Master Économiste d'entreprise

Pacing algorithm in DOOH

Displayce
M. Soueidan

Thomas Maurice

Université de Tours
M. Favard

3 septembre 2020

université de TOURS
Faculté de droit, économie & sciences sociales

Thanks

I would like to thank Mr. Favard for having supervised this internship as well as the MÉcEn teachers for their availability and their help.
I would like to thank Hayssam for taking me on this internship and helping me to improve throughout it.
Finally, I would like to thank the Displayce team for their welcome, in particular JS and Fabien for their precious help.

Executive Summary

Today billions of ad impressions are purchased through open real-time auctions. The world of digital out of home is opening up more and more to real-time impressions purchasing. The budget constraint provided by advertisers can be difficult to integrate by programmatic purchasing platforms such as Displayce. In addition, advertisers prefer that the budget expenditure to be smooth throughout the campaign. This is where the need for an efficient pacing algorithm comes in. Since the number of impressions received at any given time is not predictable, it is very challenging to achieve a uniform and total budget spending target at the same time. This paper addresses this issue by providing a methodological approach to build the algorithm with the constraints and difficulties encountered at each step. The final proposed algorithm allows deciding autonomously which impressions should be purchased in order to efficiently guarantee the pacing objectives and best satisfy the advertisers' constraints.

Table of contents

1 Introduction 4
- 1.1 What is Displayce? 4
- 1.2 How does DOOH works? 4
  - 1.2.1 Booking and RTB 4
  - 1.2.2 Digital poster operation 5
- 1.3 What is a bid request? 6

2 Auction theory 7
- 2.1 Types of standard auctions 7
- 2.2 Revenue Equivalence Theorem analysis 8
  - 2.2.1 Second-price auction 8
  - 2.2.2 First-price auction 10
- 2.3 Entry cost in Vickrey's auction 12

3 Temporal pacing 14
- 3.1 Simulation 14
- 3.2 Algorithm evolution 15
  - 3.2.1 Hourly capping 16
  - 3.2.2 Per second budget 17
  - 3.2.3 Evolutive hourly budget 21

4 Spatial pacing 25
- 4.1 Pacing meta class 25
- 4.2 Application of the meta-class 26
- 4.3 Budget reallocation between time zones 27

5 Conclusion 29

Appendix 31

References 31

1 Introduction

1.1 What is Displayce?

DISPLAYCE is the first DSP[1] in France and the leading technological platform designed to optimize ad campaigns run on digital posters and billboards in the field of the DOOH which stands for *Digital Out Of Home*. An ad campaign's quality[2] can be optimized thanks to the programmatic purchase[3] which falls within the scope of the open real time bidding. It allows to spend an ad budget efficiently over time and space. Hence, DISPLAYCE is an intermediary between supply and demand, where on the one side, the supply is a publisher who owns digital posters, and on the other side, the demand is an advertiser.

Founded in 2014 by Laure Malergue, DISPLAYCE is the leading DSP in France with access in more than 38 000 digital posters within different points of interest such as shopping malls, car parks, tobacco shops and avenues. Being the main intermediary, the platform then is of interest for both supply and demand. It allows publishers to reach multiple potential advertisers (supply side) and media agencies to benefit from a centralization of a major part of the offer (demand side). In general, it eliminates all the tedious negotiation and booking steps formerly effective. The interest in positioning itself as a DSP specializing in digital posters lies in the fact that the latter are becoming increasingly attractive. Indeed, their dynamic aspect, contrary to the old outdoor displays, is a force of attraction for consumers and the cost of setting up an advertisement display is practically nil.

1.2 How does DOOH works?

DISPLAYCE's programmatic purchasing platform is divided into two main parts: the booking part and the RTB part. The main activity is located on the Booking part but the RTB part is evolving.

1.2.1 Booking and RTB

What is called the booking part is actually the part that concerns the reservations of digital panels. The DSP dedicated to booking is connected to 30 000 screens spread exclusively in France. The purchase is made on the screen. When an advertiser wishes to set up its campaign on the platform, it provides information on the location of its campaign. He can then provide information such as the type of targeted panels (by type we mean the different screen sizes, ranging from small TVs in tobacco bars to large panels in shopping centres for example), points of interest

1. Demand Side Platform: a technological solution that allows you to automate the purchase of advertising inventory.
2. The quality of an ad campaign is measured thanks to different metrics such as the number of people targeted for example
3. The automated purchase of online advertising inventory

(schools, cinemas etc) or he can directly specify the precise locations (such a shop in such a city) and these locations are accompanied by the possibility of setting a targeting radius (20 km around schools for example). In addition to this location information, the advertiser specifies the broadcast dates of its campaign and from that point, DISPLAYCE will automatically reserve the billboards corresponding to the advertiser's criteria.

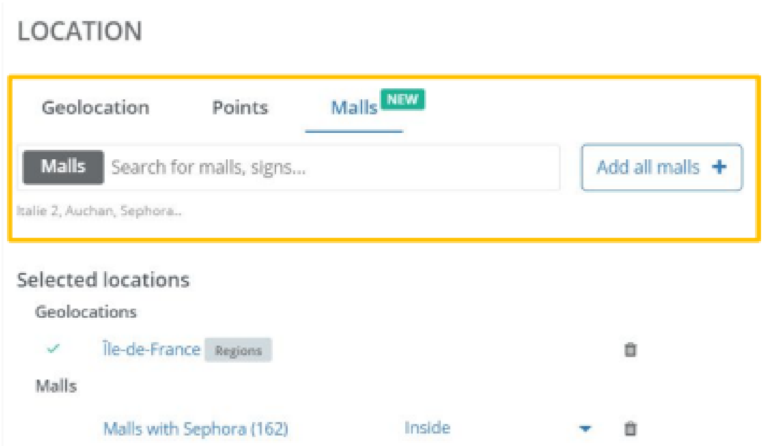

Figure 1 – Screen-shot of DISPLAYCE's platform

The other possibility for an advertiser is to use the RTB part. The major difference is that this time the purchase is not made on screen but on impression, i.e. the number of people who see the advertisement (the number of people in front of the panel). The DSP is connected to 53,000 panels around the world. The advertiser can specify the locations but in a less precise way than the booking part and so when a campaign is created, DISPLAYCE connects to the panels concerned and receives bid requests from these panels.

1.2.2 Digital poster operation

We are no going to discuss how does the connection to a digital panel technically works by taking the case of a unique digital panel. It performs a one-minute loop with 10 seconds of broadcast per ad, so there are 6 ads that run in a loop on this panel where each ad is submitted by a DSP such as DISPLAYCE. The peculiarity is that out of these 6 ads, 5 come from so-called "acquired" campaigns and one comes from a real-time auction. Acquired campaigns are generally reserved for the week and therefore this means that during a week on the same billboard every minute the same 10-second ad comes back. The last ad, the one that comes from a real-time auction, is therefore variable, meaning that every minute, the last 10 seconds for example, it will not be the same ad as the previous minute.

Let's now detail how the real-time auction works. To be able to participate in an auction, we must already be connected to the panel in question in order to have previously submitted the right advertising format to be displayed. Let's suppose that there are 4 DSP connected to a panel with 4 different ads. Just before the 10 seconds of the real-time auction ad broadcast, the panel issues a bid request to the 4 DSP. When a DSP receive a bid request, a price depending on the number of impressions [4] is associated to the bid request, and so it just have to answer 'yes we would like to buy at this given price' or 'no we would not'.

1.3 What is a bid request?

Let's explain what bid requests and real-time bidding consists of. The RTB (Real Time Bidding) is a server-to-server buying process that allows inventory (ad space) to be bought and sold on a per-impression basis (where an impression is a person who sees the specific ad). It happens instantaneously through an auction that determines who gets to buy a specific impression. It happens programmatically in the same way as financial markets do. If a bid is won, the advertisers' ad is immediately shown on the digital poster in the case of digital out of home.

A bid request provides information such as the exact location of the digital poster concerned, the exact time (to the nearest millisecond), the number of people in front of the poster and sometimes even the details of these people, namely their age group and gender. In the end, this information makes it possible to buy an audience rather than a panel location, and this is what allows optimization. The real time bidding system therefore makes it possible to develop algorithms for intermediaries such as DISPLAYCE in order to optimize the budget expenditure, reach the largest audience, smooth the budget in time and space for example, etc.

My internship at DISPLAYCE concerns the pacing part of the budget expenditure in order to ensure that the budget of a campaign is, on the one hand completely exhausted, and on the other hand spent uniformly over the day.

First, we will introduce the theoretical concept of auctions and the revenue equivalence theorem. This theoretical part will allow a better understanding of the functioning of an auction because the open real time bidding is based on this principle. We will then see the process of elaboration of the pacing algorithm on the time dimension, step by step. Finally, in a last point, we will discuss how to integrate an additional dimension in the pacing algorithm, namely the spatial dimension.

4. An impression is a person looking at a panel: if in T there are 4 people in front of the panel then the bid request will provide a number of impressions equal to 4.

2 Auction theory

2.1 Types of standard auctions

Auctions are transactions with a specific set of rules detailing resource allocation according to participants' bids (amounts of money they are willing to pay). In game theory, auctions are categorized as games with incomplete information because in the vast majority, one player will possess information that other players don't.
Standard auctions require that the winner of the auction be the participant with the highest bid. There are traditionally four types of auction that are used for the allocation of a single item:

1. *Ascending-bid auction* where the price is successively raised until only one bidder remains and that bidder wins the item at the final price.

2. *Descending-bid auction* works in exactly the opposite way. The auctioneer starts at a very high price, and then lowers the price continuously. The first bidder who agrees with the current price wins the item at that price.

3. *First-price sealed-bid auction* where each bidder independently submits a single bid, without seeing others' bids, and the object is sold to the bidder who makes the highest bid. The winner pays its bid.

4. *Second-price sealed-bid auction* works exactly the same way as the first-price sealed-bid auction except that the price the winner pays is the second-highest bidder's bid. This type of auction is also called *Vickrey's auction*.

This four types can be shortened to two types. Descending-bid auction and First-price sealed-bid auctions are based on the same principles. Each bidder must choose a price to call out, conditional on no other bidder having yet called out; and the bidder who chooses the highest price wins the item at the price called out. In both case, the winner pays the bid he/she called out. The Ascending-bid auction and the Second-price sealed-bid rest on the same principles as well but a little more reflection is needed. In an Ascending-bid auction, it is clearly a dominant strategy to stay in the bidding until the price reaches your maximum valuation of the good, that is, until you are just indifferent between winning and not winning. The next-to-last person will drop out when his/her maximum valuation of the good is reached, so the person with the highest bid will win at a price equal to the bid of the second-highest bidder. In a Second-price sealed-bid auction, a Nash equilibrium strategy is to bid the maximal valuation of the good you have, because if the worst comes to the worst you would pay your maximal valuation of the good minus ε so in every case you will have a positive payoff so there is no incentive to deviate from this strategy. Here again, the person with the highest bid will win at a price equal to the bid of the second-highest bidder. Now that we can distinguish two types of standard auction from the bidder's point of view, we should ask which is the most profitable type? The answer is in the *Revenue Equivalence Theorem* (Vickrey, 1961) which states that for certain economic environments, the expected revenue and bidder profits for a broad class of auctions will be the same provided that bidders use equilibrium strategies.

2.2 Revenue Equivalence Theorem analysis

Theorem 1 *For any two Bayesian-Nash incentive compatible mechanisms, if the surplus function is the same in both mechanisms, the valuation of each player is drawn from the same continuous distribution and each player bid their optimal strategy then the expected payments of all types are the same in both mechanisms, and hence the expected revenue (sum of payments) is the same in both mechanisms.*

Let's clarify some important terms. The maximal valuation that a consumer has for a good is the maximal price he is willing to pay for this good. The consumer's surplus is the difference between the valuation that a consumer have of a good and the price he pays for this good. Namely, it is the gain of a consumer after a purchase. Finally, the optimal strategy is the bid that a player should call in order to maximize his surplus. We call it as a Nash equilibrium when no deviation is worth it.

This part will bring a proof of the revenue equivalence theorem. To do so we will consider two standard auction mechanisms: second-price and first-price.
Let's assume an auction with two risk-neutral bidders and one seller. The seller sells a single item. Each bidder i has a valuation v_i of the good, v_i is drawn independently in a uniform distribution $[0, 1]$. We study standard auction so the bidder with the highest bid wins. Bidder i calls out a bid $b_i(v_i)$ that is increasing in v_i so that the bidder with the highest valuation of the good wins. We recall that the revenue equivalence theorem states that if there are 2 bidders with values drawn from $U[0, 1]$ then for any standard auction the winner bidder with valuation v will pay in average $\frac{1}{3}$ and will have an expected surplus $\frac{1}{2}v^2$. More generally, if there are N bidders with valuation from a continuous distribution, then any standard auction leads to the same expected highest bid and the same expected bidder surplus.

2.2.1 Second-price auction

To evolve the proof of the revenue equivalence theorem let's study the case in the second-price auction. We will first clarify the payment rule for this auction, then we will show that the optimal strategy for each player is to bid their maximal valuation of the good noticed v_i for player i. Next step will be to find the expected v_i assuming that $v_i > v_j$ is $\frac{2}{3}$ in our framework. Finally we will deduce that the expected bid of player i is $b_i^* = \frac{1}{2}v_i = \frac{1}{3}$ and the expected surplus is given by $S(v_i)^* = \frac{1}{2}v_i^2 = \frac{2}{9}$.

The payment rule of this auction is the following:

- if $b_i < b_j$, bidder i pays 0

- if $b_i > b_j$, bidder i pays b_j

The Nash equilibrium for a Vickrey auction is that all players bid their maximal valuation for the good namely: $b_i = v_i$. It is a dominant strategy because if $v_i > v_j$ then player i has a payoff equal to $v_i - v_j > 0$ because i would win the auction and pays the second price namely v_j. Player j's payoff is 0. There is no incentive to deviate for each player, indeed, if player i deviates, he loses surplus. If player j wants to change is payoff of 0, then he has to bid more than v_i in which case the payoff would be $v_j - v_i < 0$ so neither player i nor player j have an incentive to deviate.

The equilibrium strategies for both players are:
$b_i^* = v_i$
$b_j^* = v_j$
Now let's assume that $v_i > v_j$ so that player i wins. The next step is to find to what v_i and v_j are equal. To do so we have to find the expected value of the highest draw in a uniform law $U[0,1]$.

Consider N independent draws from a uniform distribution over $[0,1]$. On average, what is the highest draw?

Let X_i be a single draw from the uniform distribution. Then it follows that it has a cumulative density function of

$$F(x) = \begin{cases} x, & 0 \leq x < 1, \\ 1, & x = 1, \\ 0, & \text{Otherwise.} \end{cases}$$

For N draws, then, what we are looking for is $Y = \max(X_i)$. The cumulative density function of Y is equal to $\mathbb{P}(Y \leq y)$.
Since Y is the max, no independent draw X_i can be greater than y thus we can write:

$$\begin{aligned} \mathbb{P}(Y \leq y) &= \mathbb{P}(X_1 \leq y, X_2 \leq y, \ldots, X_N \leq y) \\ &= \mathbb{P}(X_1 \leq y)\mathbb{P}(X_2 \leq y) \ldots \mathbb{P}(X_N \leq y) \\ &= F(y)F(y) \ldots F(y) \\ &= [F(y)]^N \end{aligned}$$

The above cumulative density function is continuous, so we can find the probability density function of Y by taking its derivative: $NF(y)^{N-1} \times f(y)$ and since we are over $[0,1]$, $f(y) = 1$.

Then reminding that $\mathbb{E}(X) = \int_a^b xf(x)\,dx$, we can calculate:

$$\begin{aligned}\mathbb{E}(Y) &= \int_0^1 y(Ny^{N-1})\,dy \\ &= \int_0^1 Ny^N\,dy \\ &= \left[\frac{N}{N+1}y^{N+1}\right]_0^1\end{aligned}$$

So we obtain:

$$\boxed{\mathbb{E}(Y) = \frac{N}{N+1}}$$

Symmetrically we can obtain that on average the lowest draw is given by:

$$\boxed{\mathbb{E}(Y) = 1 - \frac{N}{N+1}}$$

Let's go back to our model. As we have 2 players, $N = 2$ so the expected highest valuation of the good is $v_i = \frac{2}{3}$ and the expected lowest valuation is $v_j = \frac{1}{3}$ which is more generally $v_j = \frac{1}{2}v_i$.

We can see that the expected payment by the winner is as we have seen in the theorem:

$$\boxed{b_i^* = v_j = \frac{1}{2}v_i = \frac{1}{3}}$$

Now let's calculate the expected surplus of the winner. Notice that if a bidder has value v_i, he expects to win whenever the other bidder has a value less than v_i; which happens with probability equal to v_i. We can deduce the expected surplus: $S(v_i) = v_i(v_i - \frac{1}{2}v_i)$.

$$\boxed{S(v_i)^* = \frac{1}{2}v_i^2 = \frac{2}{9}}$$

2.2.2 First-price auction

Now we need to find the expected bid and revenue of this type of auction in order to compare with the second-price auction results and so prove the revenue equivalence theorem. We will first clarify the payment rule for this auction, then we will show that the optimal strategy for each player is to bid half of their maximal valuation of the good noticed v_i for player i. Finally, we will deduce that the expected bid of player i is $b_i^* = \frac{1}{2}v_i = \frac{1}{3}$ and the expected surplus is given by

$S(v_i)^* = \frac{1}{2}v_i^2 = \frac{2}{9}$ exactly the same as the second price auction.

The payment rule of this auction is the following:

- if $b_i < b_j$, bidder i pays 0
- if $b_i > b_j$, bidder i pays b_i

A first price auction with two risk-neutral bidders whose valuations are independently drawn from a uniform distribution $U[0,1]$ has Nash equilibrium strategies: $(\frac{1}{2}v_i,\ \frac{1}{2}v_j)$. Let's prove that this equilibrium exists.

Assume that bidder j bids $b_j = \dfrac{1}{2}v_j$, we need to find the best response b_i of bidder i.
Player i wins when $b_i > \dfrac{1}{2}v_j$, namely $v_j < 2b_i$. If this inequality is true, then player i has a surplus equals to $v_i - b_i$.
Inversely, player i loses when $v_j > 2b_i$ and so has a surplus equals to 0.
We can then deduce the expected surplus of bidder i given the strategy of bidder j.

$$\begin{aligned}
\mathbb{E}(S_i) &= \int_0^{2b_i} (v_i - b_i)\ dv_j + \int_{2b_i}^{1} 0\ dv_j \\
&= \Big[(v_i - b_i)v_j\Big]_0^{2b_i} \\
&= 2v_ib_i - 2b_i^2
\end{aligned}$$

To find the best b_i we have to maximize the expected surplus by taking its derivative and by equalizing it to 0.

$\dfrac{\partial \mathbb{E}(S_i)}{\partial b_i} = 0 \Leftrightarrow 2v_i - 4b_i = 0$, so we have: $b_i = \dfrac{1}{2}v_i$.
The best strategy of player i is to bid half of his valuation, and as the game is symmetric player j's best response is $\dfrac{1}{2}v_j$.
As shown in the second-auction case, assuming that valuations are drawn independently of a uniform distribution and that $v_i > v_j$, the expected valuation of bidder i is $v_i = \dfrac{2}{3}$. As $v_i > v_j$, bidder i wins and pay its bid, so we have:

$$\boxed{b_i^* = \frac{1}{2}v_i = \frac{1}{3}}$$

The optimal bid of the winner is exactly the same under second-price and first-price auction. Here again, notice that if a bidder has value v_i, he expects to win whenever the other bidder has a value less than v_i; which happens with probability equal to v_i. The expected surplus will be the same as the second-price auction one: $S(v_i) = v_i(v_i - \dfrac{1}{2}v_i)$.

$$\boxed{S(v_i)^* = \frac{1}{2}v_i^2 = \frac{2}{9}}$$

More generally, for any given standard auction, denoting $S(v)$ the expected utility of player with a valuation v of the good and b the bid he called out, his surplus is the following:
$S(v) = v\mathbb{P}(v) - b$, so
$S(v)' = \mathbb{P}(v) = v$ in our context. $S(v)'$ is the probability to win the auction. Now for any standard auction, we denote $S(0) = 0$, namely a bidder with the lowest possible valuation of the good make 0 expected surplus.
Given a uniform distribution $U[0,1]$, the *Fundamental Theorem of Calculus* gives us that:
$S(v) = S(0) + \int_0^v S(v)'\ dv = \int_0^v v\ dv$ which gives us the expected surplus of a bidder with valuation v for any standard action.

$$\boxed{S(v) = \frac{1}{2}v^2}$$

2.3 Entry cost in Vickrey's auction

The objective of this part is to ease the assumption that there is no entry cost in order to find the optimal bidding strategy as part of the second-price auction. The main difference with the previous analysis is that the game is now in two steps: the first step is to decide whether to participate in the auction and the second step is to define the optimal bidding strategy.

Let's focus on a simple case to understand the mechanism of the first step. We will see that the objective is to find a cut-off, where we decide to enter the auction. The idea is that this cut-off will be called $\tilde{v}$ and if our valuation is higher than $\tilde{v}$ then we enter, if our valuation is lower, then we do not participate.

To find $\tilde{v}$ let's take a simple model in the case of a Vickrey's auction (second-price auction). There are two bidders whose valuations are drawn from a continuous distribution $U[0,1]$ and there is a cost equal to F if we enter the auction.

Firstly, the objective is to find the threshold $\tilde{v}$. As $\tilde{v}$ is the minimal valuation that allows a participation, a bidder with a valuation equals to $\tilde{v}$ wins only if he is the only one to participate. He is the only one to participate only if other players have a valuation inferior to $\tilde{v}$ which happens with the probability: $\mathbb{P}(v_1 < \tilde{v}) \times ... \times \mathbb{P}(v_{N-1} < \tilde{v})$ which is equal to $\tilde{v}^{N-1}$. We assume that the reserve price is 0. Then the bidder with valuation $\tilde{v}$ would have a payoff equals to: $\tilde{v} \times \tilde{v}^{N-1} = \tilde{v}^N$. The last step in order to find the cut-off $\tilde{v}$ is to consider the case when the bidder which has a valuation $\tilde{v}$ is indifferent between participating or not to the auction. This player is indifferent when his payoff is equal to the cost of entry, namely: $\tilde{v}^N = F$ which gives us: $\tilde{v} = F^{\frac{1}{N}}$.

Now that we have determine the first stage of the game, we can deduce the optimal bidding strategy in the second stage. If a player has a valuation higher than $\tilde{v}$ then the optimal strategy doesn't change with the previous case. The Nash equilibrium strategy is to bid his own valuation of the good. The solution with entry cost in a Vickrey's auction is the following:

$$b_i = \begin{cases} v_i, & v_i > \tilde{v} = F^{\frac{1}{N}}, \\ \text{No entry}, & v_i < \tilde{v} = F^{\frac{1}{N}} \end{cases}$$

The very interesting thing about the cost of entry is the idea of sunk cost fallacy. The basic previous model predicts that once you have paid the entry cost you should bid the same way as you would if there were no entry cost. We can imagine that intuitively players are encouraged to deduce the entry cost to the bid they wanted to call before the auction. However, the previous model show that when a bidder has paid to enter the auction, any entry fee that he might have paid should have no bearing on his bidding strategy.

3 Temporal pacing

Time pacing is the main problem concerning the budgetary expenditure of a campaign. The objective is to create an algorithm capable of smoothing the advertising expenditure over a given time interval. Notice that pacing can concern other dimensions than time, for example, we can also consider smoothing the advertising expenditure over space but in our case we will first focus on the temporal dimension.

This algorithm is built into the RTB part of the platform. Firstly, let's set up a simulation framework that is as close as possible to reality. Each day Displayce receives bid requests during the day containing information such as the precise timestamp of reception, the number of impressions and the total price of the bid request (which therefore depends on the number of impressions). When Displayce receives a bid request, it passes through a series of automatic filters which depending on the information will return a response in the form of 'Yes' or 'No'. The pacing algorithm will be the last filter on the bid requests that needs to be answered. If all the filters say 'Yes' then a positive response is sent to the SSP that have sent the bid request and within a delay ranging from 2 seconds to 15 minutes a notification is sent back to Displayce. This notification confirms the purchase or cancels it. Even if the reception of bid requests can be approximately known in advance, the number of impressions is unknown and is an exogenous factor because it depends on the number of people in front of a digital panel. Consequently, we will see that the estimation of the number of impressions per hour will be the key point to set up an efficient pacing algorithm.

3.1 Simulation

The objective is to simulate a situation that is as close as possible to reality but with some control over the parameters in order to develop an algorithm that works in the majority of cases. The key point is to obtain a simulation that generates bid requests with a variable delay between each bid request. In reality, this delay depends on the number of persons in front of a digital panel, namely the number of impressions. So we simulate the time between two bid requests thanks to a random variable which follows a Poisson Law of parameter λ ($Pois(\lambda)$) where the λ parameter can be thought of as the expected number of events in the interval. In our case the parameter λ is then the number of seconds between two bid requests. As the number of seconds between two bid requests is not stable during the day, we should be able to set a scalable λ parameter to simulate slowdowns (high number of seconds between two bid requests) and accelerations. To do so, we decided to set per day threshold parameters, and we can link them by linear interpolation in order to have an evolutive function. As each day has its own threshold parameters, each day behaves differently but every Monday for example looks the same.

The simulation gives us a data frame with the following structure:

	ID	Imp-ressions	Price	Win	Seconds before response
2020-07-08 06:00:00	1.0	10.0	10.0	True	365.0
2020-07-08 06:00:00	2.0	1.0	1.0	True	361.0
2020-07-08 06:00:01	3.0	5.0	5.0	True	555.0
2020-07-08 06:00:01	4.0	1.0	1.0	True	645.0
2020-07-08 06:00:03	5.0	1.0	1.0	True	310.0
2020-07-08 06:00:03	6.0	2.0	2.0	True	355.0
2020-07-08 06:00:07	7.0	8.0	8.0	True	775.0
2020-07-08 06:00:10	8.0	3.0	3.0	False	653.0
2020-07-08 06:00:11	10.0	6.0	6.0	True	432.0
2020-07-08 06:00:11	9.0	5.0	5.0	True	458.0
2020-07-08 06:00:12	11.0	5.0	5.0	True	755.0
2020-07-08 06:00:13	12.0	1.0	1.0	True	699.0
2020-07-08 06:00:14	13.0	9.0	9.0	True	2.0
2020-07-08 06:00:15	14.0	6.0	6.0	True	582.0
2020-07-08 06:00:16	15.0	5.0	5.0	True	52.0
2020-07-08 06:00:18	16.0	3.0	3.0	True	171.0
2020-07-08 06:00:21	19.0	4.0	4.0	True	636.0
2020-07-08 06:00:21	18.0	3.0	3.0	True	815.0
2020-07-08 06:00:21	17.0	5.0	5.0	True	447.0

The last two columns allow us to take into account the delay of notification after responding to a bid request. In fact, when we receive a bid request, and we want to buy it, the purchase is not immediate. The price goes into a part called committed budget a few moments later (between 2 seconds and 15 minutes) we receive a notification of response that either confirms the purchase (in this case the budget will be an actually spent budget) or rejects it and in this case the budget will not be spent. Theoretically this information doesn't change anything but it would change the behaviour of the algorithm technically because it needed to know the state of the real expenditure each time in order to calculate fairly the remaining budget.

3.2 Algorithm evolution

In this section we will really detail the scientific approach of building the algorithm step by step. The structure of evolution is the following: we start from the basic algorithm already implemented, and thanks to simulated data we are able to create situation in which the algorithm is not efficient The objective is then to find an improvement that makes it efficient, and we repeat the process until we get a suitable algorithm. Finally, thanks to a set of real data over a week, we will be able to test the algorithm in real conditions.

3.2.1 Hourly capping

The algorithm already implemented at DISPLAYCE is a time capping algorithm. When a campaign is launched, it is given a total budget and release dates. The algorithm will then divide the total budget into an hourly budget which will be a capping (maximum budget that can be spent per hour). If the impressions potential is high, in other words, if we receive a lot of impressions per hour, the total budget will be spent and each hour will have the same budget spent. Basically this meets the objectives of pacing. Figure 2 shows how a daily budget is spent with an hourly capping algorithm.

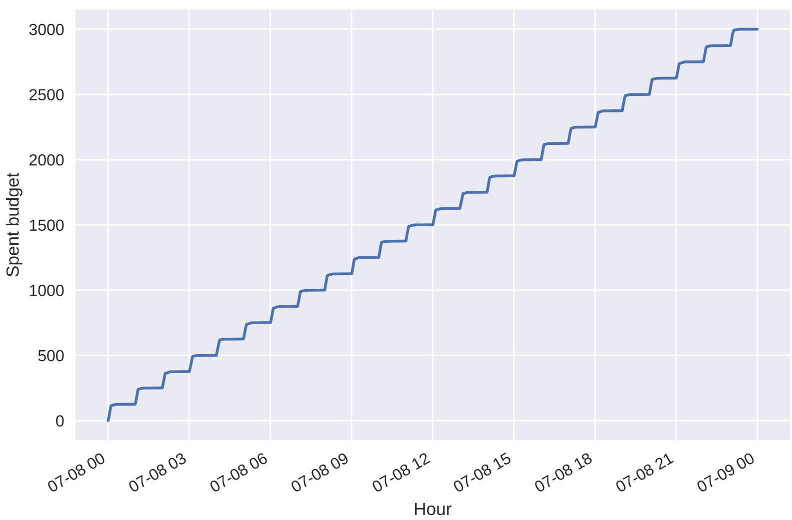

Figure 2 – Spent budget over a day with capping algorithm (2020-07-08)

We notice that the curve is in the shape of a staircase. This phenomenon is explained by the fact that the algorithm buys as long as it has budget left to spend (the curve increases) and therefore stops buying when the hourly capping is reached (the curve remains horizontal for the rest of the hour). This is a limit of the algorithm. First the expenditure within each hour is not completely uniform since as long as there is budget left the algorithm systematically makes a purchase decision. If we receive a lot of impressions per hour as it is the case graphically, the algorithm will spend the entire budget for the hour on the first few minutes and will stop buying for the whole end of the hour and therefore does not guarantee uniformity. The second limitation is the situation where we don't get enough impressions to spend the whole budget in a given hour. The problem with the current algorithm is that the unspent budget of one hour will never be spent later and therefore when the campaign is finished the whole budget has not been spent and there is a surplus left. The following graph (Figure 3) shows that the budget spent during the day does not reach the €3000 of daily budget.

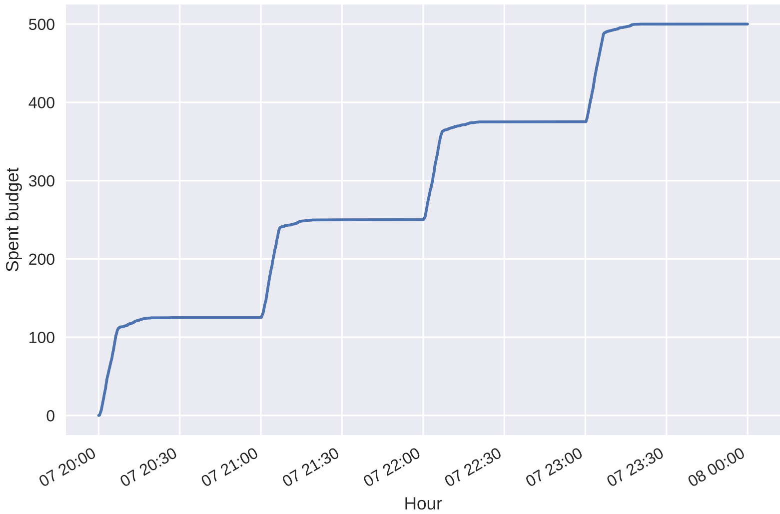

Figure 3 – Spent budget over a day with capping algorithm (2020-07-07)

The curve looks exactly the same as Figure 2 but this time the curve doesn't reach the €3000 daily budget at the end of the day. Indeed, we can see that the spending started at 8 p.m and therefore the unspent budget from the previous hours have been definitively "lost".

3.2.2 Per second budget

In order to resolve the limitations of the previous algorithm, it was decided to implement an algorithm that calculates a maximum budget per second that can be spent. Calculating this budget per second would theoretically allow a more uniform expenditure over the day and thus not end up with a stepped curve. Moreover, since the budget per second is recalculated each time a bid request is received, this allows the budget that has not been previously spent to be spent and to be freed from the constraint of the budget per hour. Given the fact that we know when the day ends, we calculate an ideal per second budget called 'target'. Then we try to keep the real expenditure per second close to this target. The idea is to have the same behaviour as a cruise control algorithm, but instead of controlling the speed we control the budget delivery. It posed the problem as a mathematically constrained optimization. The key point here is that currently we don't have a measure of quality of impressions and therefore we can't maximize on a variable that would measure the impressions' quality. That's why, we can simply reduce such an optimization program to the following form:

$$\max_{b_t} \quad I$$

$$s.c. \qquad \begin{cases} \sum_{t=1}^{T} s(t) = B \\ |s(t) - b_t| \leq \varepsilon_t \end{cases}$$

Where I is the number of impressions, $s(t)$ is the time expenditure t, B is the total budget and ε is a number small enough to ensure the uniformity of expenditure. The first constraint is the budget constraint, and the second constraint guarantees the objective of smoothing the expenditure in order to avoid lows or peaks of expenditure during the day. The optimization parameter is therefore b_t, the budget that is allocated in t. However, the number of impressions is exogenous, indeed the presence in front of a panel of individuals in t does not depend on us. This is why at this stage the only way to guarantee a smooth expenditure over the day which thus respects the constraints presented above is to have a classic pacing algorithm which offers a budget per t slice of the day. The algorithm could therefore estimate the times at which we receive bids requests and according to these times divide the day into t time slots (in seconds or even more precisely) and therefore assign a budget per time slot.

$$b_{t+1} = \left(B - \sum_{s=1}^{t} S(s)\right) \frac{1}{T - t}$$

where b_{t+1} is the budget to be allocated to the second $t + 1$, B is the total budget for the day, $S(s)$ is the actual expenditure to the second t and finally $T - t$ is the remaining time in seconds until the end of the day. It is important to highlight that the operation explained above is based on the estimation of the exact time of the last bid request, so that the $T - t$ calculation is precise and fair.

Since the algorithm is based on increasing the remaining budget per second when not enough impressions are received, the calculation of the budget per second can already be slightly improved by taking into account its variation. Indeed, if it increases, it means that we receive less and less impressions and therefore we buy less and consequently we should accelerate the purchase to be sure to spend the whole budget. We can therefore take into account the variation $v_t = b_t - b_{t-1}$ and the speed of the variation $a_t = v_t - v_{t-1}$.
The calculation of b_t becomes the following:

$$b_{t+1} = \left(B - \sum_{s=1}^{t} S(s)\right) \frac{1 + (\bar{a}\bar{v})}{T - t}$$

Where $\bar{a}$ is the average speed in the variation of b_t over the last 30 minutes and $\bar{v}$ is the average variation of b_t over the last 30 minutes as well. If b_t increases more and more, $\bar{v}$ and $\bar{a}$ will increase as well, it would consequently boost the buying to catch up if we ever get less and less impressions. We can then simulate this new algorithm on the same data as the previous one and see if it better meets the objectives and thus surpasses the limits of the hourly capping algorithm. Figure 4 shows us that the expenditure over the day is much smoother and more uniform than the hourly capping. We can notice that the expenditure curve becomes horizontal at the very end of the day. This behaviour is explained by the fact that in the calculation of the remaining time, the algorithm considers the end of the day at 11:40 pm to ensure that the entire daily budget is spent.

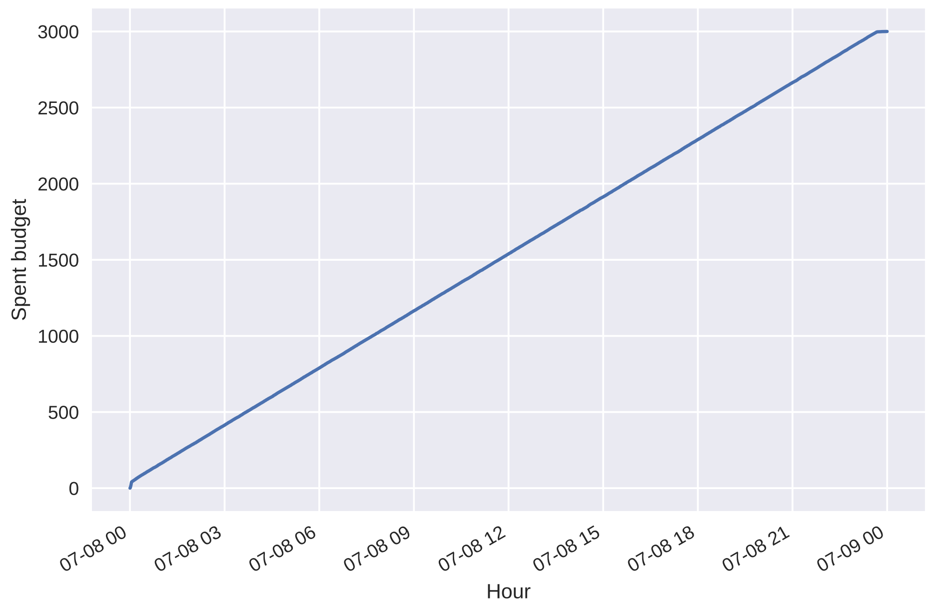

Figure 4 – Spent budget over a day with budget per second algorithm (2020-07-08)

The second improvement of this algorithm compared to the previous one can be seen on the day the bid requests arrive at 8pm. In this configuration Figure 5 shows us that the unspent budget from the previous hours is caught up at the end of the day and therefore all the budget has been spent contrary to the hourly capping algorithm. We therefore conclude that the algorithm with the calculation of the budget per second is more efficient than the hourly capping algorithm.

The disadvantage of this algorithm is that when there is a backlog to catch up on, as is the case here, the algorithm will buy until it catches up and then slow down the purchase until the end of the day. It is this behaviour that explains the elbow in spending around 10:20 pm. We could then find a better way to standardize the expenditure to avoid this elbow.

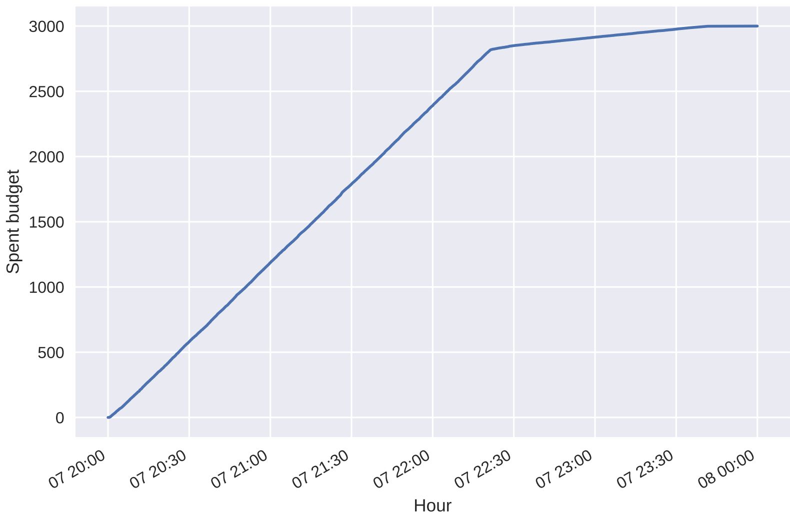

Figure 5 – Spent budget over a day with budget per second algorithm (2020-07-07)

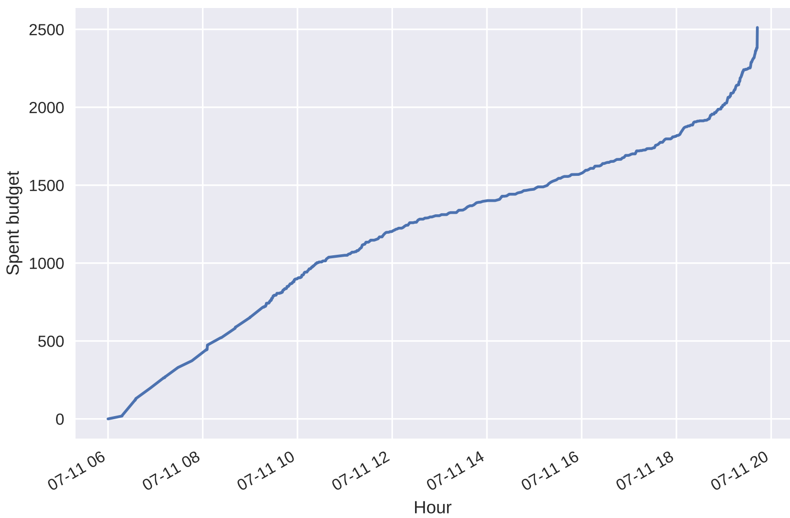

Figure 6 – Spent budget over a day with budget per second algorithm (simulation)

Despite the efficiency of this algorithm, when there is a big slowdown in the number of impressions during the day and the number of impressions received in the remaining hours is not enough to catch up, then there will be some budget left at the end of the day. To show this situation, we can use simulated data to create a big slowdown in the course of the day. The result of the algorithm are shown in Figure 6. It can be seen that the slowdown could not be caught up before the end of the day and therefore the budget was not fully spent (only €2500 were spent).

3.2.3 Evolutive hourly budget

We are now going to discuss the new limitation that have been raised. The only way to deal with this is to be able to predict whether there will be a slowdown in the day. One can imagine that in the DOOH sector, where the number of impressions depends on the number of people in front of the digital panels, each day is more or less the same and the hours when there are fewer people are perhaps predictable. The idea is to use historical data and perform a simple linear regression on hours of the day and days of the week to predict the proportion per hour of impressions received. If the hours with a lower proportion of impressions received are known, then a lower budget will be allocated to those hours and conversely for hours with a high proportion of impressions received. The calculation of b_t then becomes:

$$b_{t+1} = \left(B(h) - \sum_{s=1}^{t} S(s) \right) \frac{1 + (\bar{a}\bar{v})}{T - t}$$

Where $B(h)$ is the budget allocated to the hour h, $\sum_{s=1}^{t} S(s)$ is the budget spent in the current hour h and $T-t$ is the remaining number of seconds before the end of the hour h.

The calculation of $B(h)$ will depend on how long the campaign has been running. Indeed, on the first day we have no information on the number of impressions received during the day because we don't know which SSP will send bid requests nor at what time of the day they will do it. The best distribution that can be made without information in this case is a uniform distribution. From the second day a linear regression based on the data of the first day can be done to estimate the proportion per hour. This regression will therefore have the proportion of impressions as an explained variable and each hour of the day as an explanatory variable.

$$Prop_i = \alpha + \boldsymbol{\beta_k} \mathbf{X_k} + \varepsilon_i$$

Where $\boldsymbol{\beta_k} \mathbf{X_k}$ is the vector of coefficients associated with the vector of explanatory variables. When we have at least 7 campaign days, we can add the day of the week in the vector of explanatory variables in order to have a more accurate estimation.

This method therefore makes it possible to predict (if the estimation is efficient) the hours during which there is a big slowdown and thus be able to allocate a larger budget to the hours before the slowdown. In addition, the algorithm can also be improved by distributing the remaining budget of the previous hours over all the remaining hours of the day to avoid an elbow-shaped expenditure curve.

Figure 7 gives us the same result as the previous algorithm to prove that we haven't regressed to a 'normal' situation. Figure 8 shows us that when there is a catch-up to be made, the algorithm catches up with the unspent budget much more uniformly than the previous one. There is no longer the elbow shape and all the daily budget has been spent. Figure 9 compares the three algorithms on the same day to see how the last one is much more efficient. The algorithm with a hourly budget per second not only makes it possible to catch up on unspent

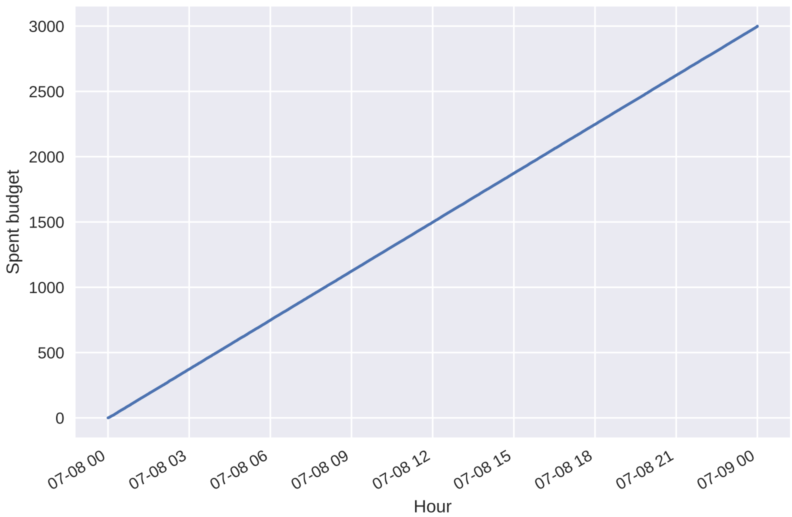

Figure 7 – Spent budget over a day with hourly budget per second algorithm (2020-07-08)

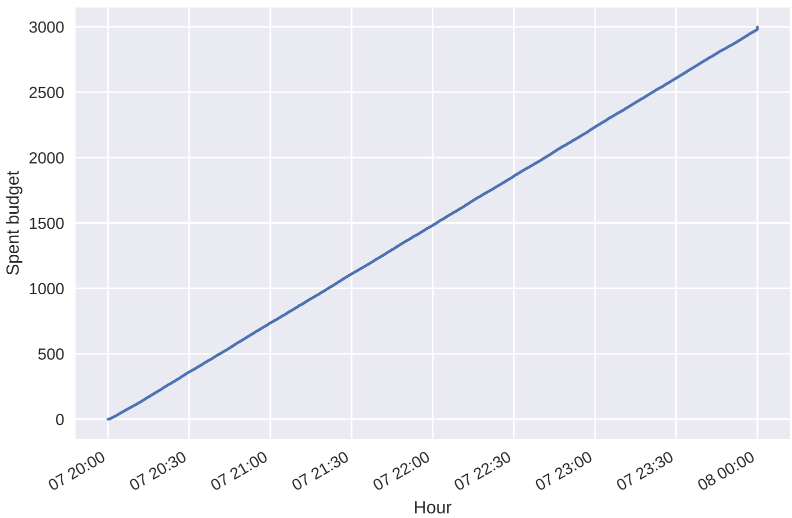

Figure 8 – Spent budget over a day with hourly budget per second algorithm (2020-07-07)

budget before the end of the day, but also the expenditure over the last hour is uniform unlike the other two algorithms. Thus, it better meets the objectives of pacing where other algorithms have limitations.

Furthermore, if we run the algorithm on simulated data, the algorithm which have learned the distribution of impressions on historical data is much more efficient as shown in Figure 10. We were able to increase spending before the slowdown and so the entire budget was spent by the end of the day.

Despite the fact that this new algorithm is more efficient than the two previous ones, it also has its limitations. Indeed, if a slowdown not foreseen by the esti-

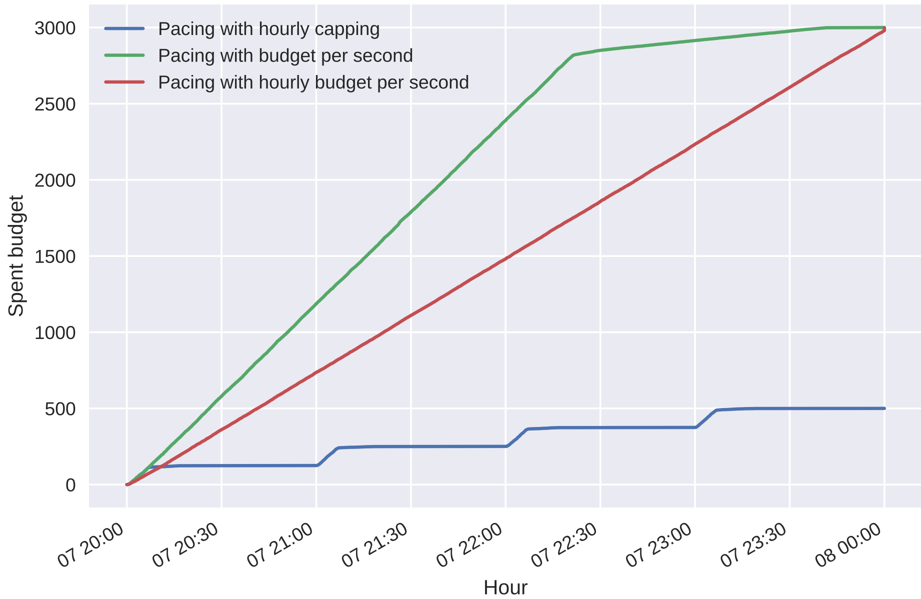

Figure 9 – Comparison of 3 algorithms (2020-07-07)

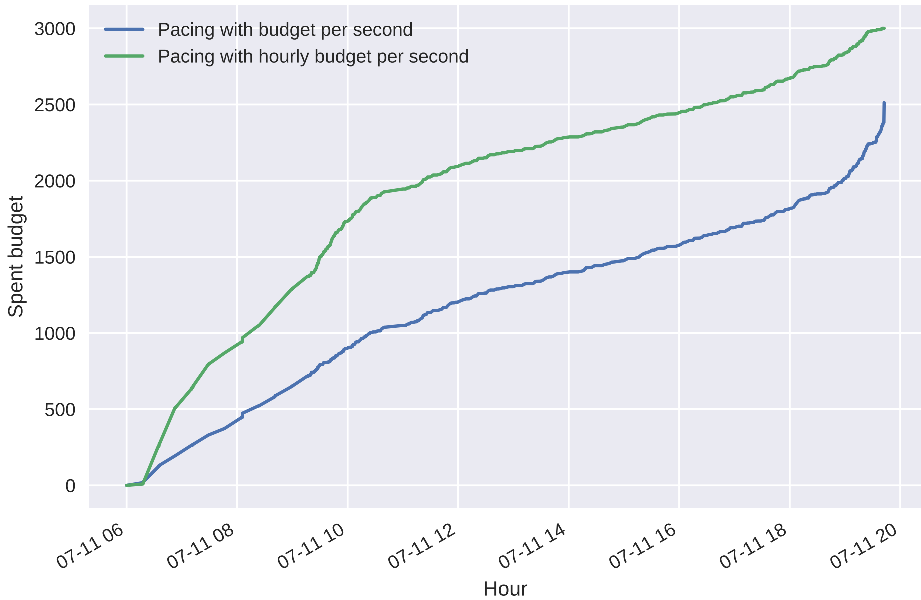

Figure 10 – Comparisons of algorithms (Simulated)

mates occurs, then it will not be efficient. Even if this aspect seems difficult or even impossible to improve, we can work on other ways to improve the algorithm. However, a campaign normally lasts several days and the algorithm presented above incorporates a distribution of the unspent budget at the end of one day over the following days. This pacing between days then makes it possible to reduce the risk that there is a budget remaining at the end of a campaign because if ever there was an undetected slowdown on a given day, the unspent budget on that day will be distributed over the remaining days before the end of the campaign. We will then see what are the next steps to improve the algorithm.

Actually, the current state the algorithm is efficient when it receives bid requests from a single time zone. However, in reality it is necessary to integrate to the algorithm the constraint of the time-zones because with the time shift the current expenditure per hour is no longer guaranteed. We will therefore look at the spatial dimension in the next section.

4 Spatial pacing

4.1 Pacing meta class

The purpose of this section is to give the algorithm the possibility to add other dimensions to the advertising expenditure pacing criterion. Initially, we consider the possibility of adding only a spatial dimension to the temporal dimension, but very soon it will be necessary to consider that the algorithm can take other dimensions into account. These can be based for example on sociodemographic criteria. We could say that we wish to distribute the budget uniformly over 3 days, between Europe and the United States, and allocate 75% of the budget to target the over 50s years old and 25% to target the under 50s.

In order to integrate the time zones, it will already be necessary to make some modifications in the code to integrate the time difference between time zones. Indeed, one could imagine taking back the initial algorithm and when we calculate the budget per hour, we divide it equitably between each time zone. However, the time difference makes this idea impossible because the budget per hour will not be calculated at the same time for a given hour. The most obvious solution is then to do the opposite and initially allocate a budget per time zone and run the time pacing algorithm independently for each time zone.

This is where the idea of a meta-class comes in. In Python language, the algorithm is based on an object called a class. This class is autonomous in its functioning, i.e. when it is given original inputs concerning a campaign, the class will manage to know its state of progress and thus take the decision to buy or not to buy. In practice, we create what is called a class instance for each campaign by providing the following information: the campaign start date, the end date, the time zone concerned and the total budget allocated to the campaign. As a pacing class is autonomous, the idea is then to create a meta-class that would create its own instances of the pacing class for each time zone entered the meta-class. In fact, it is as if we recreate an algorithm per time zone, with the ease that each of these algorithms are connected to the same class.

Just like the class in the temporal dimension pacing algorithm, the meta class is given the same inputs, namely, the start date of the campaign, the end date, the total budget allocated. The only difference is on the last parameter because previously it was necessary to fill in a time zone, now it is necessary to give the meta class a list of diffusion time zones. From this new parameter, the meta class will create as many instances of the pacing class as there are time zones in the list. All the difficulty from now on will be to manage the interaction between the meta class and all its instances to determine the distribution of the total budget between all the time zones and this is what we will discuss in the next point.

4.2 Application of the meta-class

Currently, I have been able to implement the meta-class with an initial proportion based on historical data. Indeed, when the algorithm is launched, it uses data (that it is not supposed to know) in order to determine over the period of the campaign exactly the proportion of impressions received by time zone. When it creates an instance of the pacing class for each time zone, the algorithm allocates it the share of the total budget according to the proportion of impressions received. For example, if we have two time zones: America/New-York and Europe/Paris, and on the campaign data we receive 75% of the impressions of the America/New-York time zone and therefore 25% for the Europe/Paris time zone, then if we have to distribute €1000 of budget between these time zones, the algorithm will allocate €750 for the America/New-York time zone and €250 for the Europe/Paris time zone.

This new algorithm has been executed on a database of 3,342,804 bid requests with 11 different time zones. The start date of the campaign is July 8, 2020 and the end date is July 10, 2020. There is €10000 of budget allocated to this campaign. There remains €0.5 in the budget at the end of the campaign, so the total expenditure target of the budget has been reached. In addition, the budget was spent uniformly within each time zone as shown in Figure 11.

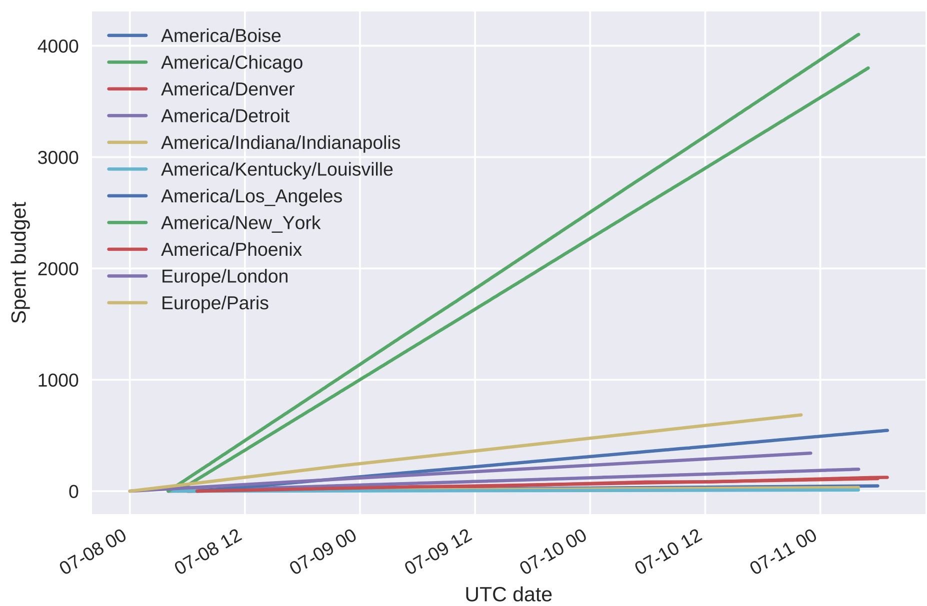

Figure 11 – Spent budget over 3 days of campaign for each time zone

The previous graph shows us the evolution of the expenditure during the 3 campaign days for each time zone. The first observation that can be made is that each time zone has a relatively uniform expenditure, without peaks or lows. The two green lines that stand out from the others correspond respectively to the time zones America/New-York and America/Chicago. We can see that both time zones are in the majority and therefore the budget allocated for each of them represents a large part of the total budget of the campaign, which explains why the expenditure is much higher for both time zones. The problem with the current state of the algorithm is due to the fact that the proportion of impressions received per time zone cannot be known in advance and therefore the algorithm cannot optimally allocate a budget per time zone prior to the launch of the campaign. We will see next which solutions are considered to reallocate the budget in an optimal way during the campaign. It is however important to note that the next part is only indicative because currently the algorithm is not complete and stops at the stage where the proportions are given in advance.

4.3 Budget reallocation between time zones

As we cannot know in advance the proportion of impressions received per time zone, when a campaign is launched, the algorithm has no choice but to distribute the budget equally among the time zones. If, for example, the campaign takes place on 4 different time zones, then each time zone will be allocated 25% of the budget. The problem of this distribution appears in situations where we receive very few impressions from a time zone; if this is the case then we will not be able to spend the totality of the budget allocated to this time zone and therefore this would not meet the objectives of the pacing which are, let us recall, to spend the budget uniformly and in its totality.

From there, the idea is to adjust in real time the budget allocated by time zone. To do so, the algorithm could look for each time zone the proportion of bid requests purchased. If the latter is at 100% and the expenditure target is not reached then this means that we do not receive enough impressions for this time zone and therefore we have to reallocate part of the budget to other time zones. Let's take a numerical example to illustrate this situation.

Let T_1, T_2, T_3 and T_4 stand for 4 time zones. The campaign lasts one day and the total budget is €1000. At the very beginning the algorithm allocates 25% of the budget to each time zone, thus, $T_1 = T_2 = T_3 = T_4 = 250$. Assuming that T_1 is the minority time zone, the proportion of bid requests bought buy T_1 is 100%. Moreover, even if we buy 100% of the bid requests in T_1 we expect that we can meet only half of the original objective in this time zone. Therefore, the algorithm we reallocate 50% of €250 to the other time zones, namely, $\frac{125}{3}$ for each time zone. We will then have $T_1 = 125$ and $T_2 = T_3 = T_4 = 250 + \dfrac{125}{3}$

Each time we receive a bid request, we repeat the previous calculation until we reach a stable situation where each time zone is allocated an optimal budget that can be fully spent. Thus, we can give the algorithm only information known in advance, such as broadcast dates, the total budget and the list of time zones. The algorithm will be able to spend the budget in order to meet the initial pacing objectives.

In addition, it should be noted that the structure of the meta-class presented above has the advantage of adapting to all the dimensions of the pacing that one wishes to provide. Indeed, as the meta-class creates its own instances of the pacing class, one could imagine providing not only a time zone list but also a targeting list based on age, for example. If the algorithm then receives a list of 4 time zones for example with two distinct age ranges, the meta class would create 8 instances, i.e. two per time zone corresponding to the two age ranges.

5 Conclusion

The initial objective of this report was to implement a pacing algorithm that is more efficient and better suited to the objectives than the hourly capping algorithm currently in place. It should be remembered that the objective of pacing is to spend the advertising budget of a campaign uniformly and in its entirety. The old algorithm that works on the basis of hourly capping does not effectively meet these objectives. Indeed, the budget allocated each hour is spent at the very beginning of the hour, which does not guarantee a uniform expenditure, and when the budget of a given hour could not be spent, then it will never be spent, which does not guarantee the objective of spending the budget in its entirety.

The solution proposed in this report to improve the algorithm is based on the calculation of a target budget per second updated every hour. When a campaign is launched, the algorithm allocates an initial daily budget. On the first day, it allocates a fair proportion of this daily budget per hour. Based on this budget per hour, at each new hour the algorithm calculates an ideal per-second spending target, i.e. how much it would have to spend per second to reach the spending target. If the remaining budget per second is greater than this target then the algorithm buys, otherwise it does not buy. The unspent budget at the end of an hour is spread over the following hours of the day and more globally the remaining budget at the end of a day is spread over the following days of the campaign. On each new day, the algorithm allocates a budget per hour based on the historical data of that campaign, so hours with a lower proportion of received impressions will be allocated a lower share of the daily budget and conversely for hours with a higher proportion of received impressions. This algorithm meets the original pacing objectives and is therefore more efficient than the old capping algorithm.

It should be noted, however, that the time zone constraint must be added to the previous algorithm in order to put it into production. Indeed, the latter only works for a single time zone. The addition of the spatial dimension thus required a modification of the algorithm. The solution found is based on the use of a meta class that creates its own instances of the pacing class. During its initialization, the algorithm will create one instance per time zone. Currently, the distribution of the campaign budget between each time zone is based on the proportion of impressions received for these time zones. The fact that this distribution is known in advance is not at all realistic, which means that in this state the algorithm cannot be put into production.

Although the algorithm built in this report seems to be more efficient than the hourly capping algorithm already implemented, there are clear prospects for improvement.

Firstly, in order to make the algorithm usable, a solution must be found to distribute the budget in an optimal way between the instances of the meta class. The envisaged solution has been explained in the last part of this report and has not yet been produced. The idea is initially to allocate to each time zone the same budget and then to calculate for each time zone the proportion of bid requests purchased. If the proportion of bid requests purchased is 100% and the expenditure target is not guaranteed, then the algorithm distributes the surplus budget to the other time zones. If this version of the algorithm proves to be functional, then, it could replace the time capping algorithm.

Another area for improvement concerns the part of the algorithm in which the proportion of impressions received per hour is estimated. During the first days of the campaign, the linear regression that estimates this proportion has very few points and therefore the estimate is potentially imprecise. One could then imagine a confidence interval calculation which would be an acceptance criterion for the estimate. If this confidence interval is too large, one could imagine that the algorithm will formulate an estimate of the proportion that is too imprecise and therefore would keep a uniform distribution between the hours of the day.

Appendix

You can get all Python code and notebooks to my Github repository: Repository

References

[1] Paul Klemperer. *Auctions: Theory and Practice*. SUNY-Oswego, Department of Economics, 2004.

[2] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. Real time bid optimization with smooth budget delivery in online advertising. 05 2013.