition changes slightly depending on which bidding algorithm we choose. This is understandable because, by changing the base bid, we actually modify the perceived quality of the media object and thus generate different inputs for the different iterations of the budget partitioning algorithm. However, these modifications are small enough to be neglected at first order.

8. Conclusion

We have introduced a method for advertisers to optimize the management of Demand Side Platforms when running an advertising campaign composed of many separate media objects. The method, that we call SKOTT algorithm, is an iterative method that makes only a few general assumptions on the mathematical model of the market. We present it here applied to a campaign for the optimization of the number of generated clicks.

The SKOTT algorithm is composed of three complementary parts: Firstly, the best partitioning for the budget across all media objects is calculated. This is achieved by estimating the quality of each media object and trying to obtain the maximum number of clicks through an exponentiated gradient descent method. Second, the best base bid for each media object is calculated. Here we use the assumption and corresponding evidence from [30] that relates the bid to the probability of winning the corresponding auction. We expand on this assumption to propose a model relating variation in bids to variation in the number of clicks obtained by each media object. We finally apply a Nadam technique [8, 15] on the market data to find the best base bids for maximizing the number of clicks. The third and last part determines the amount of budget to use at every epoch in order to stay as close as possible to the desired spend profile.

The proposed algorithm has been tested on a simulated environment that we created for the occasion and that we present in the appendices. Under these circumstances, the proposed algorithm gives impressive results, more than doubling the total amount of obtained clicks in the considered experiments.

Appendix A. Model of the market

We created a back-test platform for analyzing different campaign management algorithms. The work-flow of the platform is conceptually divided in five steps:
1. Parameters are chosen that describe the problem we have at hand.
2. Data is created using these parameters.
3. Loss functions are chosen.
4. All algorithms are launched independently. They work on the same data and their goal is to maximize the total number obtained during the campaign.
5. The results of the different algorithms are compared: we plot budget repartition, base bids, greediness of the algorithms, cumulative CPC, spend profiles, and collected clicks over time.

In this section, we will discuss the first two steps, that relate to the creation of the back-test platform itself. Step 3 and 4 are discussed in Section 4, 5, and 6 of the main body of the article, while some plots of the results are presented in section 7.

Appendix A.1. Choice of the parameters

There are just a few important parameters for the problem. They can be chosen independently, but the quality of the output policy heavily depends on their interaction. These parameters are:

- The total number of epochs $T$. This is fixed by the length of the campaign and by the duration the data batches. Since we consider hourly data and a campaign lasts for one month, we typically work with $T = 24 \times 30 = 720$.
- The day parting boolean: If true, we only consider data that comes from the same hour of the day, effectively launching 24 parallel algorithms and reducing the number of epochs per algorithm by a factor of 24. The reasons behind this are further explored in Appendix B.
- The number of media objects $K$. Typically, a larger $K$ increases the probability of having good media objects but also increases the cost of exploration. The optimal number of media objects, therefore, depends on the budget at our disposal.
- The CPC goal, a business-driven parameter that says how much money the advertiser wants to spend on average to get a click. We use this only for the pst algorithm.
- The total campaign budget $B$ or, equivalently, the average budget per epoch per media object $bem = B/(T \cdot K)$. In particular, there is a sort of phase transition at the critical value $bem = CPC_{\text{goal}}$: Above that value, all media objects should statistically be able to obtain at least one click per epoch, making for efficient exploration even in the first epoch alone, when no optimization has started yet. Below that, clicks become rare and many epochs are needed to efficiently understand the relative quality of the media objects.
• The number of repetitions of the experiment, $E$. The process of determining the best algorithm heavily depends on the input data, which is randomly generated. A single sample from a random generation might have some characteristics that are more convenient for a particular algorithm. Taking several samples of data and averaging the results of the numerical calculations over all the samples reduces the impact of this error at the cost of a linear increase in computational time. Typically we take $E = 20$.

Appendix A.2. Data creation

Data creation is handled by a Python class. Given the budgets $B_t$ and the base bids $b_t$ of all media objects for epoch $t$, it returns the impressions bought $N_t$, the clicks obtained $C_t$, and the money spent $S_t$ in that epoch for all media objects. To make synthetic data closer to real data, the class is constructed to mimic the structure of the market data we can access. Therefore, it takes as an input a subset of the same parameters that we send to the market during real-life campaigns. Also the output is made of a subset of the same results we obtain from the market in real life.

The underlying model of the market is exactly the same as described in section 5. For the creation of data, we generate the CTRs, the total inventory available on the market $I_{tot}$, and the median winning bids $\beta$ separately for each media object, sampling from appropriate intervals uniformly at random. Then we use (15) to calculate the average CPM of the media objects as a function of their bids, $b_t$. With all this, we can find out how many impressions each media object will buy at epoch $t$ by using:

$$N_t = \min \left( \frac{1000 B_t}{CPM_t}, I_{tot,t} \cdot P(b_t) \right),$$  \hspace{1cm} (A.1)$$

where the first element is the number of impressions one would buy with infinite available inventory, and the second one is the total available pieces of inventory with bid $b_t$.

We finally use the equation:

$$S_t = N_t \cdot \frac{CPM_t}{1000}$$  \hspace{1cm} (A.2)$$

to find out how much money each the media object spent and, consequently, how much it under-delivered.

Once the purchase of impressions is done, the market simulator assigns the clicks by sampling from a binomial distribution with probability of positive outcome given by the CTR of each media object.

The market simulator allows for a certain amount of control on the variation over time of the quality of the media objects, since all the quantities we generate can be modified at any epoch. This is a very important feature of the data: a model with static media objects tends to select algorithms that don’t explore, while dynamic media objects require exploration to keep track of the changing
environment. Different time dependencies, therefore, lead to different best algorithms. More details about sources of periodicity can be found in Appendix B.

Appendix B. Dealing with time variations

As already mentioned in Appendix A, there can be variations in the quality of media objects with time. While the causes are various, the main effect is the double periodicity induced by the day-night cycle and the weekly cycle [27]. In particular, there is a big drop in the volume of impressions dealt and the number of clicks at night that often leads advertisers to forgo buying impressions at these times.

But variations are not always periodic, nor globally affecting all the media objects at the same time. A change in the relative quality of the media objects can happen over time due to external factors. A typical example could be a media object advertising a live event: the distance in time from the day of the event is an important parameter for users to decide whether to buy a ticket.

Finally, some changes might be due to correlations not considered in our model. For example, a change in the base bid that we offer during auctions might lead to a modification of the CTR of the impressions we are able to buy.

The solution we take in order to deal with such issues is to launch 24 different algorithms, one for every hour. The advantage is clear: in case some media objects are turned off at a certain moment of the day, they won’t affect the perceived quality of the media object during other hours. However, there are obvious disadvantages as well: we discard data that might still give valuable information and convergence is 24 times slower.

For the aperiodic modifications over time, we want to have fast responses to the changes in the market. Since information on the changes is obtained through the purchase of impressions, we try to keep the algorithm as far from greedy as possible while still increasing the number of obtained clicks.

Appendix C. Data pre-processing

In the real world, when we receive the data, there are often missing values that need to be filled before starting the optimization. Here, we fill the missing values using a combination of three different approaches: a) backward filling: where we propagate the next valid observation, b) linear interpolation method: which is a method of curve fitting using linear polynomials to construct the missing values, and c) weighted moving average approach: which is an averaging that has multiplying factors to give different weights to data points at different positions.

Let \( \mathbf{x} = [x_1, x_2, \text{NAN}, x_4, ..., x_t, \text{NAN}, ..., \text{NAN}] \) be of assumed length \( \tau \). As one can see, there are some \text{NANs} before epoch \( t \), and a series of \text{NANs} between \( t \) and \( \tau \). In the case of having some missing values at the beginning of the vector, we fill the \text{NANs} using the backward filling method. For instance, \([\text{NAN}, x_i, x_j]\) gives \([x_i, x_i, x_j]\).
Next, we fill the nans up to epoch $t$ using a linear interpolate method. As a result, there will be no missing values between the epochs 1 and $t$.

Last, we fill the nans between the epoch $t$ and $\tau$ using a weighted moving averaging approach. Let us consider $x' = [x_1, ..., x_t, \text{NAN}, ..., \text{NAN}]$, where for all $1 \leq i \leq t$, $x_i \neq \text{NAN}$, and for all $t + 1 \leq i \leq \tau$, $x_i = \text{NAN}$. To do so, we consider a weight vector in the filling process. In an $t$-index weighted moving average, the latest data point has weight $t$, the second latest $t-1$, ..., terminating at one. Therefore, the estimation of data point $t + 1$ is defined as:

$$x_{t+1} = \frac{(1 \cdot x_1 + 2 \cdot x_2 + \ldots + (t-1) \cdot x_{t-1} + t \cdot x_t)}{t \cdot (t+1)/2} \quad (C.1)$$

All the missing values $x_i$ for $t + 1 \leq i \leq \tau$ are filled using (C.1).

Appendix D. A quick glance at the competing algorithms

In section 7 we mention three algorithms that we use as a comparison against our own. The \texttt{vnl} algorithm needs no explanation, because it corresponds to a non-optimized campaign in which the initial parameters are kept constant. On the other hand, \texttt{mab} and \texttt{lop} deserve a few words, which we will spend in the next paragraphs.

Appendix D.1. The multi-armed bandit algorithm

The Multi-Armed Bandit problem (MAB) deals with an agent (that is, whoever or whatever is able to take an action, such as a person or a robot) which, at definite time moments $t$ known as epochs, is faced with a number $K$ of possible different actions, each leading to a different reward. The goal is typically to find the course of actions, also called a policy, that maximizes the received reward. The difficulty of the problem is that the rewards are not known in advance and can be stochastic; a certain number of trials is necessary in order to explore the system before understanding what is the optimal policy to exploit.

The exploration-exploitation dichotomy is at the heart of all multi-armed bandit algorithms: the information acquired during the trials comes at the cost of exploring unknown actions that might lead to poor immediate rewards.

Stochasticity in a policy assures the exploration: therefore, it is needed at the beginning and is often reduced over time to ensure eventual convergence to the best action. This is no longer valid when the rewards change over time: In that case, the exploration is always needed to keep track of the moving averages.

Mathematically speaking, the MAB is defined by a tuple $\langle A, R \rangle$, where $A$ is the set of all possible actions the agent can take and $R$ is the set of the rewards associated with those actions. The policy $\pi$ is a probability distribution over $A$; the next action is chosen by sampling over the policy $\pi$.

In our case, we want to choose the best media object to bid with on incoming inventory items. The $K$ actions inside $A$ correspond to “choose media object 1”, ..., “choose media object $K$”. However, the agent that chooses the action to perform is a bidding algorithm that is not directly under our control. The
only thing we can influence is the distribution of the results: If, for example, we want to enact a policy for which media object 1 should be preferred 80\% of times (exploitation) and all the others should be equally distributed among the remaining 20\% (exploration), we can set the budget associated to every policy accordingly. Since a media object can not place a bid unless it has enough available budget, this effectively forces the desired average behavior over a certain time period.

A second difference with the standard MAB problem relates to the information we have access to. We don’t have access to all the information given to the bidding algorithm, but only to aggregate information on winning impressions every hour, therefore, to the average reward of every media object at every epoch. However, this is strictly related to our inability to access the bid itself and doesn’t introduce any additional constraint.

In spite of these differences, we can still create an algorithm based on the multi-armed bandit problem. To do so, we follow the algorithm \texttt{exp3} as presented in [1], copied in Figure D.11 for convenience.

\begin{figure}[h]
\centering
\begin{algorithm}
\begin{footnotesize}
\begin{algorithmic}
\Function{Exp3 budget partitioning}{\p
gamma: real number $\in (0, 1]$}
\Procedure{Initialization}{\p} \vspace{1em}
\State $w_i(1) = 1$ for $i = 1, \ldots, K$
\EndProcedure
\Procedure{Budget repartitioning}{\p} \vspace{1em}
\For{$t = 1, 2, \ldots$}
\State $p_i(t) \leftarrow (1 - \gamma) \frac{w_i(t)}{\sum_{j=1}^{K} w_j(t)} + \frac{\gamma}{K} \quad i = 1, \ldots, K$
\EndFor
\State draw $i_t$ randomly accordingly to the probabilities $p_1(t), \ldots, p_K(t)$.
\State receive reward $R_{i_t}(t) \in [0, 1]$.
\For{$j = 1, \ldots, J$} \vspace{1em}
\State $\hat{x}_j(t) \leftarrow \begin{cases} R_{i_t}(t)/p_j(t) & \text{if } j = i_t, \\
0 & \text{otherwise} \end{cases}$ \hfill (D.1)
\State $w_j(t + 1) \leftarrow w_j(t) \exp\left(\frac{\gamma \hat{x}_j(t)}{K}\right)$ \hfill (D.2)
\EndFor
\EndProcedure
\end{algorithmic}
\end{footnotesize}
\caption{The \texttt{exp3} algorithm as presented in [1].}
\end{algorithm}
\end{figure}

The reward function we choose is:

$$R = \frac{|x|}{1 + |x|}, \quad x = \frac{C}{C_{\text{goal}}} * \frac{C_{\text{goal}}}{\text{CPC}}, \quad (D.3)$$

where $C$ is a vector of clicks and the goal (in terms of clicks) is given by the maximum of the running exponential average of the number of clicks obtained by the media object and the ratio between the allotted budget and the goal.
CPC. This reward has been chosen because we value media objects that have 
low cost per goal and attract many clicks. We don’t use the variable \( x \) as a 
reward because we prefer to work with rewards bounded between 0 and 1.

Notice that this algorithm, while certainly offering improvements over the 
vn1 algorithm, is only using a fraction of all the information that we have since, 
conforming to the possibility of a real MAB problem, it uses the reward of a 
single media object at every epoch.

Appendix D.2. The linear programming algorithm

The problem of collecting the largest possible number of clicks can be written 
as an optimization under some linear constraints. In particular, we try to 
maximize the number of clicks under the (external) constraint of the total bud-
get and a (self-imposed) constraint on the amount of variation of the budget 
per each epoch.

![Linear programming algorithm for budget partitioning](Figure D.12: Linear programming algorithm for budget partitioning)

Let’s assign to the variables \( B \in R^K \) the budgets of the \( K \) media objects. 
The function to maximize is:

\[
f(x) = c^T \cdot B, \tag{D.4}
\]

where \( c = 1/\text{CPC} \) is a vector of coefficients that relates the budget to the 
expected number of clicks. The vector \( \text{CPC} \) is estimated from real market data 
with the formulas:

\[
\hat{S}_t = S_t + \gamma \hat{S}_{t-1}, \quad \hat{C}_t = C_t + \gamma \hat{C}_{t-1}, \tag{D.5}
\]

\[
\text{CPC}_t = \frac{\hat{S}_t}{\hat{C}_t}. \tag{D.6}
\]
where \( \mathbf{S}_t \) and \( \mathbf{C}_t \) are respectively the vector of the total money spent and the total clicks gathered at a certain epoch \( t \). The values \( \hat{\mathbf{S}} \) and \( \hat{\mathbf{C}} \) are initialized at zero when \( t = 0 \) and \( \gamma \in [0, 1] \) is the discount factor (see Section 4 for more details on the discount factor).

The constraints can be written in the form:

\[
A \mathbf{B} \leq \mathbf{v}, \quad A \in R(2k + 1, k), \quad \mathbf{v} \in R(2k + 1).
\]

The matrix \( A \) has only ones in the first row followed by two \( K \times K \) diagonal matrices containing respectively only \(-1\) and only \(1\). The vector \( \mathbf{v} \), instead, contains the remaining budget \( \tilde{\mathcal{B}}_t \) as the first element followed by the \( 2K \) bounds to the budget to be assigned to each media object: The first \( K \) are the lower bounds \( l_{i,t} \) (with a minus sign in front to reverse the direction of the inequality) while the second \( K \) are the upper bounds \( u_{i,t} \). For example, in the case \( K = 3 \), they read:

\[
A = \begin{pmatrix}
1 & 1 & 1 \\
-1 & 0 & 0 \\
0 & -1 & 0 \\
0 & 0 & -1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{pmatrix}, \quad \mathbf{v} = \begin{pmatrix}
\tilde{\mathcal{B}}_t \\
-l_{0,t} \\
-l_{1,t} \\
-l_{2,t} \\
u_{0,t} \\
u_{1,t} \\
u_{2,t}
\end{pmatrix}
\]

The values of the lower and upper bounds control are set as following:

\[
l_{i,t+1} = \alpha_l \cdot \mathbf{S}_{i,t}, \quad u_{i,t+1} = \alpha_u \cdot \mathbf{S}_{i,t},
\]

where \( \alpha_l < 1 \) and \( \alpha_u > 1 \). These two quantities control how much variation is possible to introduce between epochs and are therefore related to the learning rate of the algorithm.

The solution to this system can be found analytically and is presented in Figure D.12. It is important to notice that, in case the estimation of the CPC is sufficiently constant, this algorithm converges exponentially to the greedy solution. However, in our back-test simulations, stochasticity in the market and the use of the discount factor \( \gamma > 0 \) imply a fluctuation in the relative order of preference for the media object based on their CPC, which in turn leads to a behavior that is - at least partly - exploratory.

References

[1] P Auer, N Cesa-Bianchi, Y Freund, and Shapire R.E. The Non-Stochastic Multi-Armed Bandit Problem. SIAM J.Comput, 32(1)(1):48–77, 2002.

[2] Peter Auer and Ronald Ortner. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. Periodica Mathematica Hungarica, 61(1):55–65, 2010.
[3] Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic Multi-Armed-Bandit Problem with Non-stationary Rewards. *Advances in Neural Information Processing Systems (NIPS)*, pages 199–207, 2014.

[4] Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo. Real-Time Bidding by Reinforcement Learning in Display Advertising. *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining - WSDM '17*, pages 661–670, 2017.

[5] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '11*, page 1307, New York, New York, USA, 2011. ACM Press.

[6] Margy P. Conchar, Melvin R. Crask, and George M. Zinkhan. Market valuation models of the effect of advertising and promotional spending: A review and meta-analysis. *Journal of the Academy of Marketing Science*, 33(4):445, Sep 2005.

[7] Brian Donnellan, Markus Helfert, Jim Kenneally, Debra Vandermeer, Marcus Rothenberger, and Robert Winter. New horizons in design science: broadening the research agenda: 10th international conference, DESRIST 2015 Dublin, Ireland, May 20-22, 2015 proceedings. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9073:19–38, 2015.

[8] Timothy Dozat. Incorporating Nesterov Momentum into Adam. *ICLR Workshop*, (1):2013–2016, 2016.

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

[10] Jon Feldman, S. Muthukrishnan, Martin Pal, and Cliff Stein. Budget Optimization in Search-Based Advertising Auctions. In *Proceedings of the 8th ACM conference on Electronic commerce - EC ’07*, page 40, New York, New York, USA, 2006. ACM Press.

[11] Paul Grigas, Alfonso Lobos, Zheng Wen, and Kuang-chih Lee. Profit Maximization for Online Advertising Demand-Side Platforms. 2017.

[12] Maxim Gusev, Dimitri Kroujiline, and Boris Govorkov. Sell the news? A news-driven model of the stock market. *Academia*, pages 1–65, 2014.

[13] IAB. IAB internet advertising revenue report. Technical report, Interactive Advertising Bureau (IAB); PwC, 2017.
[14] Chinmay Karande, Aranyak Mehta, and Ramakrishnan Srikant. Optimizing budget constrained spend in search advertising. In Proceedings of the sixth ACM international conference on Web search and data mining - WSDM ’13, page 697, New York, New York, USA, 2013. ACM Press.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. ICLR, pages 1–15, 2014.

[16] Jyrki Kivinen and M.K Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Information and Computation, 132(1163):1–63, 1997.

[17] S. Kullback and R. A. Leibler. On Information and Sufficiency. The Annals of Mathematical Statistics, 22(1):79–86, mar 1951.

[18] Peter S H Leeflang, Dick R Wittink, and New Haven. Building models for marketing decisions: Past, present and future. International Journal of Research in Marketing, 17:0–43, 2000.

[19] Hualu Liu and Hao Wang. Dual Based DSP Bidding Strategy and its Application. (July), 2017.

[20] Roger B Myerson. Optimal auction design. Mathematics of operations research, 6(1):58–73, 1981.

[21] Raoul Pietersz and Antoon Pelsser. A comparison of single factor Markov-functional and multi factor market models. Review of Derivatives Research, 13(3):245–272, 2010.

[22] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz Openai. Parameter Space Noise for Exploration. arXiv, 2017.

[23] Kan Ren, Weinan Zhang, Ke Chang, Yifei Rong, Yong Yu, and Jun Wang. Bidding Machine: Learning to Bid for Directly Optimizing Profits in Display Advertising. IEEE Transactions on Knowledge and Data Engineering, 30(4):645–659, apr 2018.

[24] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks. In Proceedings of the 16th international conference on World Wide Web - WWW ’07, page 521, 2007.

[25] John Schulman, Filip Wolski, and Prafulla Dhariwal. Proximal Policy Optimization Algorithms Background : Policy Optimization. pages 1–10.

[26] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. The Journal of finance, 16(1):8–37, 1961.
[27] Shuai Yuan, Jun Wang, and Xiaoxue Zhao. Real-time bidding for online advertising. *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*, pages 1–8, 2013.

[28] Yong Yuan, Feiye Wang, Juanjuan Li, and Rui Qin. A survey on real time bidding advertising. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 418–423. IEEE, 2014.

[29] Chao. Zhang and Lu. Huang. A quantum model for the stock market. *Physica A: Statistical Mechanics and its Applications, 389(24), 389(24):5769–5775, 2010.*

[30] W Zhang, S Yuan, and J Wang. Optimal real-time bidding for display advertising. *Proceedings of the 20th ACM SIGKDD*, pages 1077–1086, 2014.

[31] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks. *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1369–1375, apr 2014.