Master Économiste d'entreprise

Internship in Data Science

Thomas Maurice

Displayce
M. Soueidan

Université de Tours
M. Favard

18 juin 2020

Table of contents

1 Introduction

1.1 What is Displayce?

DISPLAYCE is the first DSP [1] and the leading technological platform designed to optimise ad campaigns run on digital posters and billboards. It is what we called the DOOH which stands for *Digital Out Of Home*. An ad campaign's impact and quality can be optimized thanks to the programmatic purchase which falling within the scope of the open real time bidding, allows to spend an ad budget efficiently over time and space. Hence, DISPLAYCE is an intermediary between supply and demand where the supply is a digital posters publisher and 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 interet 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 specialising in digital psoters 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?

As we discuss previously, DOOH allows to display an ad on a digital poster. We are now going to explain how does it work technically.
Let's take the case of a single digital panel. This panel is configured in the following way: 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. Each ad is submitted by a textcdsp 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, you 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. A bid request contains the following information: the exact time, the location of the panel, the number

1. Demand Side Platform: a technological solution that allows you to automate the purchase of advertising inventory.

of impressions [2] and the qualitative detail of the impressions, i.e. the type and age class of each impression. Based on the information received, each DSP bids and the winner is the highest bidder. Depending on the type of auction he will pay the price he announced himself or the price of the second best (we will detail the different types of auction in the next section). The winner of the auction will then see his advertisement broadcast for 10 seconds on the panel but contrary to acquired campaigns this broadcast is not guaranteed for the next minute when a new bid request will be made.

1.3 What is a bid request?

We talked earlier about bid requests and real-time bidding. Let's explain what this 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 instantaneous 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 optimise the budget expenditure, reach the largest audience, smooth the budget in time and space for example etc.

2. 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 independantly 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 independant 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 revenu 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 independantly drawn from a uniform distribution $U[0, 1]$ has Nash equilibrium strategies: $(\frac{1}{2}v_i,\ \frac{1}{2}v_j)$. Let's proove that this equilibrium exists.

Assume that bidder j bids $b_j = \frac{1}{2}v_j$, we need to find the best response b_i of bidder i.
Player i wins when $b_i > \frac{1}{2}v_j$, 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.

$\frac{\partial \mathbb{E}(S_i)}{\partial b_i} = 0 \Leftrightarrow 2v_i - 4b_i = 0$, so we have: $b_i = \frac{1}{2}v_i$.
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 $\frac{1}{2}v_j$.
As shown in the second-auction case, assuming that valuations are drawn independantly from a uniform distribution and that $v_i > v_j$, the expected valuation of bidder i is $v_i = \frac{2}{3}$.
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: $S(v_i) = v_i(v_i - \frac{1}{2}v_i)$.

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

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 to the auction and the second step is to define the optimal bidding strategy.

Let's focus on the 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, is 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 incitated 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 Pacing

Now that we introduce the how the digital out of home market works and explain some basics about auction theory, we will focus on the main subject of this report: a pacing algorithm. The objective of such an algorithm is to be able to smooth the advertising expenditure over a given time interval. The pacing concerns other dimensions than time, we can also consider smoothing the advertising expenditure over space but in our case we will first focus on the temporal dimension.

To create such an algorithm, I have worked from scratch in Python. I have taken the following steps:

- Create loops to generate bid request and given the (perfect) estimation of the number of bid requests, we calculate a per second budget and we try to keep the real expenditure per second close to the 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.
- The drawback of this method is that there is often a peak of expenditure at the end of the period and the simulation if far from the reality because there are not random events.
- The next objective then, was to improve the simulation framework to be clother from the reality.
- I decide to use a discrete event simulator on Python. This kind of simulator allows to create a real dynamic of bid request arrivals.
- I then improve the framework by adding some random variables. So the time between two bid requests follows a Poisson Law and the λ parameter follows itself a probability law such as each hour this parameter is not the same.
- Now that the simulation framework is far better, we can begin to study a new pacing algorithm. The first idea is to have a dynamic estimation of the number of bid request each hour in order to allocate an hour budget. It allows us to take into account hours with high number of impressions and so to allocate more budget in those hours of the day.
- To implement such an algorithm we can use a Kalman filter to have a dynamic estimation of the number of bid requests in real time.
- The first step before implement this algorithm is to have historical data to initialize the algorithm. I have created a function in Python that generates historical data thanks to the discrete event simulator. The variables we have are the following: the id of the bid request, the number of impressions, the price and the timestamp.
- For now, the simulation framework is the following: fix price per impression, each hour there is a different number of digital posters available, so each hour the number of seconds between 2 bid requests doesn't follow the same law. There is a 1% probability of technical incident so there is no bid requests for a long period.

- After thinking about it, I 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 such as click through rate or conversion rate. Finally we can simply reduce such an optimization program to the following form:

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

  Where I is the number of impressions, $s(t)$ is the time expenditure t, B is the total budget. 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.

- A new question that it would be legitimate to ask from now on, is the question of the quality of impressions. Indeed, despite the absence of variables such as the click through rate for example, it is questionable whether we could indirectly find ways to develop quality impressions. For example, one could imagine knowing for a city the time of passage of a certain type of population as well as the location (worker, executives, young people, elderly people, etc.). This would potentially allow a finer targeting than at present and thus have a real optimisation parameter in this program.

- But before looking at the quality of impressions, there is a pacing problem. Currently the operation explained above is based on two essential points: the first one is related to the estimation of the exact time of the last bid request, so that the $T-t$ calculation is precise and fair. Moreover it is absolutely necessary that the arrival of bid requests is more or less constant over time because if at the end of the period, the frequency of bid requests is too low, or at least not high enough to guarantee the exhaustion of the budget b_t then there will be budget left at the end of

the day and remember that the double objective is, firstly, to distribute the budget evenly over the day and secondly to ensure the total expenditure of the budget at the end of the day, the second constraint would not be taken into account. It is on this second objective that the use of the b_{t+1} calculation has its limits, and so it is here that the rest of the work will be done.

- To complete the algorithm, I thought of doing a study of the proportions of bid requests purchased, because we can assume that if the budget is not totally exhausted at the end of the day it is partly due to the fact that there is a slowdown in the arrival of bids requests and therefore there are not enough to use up the entire budget. When we get to a purchase proportion that tends towards 100%, then we could imagine accelerating the purchase of bid requests to ensure that the total budget is exhausted at the end of the day. The idea would therefore be to study the proportion of bid requests purchased per hour and thus have an evolution over the day and repeat this process on historical data to see if a pattern emerges. The Figure 1 is then obtained:

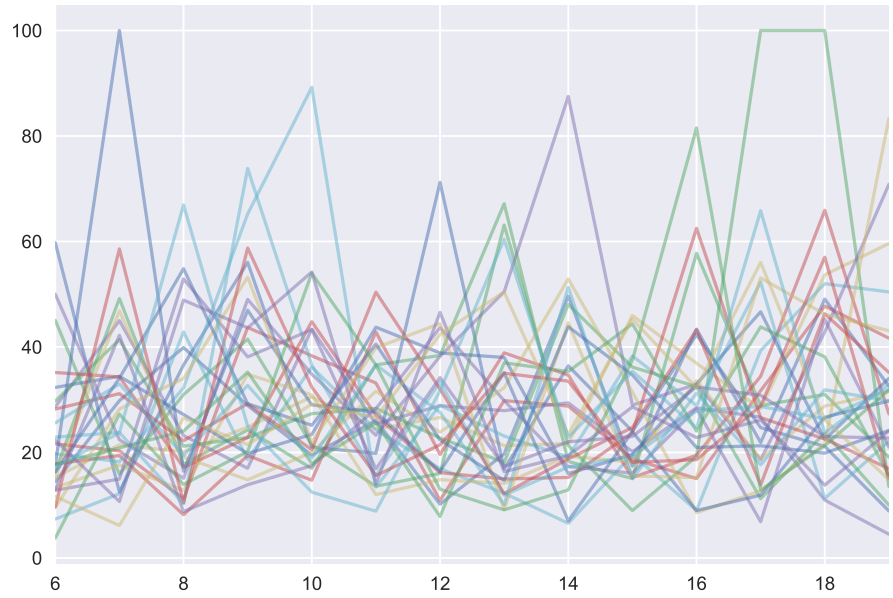

Figure 1 – Proportion of bid requests bought each hour

We can therefore see that there is no pattern that emerges on the simulated data and therefore the improvement of the algorithm will not come from the study of proportions. Nevertheless, this idea must be kept in mind because nothing excludes that it works on real data rather than simulated data.

- If we take a closer look at the simulated data, we can finally see that the cause of the problem is not directly related to the slowdown in the arrival of bid requests, but to the fact that the last bid requests end before the end time set by the simulation.

A partial solution but which works perfectly on simulated data is to make the $T-t$ period shorter than the reality in order to "naturally" accelerate the purchase over the whole day and thus exhaust the budget at the end of the day. By taking 20 minutes of margin in the calculation, one arrives at a totally satisfactory result since one respects both the constraint of uniform distribution of the budget over the day and the constraint of the total exhaustion of the budget at the end of the day. The following figure (Figure 2) shows the average expenditure per hour for each day of the simulation. It is clear that the pacing is effective because there is very few variations in the average expenditure between hours.

Figure 2 – Evolution of the average expenditure per hour

```
import numpy as np
import random
import simpy
import time
import csv
from collections import namedtuple
from datetime import datetime
import pandas as pd

def panneaux_dispo(nb_heures, total_panneaux):
        prop = np.random.normal(loc=0.8, scale=0.2, size=nb_heures)
        pann_dispo = list(map(lambda x: int(total_panneaux*x if x<=1 \
        else total_panneaux), prop))
return pann_dispo

def lambda_br(pann_dispo):
        lam = list(map(lambda x: int((1000-x)/50) if x<950 else 1, pann_dispo))
return lam

def imps():
        lam = int(np.random.normal(loc=4, scale=2, size=1))
        if lam < 1:
        lam = 1
        nb_imp = np.random.poisson(lam)
return nb_imp

def delai(lam):
secondes = np.random.poisson(lam)
        if not random.random() < 0.99:
        secondes = np.random.poisson(lam + 1000)
return secondes

def total_sec(nb_jours):
        tot = nb_jours * 86400
return tot

def sauvegarde(liste, nom_de_fichier):
        with open(nom_de_fichier, "w", encoding="utf8") as fichier:
                premier, *_ = liste
                ecrivain = csv.DictWriter(fichier, premier._fields)
                ecrivain.writeheader()
                for br in liste:
                        ecrivain.writerow(br._asdict())
```

```
def open_rtb_pacing(env, P, B, timestampnow, nb_jours, nb_hours_per_day, bidrequests, \
data):
        setup = True
        identifiant = 0
        cible = B/(nb_hours_per_day*3600)
        current_hour = datetime.strptime(datetime.fromtimestamp(env.now)\
        .strftime("%m-%d-%Y %H:%M:%S"), '%m-%d-%Y %H:%M:%S').hour
        while True:
                if current_hour >= 6 and current_hour < 20:
                        if setup:
                                setup = False
                                dispo = panneaux_dispo(nb_hours_per_day, 1000)
                                lam_secondes = lambda_br(dispo)
                                current_hour = datetime.strptime( \
                                datetime.fromtimestamp(env.now).strftime( \
                                "%m-%d-%Y %H:%M:%S"), '%m-%d-%Y %H:%M:%S').hour
                                i = 0
                                S = list()
                                end_day = env.now + (nb_hours_per_day*3600)

                        identifiant += 1
                        rt = (timestampnow+total_sec(nb_jours))-env.now

                        bt = (B - sum(S))/(end_day - env.now)
                        if bt < 0:
                                bt=0

                        time = datetime.fromtimestamp(env.now).strftime( \
                        "%m-%d-%Y %H:%M:%S")

                        nb_imp = imps()
                        Prix = P * nb_imp

                        if bt >= cible:
                                S.append(Prix)
                                achat = 1
                                else:
                                achat = 0

                        resultats = bidrequests(
                                identifiant = identifiant,
                                timestamp = env.now,
                                timestamp_string = time,
                                nombre_impressions = nb_imp,
                                prix_impression = P,
                                prix_total = Prix,
```

```
                        achat = achat,
                        budget_depense = sum(S),
                        bt = bt
                )
                data.append(resultats)

                if datetime.strptime(datetime.fromtimestamp(env.now) \
                .strftime("%m-%d-%Y %H:%M:%S"), '%m-%d-%Y %H:%M:%S') \
                .hour != current_hour:
                        i += 1
                        current_hour = datetime.strptime(datetime \
                        .fromtimestamp(env.now).strftime("%m-%d-%Y %H:%M:%S"), \
                        '%m-%d-%Y %H:%M:%S').hour

                try:
                        time_before_next = delai(lam_secondes[i])
                except IndexError:
                        time_before_next = delai(lam_secondes[i-1])

                if rt < time_before_next:
                        print(f"End at {time} !")

                yield env.timeout(time_before_next)

        else:
                setup = True
                current_hour = datetime.strptime(datetime.fromtimestamp( \
                env.now).strftime("%m-%d-%Y %H:%M:%S"), \
                '%m-%d-%Y %H:%M:%S').hour
                time = datetime.fromtimestamp(env.now) \
                .strftime("%m-%d-%Y %H:%M:%S")
                rt = (timestampnow+total_sec(nb_jours))-env.now
                if rt <= 1:
                        print(f"End at {time} !")
                yield env.timeout(1)

nb_days = 1
prix_per_imp = 1
Budget_par_jour = 3000

timestampnow = int(time.time()-3600*15)
bidrequests =  namedtuple(
        "bidrequests",
        (
                "identifiant",
                "timestamp",
```

```
                "timestamp_string",
                "nombre_impressions",
                "prix_impression",
                "prix_total",
                "achat",
                "budget_depense",
                "bt"
        )
)
data = list()
env = simpy.Environment(initial_time=timestampnow)
proc = env.process(open_rtb_pacing(env, prix_per_imp, Budget_par_jour, timestampnow, \
nb_days, 14, bidrequests, data))
env.run(until=timestampnow + total_sec(nb_days))
sauvegarde(data, 'data.csv')
```

Sources

https://www.cs.ubc.ca/~cs532l/gt2/slides/11-4.pdf: To find the Nash equilibrium in first price auction.

https://microeconomics.ca/okan_yilankaya/spa.pdf: Cost entry

Auctions Theory and Practice by Paul Klemperer: To state the Revenue equivalence theorem.