2021-11-25

Real-time bidding campaigns optimization using user profile settings

Luis Miralles-Pechuán
Technological University Dublin

Muhammad Atif Qureshi
Technological University Dublin, muhammadatif.qureshi@tudublin.ie

Brian Mac Namee
University College Dublin, Ireland

Recommended Citation

Miralles-Pechuán, L., Qureshi, M.A. & Namee, B.M. Real-time bidding campaigns optimization using user profile settings. Electron Commer Res (2021). https://doi.org/10.1007/s10660-021-09513-9

Electronic Commerce Research
https://doi.org/10.1007/s10660-021-09513-9

Real-time bidding campaigns optimization using user profile settings

Luis Miralles-Pechuán[1] · **M. Atif Qureshi**[2] · **Brian Mac Namee**[3]

Accepted: 24 September 2021

Abstract

Real-Time bidding is nowadays one of the most promising systems in the online advertising ecosystem. In the presented study, the performance of RTB campaigns is improved by optimising the parameters of the users' profiles and the publishers' websites. Most studies about optimising RTB campaigns are focused on the bidding strategy; estimating the best value for each bid. However, our research is focused on optimising RTB campaigns by finding out configurations that maximise both the number of impressions and the average profitability of the visits. An online campaign configuration generally consists of a set of parameters along with their values such as {Browser = "Chrome", Country = "Germany", Age = "20–40" and Gender = "Woman"}. The experiments show that, when the number of required visits by advertisers is low, it is easy to find configurations with high average profitability, but as the required number of visits increases, the average profitability diminishes. Additionally, configuration optimisation has been combined with other interesting strategies to increase, even more, the campaigns' profitability. In particular, the presented study considers the following complementary strategies to increase profitability: (1) selecting multiple configurations with a small number of visits rather than a unique configuration with a large number of visits, (2) discarding visits according to certain cost and profitability thresholds, (3) analysing a reduced space of the dataset and extrapolating the solution over the whole dataset, and (4) increasing the search space by including solutions below the required number of visits. The developed campaign optimisation methodology could be offered by RTB and other advertising platforms to advertisers to make their campaigns more profitable.

Keywords Real-time bidding · Online campaigns optimization · Online advertising · Dynamic programming

✉ Luis Miralles-Pechuán
luis.miralles@tudublin.ie

Extended author information available on the last page of the article

Published online: 25 November 2021

1 Introduction

Advertising Networks act as an intermediary between advertisers and publishers. The main tasks of ANs include buying and selling impressions, displaying adverts on the publishers' websites, and preventing fraud on the publicity ecosystem [1]. ANs have been a popular choice for some time but, a few years ago, a new sophisticated auction-based model called Real-time bidding (RTB) emerged. RTB offers important advantages over the AN model; it directly connects advertisers with publishers, eliminating intermediaries and their respective commissions, and increasing the cost-benefit ratio of both sides [2].

Real-time bidding is a real-time auction platform where buying and selling online impressions take place instantly via programmable criteria [3]. These bidding and winning processes are executed approximately within the time it takes a user to load a page (100 ms). Additionally, RTB can target audiences based on interests and profiles by examining the HTTP cookies stored on the users' browsers [4].

There are different strategies to optimize online campaigns but some of them are more suitable than others depending on the platform. Optimizing online campaigns has been a very interesting line of research from the beginnings of online advertising. For example, Li Zhiwei et al. [5] proposed a solution to monetize the time it takes a full-resolution image to be displayed to show users an advert stored in a small file related to the image. Spotting this gap and proposing a solution to monetize it, could bring huge benefits to the advertising networks.

In the same line, other researchers realized that live streaming videos were an advertising niche that was not fully optimized [6]. Traditional video algorithms such as pre-roll and context mid-roll advertising worked well with the content on the network that could be fully processed which requires the content to be previously stored. However, for content that had to be analysed in real-time the performance was poor, this fact brought the opportunity for developing a new methodology for monetizing live stream videos called LiveSense. LiveSense implements a deep neural network trained with historical values to display a non-intrusive contextual advert at the right moment increasing the monetizing capacity of live stream videos.

Real-time bidding (RTB) optimization differs from other platforms such as search engine optimization in that advertisers bid based on individual impressions rather than on particular keywords [3]. To optimise an RTB platform, several factors are analysed such as the duration of the campaign, the preferences of each advertiser segment, the competitors' bidding strategies, the reserve price of the publishers, or the number of networks [4].

Predicting the optimal bid price for each impression is one of the most recurring optimization techniques in the literature related to RTB campaigns. It is a very challenging problem because, if the price is too low, it is unlikely that the advertiser gets the bid and if the price is too high, the advertiser may pay in excess [7]. Additionally, advertisers prefer their adverts to be displayed uniformly over time to reach a more diverse audience [8].

One of the main benefits of RTB is that it allows advertisers to target specific audiences who are assumed to become receptive towards the marketing of a

product [9]. This strategy of directing campaigns to a small group of people having common interests is useful and is known as microtargeting [10].

Liu et al. [11] showed that by combining using user features with metadata from web pages, it is possible to estimate which users are more likely to generate a conversion. Additionally, it is also possible to create models to predict the probability of new users accessing new web pages [11]. However, making appropriate matchmaking is not a simple task, and even small improvements, when applied to billions of transactions per day, turns into huge economic benefits [12].

Liu et al. [11] optimizes RTB campaigns through text mining and the selection of attributes from the users and web pages. In their approach, they optimize the online campaign based on the results. The approach detects the profile of the users more willing to convert based on machine learning models and the results of the campaign during its execution. Additionally, it implements text mining and natural language processing techniques.

The main difference between the research of Liu et al. [11] and ours is that they use the metadata from the advertiser's campaigns and apply text mining techniques. They also use the results of the live campaigns to tune some of the parameters of the model. They are also specialized in small campaigns rather than in analyzing lots of data from multiple campaigns. Our approach does not apply text mining and is based purely on the parameters (numerical and categorical) from the users and the websites where the adverts are displayed. Additionally, we apply different approaches to refine and improve parameter optimization.

In this paper, we present a methodology focused on finding parameter configurations that maximise the number of visits and also the average conversion probability for those visits. It is similar to one of our previous publications but this time it is focused on RTB campaigns [10]. Additionally, the parameter configuration technique is combined with other approaches such as multiple configurations, discarding visits based on cost and profitability thresholds, configuration extrapolation, and increasing the search space, which is a natural continuation of the previous paper. The combination of multiple approaches with the configuration optimization increases, even more, the obtained performance.

Our goal consists of detecting configurations (sets of parameters from users and web attributes, along with their values) for suggesting them to advertisers so that they can launch their campaigns more effectively. An example of parameter configuration could be: {Browser = "Chrome", Device OS = "Android", Age = "25–34", Gender = "Male", Time = "10:00–11:00", Country = "UK"}.

The major contributions of the proposed work are:

- Developing a method to recommend advertisers configurations to increase the campaigns' profitability of the advertisers. Our method is focused on optimising the attribute selection rather than optimising the bidding strategy.
- Increasing the campaigns' performance. Our methodology explores all the possible configurations and ranks them according to a defined metric that considers the number of visits that meet the requirements of the advertiser and their average profitability.

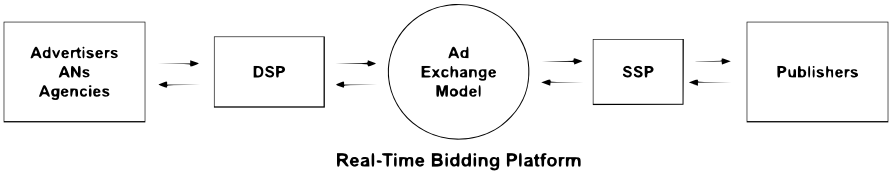

Fig. 1 Structure of the most important modules and roles in a Real-time Bidding platform

- Exploring additional strategies to reach higher levels of profitability. In particular, we explore the following ones: (1) discovering multiple configurations with a fewer number of users. (2) setting an economic and profitability threshold to discard visits. (3) analyzing a small part of the dataset and extrapolating the solution to save computation time. (4) being more flexible by accepting solutions where the number of required visits is lower than the number required by the advertisers.

The rest of the paper is organised as follows: In Sect. 3 a description of the Conversion rate estimator is presented, which calculates the probability of a conversion from a visit. Similarly, in Sect. 4 another important element of the proposed methodology is introduced: the Quality Score Calculator. This module evaluates how good a configuration is based on the number of visits and their profitability. In Sect. 5, experiments are conducted to evaluate the different strategies for improving optimization based on parameters. Finally, Sect. 7 presents the conclusions and some directions for future work.

2 Description of the proposed methodology

RTB allows publishers to connect with multiple ANs and advertising agencies, making it a flexible ecosystem. Additionally, RTB eliminates the need for commissions or fees to AN by allowing advertisers to directly buy impressions from the publisher. Impressions can be bought individually, in a fine-grained fashion, which leads to more effective campaigns [13].

RTB has a hierarchical bidding system where impressions are offered through several subsystems and where the highest price is selected. When a bid is won by an advertiser, the advert is displayed on the publisher's website [3].

As shown in Fig. 1, two modules define the RTB ecosystem: the demand-side platform (DSP), representing the advertisers' interests, and the supply-side platform (SSP), representing the publishers' interests. The DSP manages advertisers and ANs campaigns efficiently, bidding directly on the auctions. To select the best advert, the DSP implements Machine Learning models to estimate the probability of generating a conversion or a click from a given advert [14, 15]. On the other hand, the SSP aims at managing and optimising publishers' web spaces. The SSP distributes the information of a publisher across multiple platforms whenever a user visits a website and selects the most cost-effective advert [16].

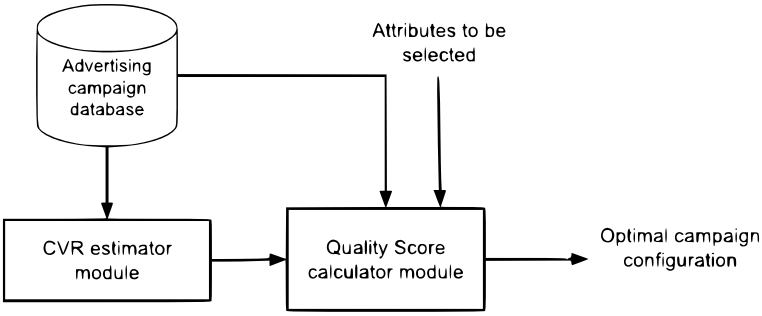

Fig. 2 The methodology used to find out the best campaign configuration

Our methodology falls on the side of the DSP module; the demand side of the platform. Its purpose is to identify configurations to make online campaigns in RTB more effective by selecting the right values for the users and websites parameters [10]. The overall architecture of the proposed methodology is shown in Fig. 2. The most important modules: the *Conversion rate* (CVR) estimator and the *Quality Score* calculator (QSC) are described in detail in the next sections.

3 Description of the conversion rate estimator

In this section, we describe the Conversion Rate Estimator (CVR) estimator module. The CVR module calculates the conversion probability (the probability that a conversion is generated when an advert is displayed to a user) for each visit based on historical data. The CVR module is implemented using a supervised model based on logistic regression.

To build the CVR we used a database of display adverts, where each row represents a displayed advert on the website of a publisher. The dataset contains attributes related to the user, the publisher's web page, the campaign identifier, the indicator of whether the conversion was generated, and the price paid for an impression.

Equation 1 shows the formulation of the profitability of an impression using the CVR and the $Impression_{Price}$. The $Impression_{Price}$ is the price of the impression whose value is extracted from the dataset. We implemented the supervised model to estimate the CVR value proposed in [14, 17]. According to Eq. 1, the profitability is higher for those impressions that have a high probability of generating a conversion (numerator) and a low price (denominator).

$$Impression_{Profitability} = \frac{CVR}{Impression_{Price}} \tag{1}$$

We used a train/test split of 6.3 M and 10 M to build and validate the performance of the model. The dataset has a total of ten columns, out of which nine are categorical (cat1, ..., cat9) and form the input variables of the machine learning model $e_i, i = 1, 2, \ldots 9$, and the final column is *Profitability*, which is the target or the output variable of the model y. The output, y, represents the probability that a conversion (a

purchase, a call, the filling of a form...) takes place. For instance, a predicted value of 0.225 means that the model estimates that the user has 22.5% chance of generating a conversion from the displayed advert. The model is used to expressed p, where $p = P(y|e_i, \forall i = 1, 2, \dots 9)$.

The CVR model is built using a well-known methodology for making predictions based on a Logistic Regression model [18, 19]. We configured the adaptive learning rate value by following the criteria defined by Brendan McMahan et al. [20]. In their investigation, they explain that the value of alpha, the heuristic adaptive learning rate of the model, highly depends on the nature of the dataset and the type of data. In our particular case, we used several parameters and chose the one that gave the best performance. The results and performance of the logistic regression algorithm are presented in Sect. 5.2. To create the model, we used a simple but effective technique called hashing trick. The hashing trick is used to reduce the sparsity of the values of the dataset, not the number of features in the dataset [19].

A typical dataset for representing users' visits has N different values for each category, where N can be bigger than the length of the arrays n and w. The hashing trick divides the values of N by D, where $D = \text{length}(w) = \text{length}(n)$, so the sparsity of the values will be at maximum D. Initially, all of the elements of both vectors are set to zero. Afterwards, the elements of n and w are updated using Eqs. 2 and 3.

$$n[i] = n[i] + 1 \tag{2}$$

$$w[i] = w[i] - \frac{\alpha \times (p - y)}{\sqrt{n[i] + 1}} \tag{3}$$

Once the vectors w and n have been updated, we apply Eq. 4 to estimate using a sigmoid function the output of the model for each impression.

$$p = \frac{1}{1 + exp\left(-\sum_{i=1}^{N} w[f_i]\right)} \tag{4}$$

Where each variable is described as follows:

- i represents the index of the arrays w and n.
- $w[i]$ represents the weight of i-th element.
- $n[i]$ represents the number of times the value of i appears after the application of the hashing trick for the elements of the original vector.
- $p \in [0, 1]$ represents if there was a conversion and it is also the estimated output value.
- α represents the heuristic adaptive learning rate. It is used to optimise the weights of the model. The higher the value of the learning rate, the faster it adapts but it may overshoot the gradient. By contrast, the lower the value, the more time it takes to converge.

Once the model and its variables have been explained, we define algorithm 1, which summarises the implementation for both training and testing of the CVR.

Algorithm 1 Algorithm for training and testing the model to estimate the probability of generating a conversion from each visit.

Require: Data set
Ensure: Prediction Conversion model

$data \leftarrow$ Randomly select 12M samples visits from the original dataset
$data_{hash} \leftarrow$ Apply the hashing trick to $data$
Divide $data_{hash}$ into $training$ and $testing$ sets ▷ 6.3M $training$ and 10M for $testing$
$D \leftarrow 2^{20}$ ▷ Length of w and n
$\alpha \leftarrow 0.1$
$w \leftarrow \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$ of length D
$n \leftarrow \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$ of length D

▷ Training the CTR model
for all $row_k \in training$ **do** ▷ row represents a user visit in which adverts display their adverts
    $s \leftarrow 0$
    for all $f_i \in row_k$ **do**
        $s \leftarrow s + w[f_i + 1]$
    end for
    $p_k \leftarrow 1/(1 + exp(s))$
    for all $f_i \in v_k$ **do**
        $w[f_i] \leftarrow w[f_i] - \alpha(p_k - y_k)/(\sqrt{n[f_i] + 1})$
        $n[f_i] \leftarrow n[f_i] + 1$
    end for
end for

▷ Testing the Conversion model
for all $row_k \in testing$ **do** ▷ v_k is a $testing$ sample
    $s \leftarrow 0$
    for all $f_i \in row_k$ **do**
        $s \leftarrow s + w[f_i + 1]$
    end for
    $p_k \leftarrow 1/(1 + exp(s))$
end for
Compute the accuracy of the model

4 Estimating the quality score of an advert

The second module used in our methodology (see Fig. 2) is the Quality Score Calculator (QSC) and ranks how good a configuration is according to a defined metric called *Quality Score*. The *Quality Score* rewards configurations that presumably have high profitability (sets of visits with a high probability of conversion at a low price) and also guarantee a sufficient number of impressions for the advertiser's campaign. The *Quality Score* is calculated for all the possible configurations without repetition and its main purpose is to score all of the possible configurations (i.e. combinations of different attributes).

The QSC module makes use of all the attributes of the impressions and optimises them to achieve the highest performance. Firstly, QSC explores all the possible configurations with a single attribute, then, it continues with two attributes, then with three, and so on until it finally reaches all the combinations with all the

cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	Profitability
5824233	9312274	3490278	29196072	11409686	1973606	25162884	29196072	29196072	753.02
30763035	9312274	14584482	29196072	11409686	1973606	22644417	29196072	21091111	140.56
138937	**9312274**	10769841	29196072	**5824237**	138937	1795451	**9312274**	15351056	907.46
28928366	26597095	12435261	23549932	5824237	1973606	9180723	29841067	29196072	2308.49
138937	26597094	31616034	29196072	11409684	26597096	4480345	29196072	29196072	175.13
30763035	**9312274**	9107790	29196072	**5824237**	1973606	2687461	**9312274**	21091108	83.71
30763035	9068207	5028397	29196072	32440044	32440041	14074087	29196072	21091108	1438.07
138937	9312274	15403272	29196072	32440042	28928366	8556462	29196072	29196072	1206.20
138937	9312274	4281154	29196072	28928366	29196072	21857352	29196072	29196072	217.03
28928366	26597095	7711526	29196072	3225250	1973606	17737135	29841067	29196072	571.40

Fig. 3 Extracting from the original dataset those visits that fit the following configuration: (columns 2, 5, and 8 with values 9,312,274, 582,437 and 9312274 respectively

attributes. All configurations are ranked according to *Quality Score* metric defined by Eq. 6. Equation 5 shows the formula to calculate the total number of combinations. The higher-ranked configurations scored by the QSC are offered to advertisers to increase the performance of their campaigns.

$$Combinations\,without\,repetitions = \sum_{k=1}^{n} \frac{n!}{k!(n-k)!} \tag{5}$$

To begin calculating the value of a configuration, we first estimate the average profitability. To this end, we calculate the profitability (using Eq. 1) for all the impressions that fit the configuration, and then, we calculate the average. Even though the configurations may have some unprofitable impressions, these impressions are offset by the rest of the impressions making the metric robust. Furthermore, we multiply the $Profitability_{Average}$ by the minimum between the number of impressions that satisfy the configuration and the number of visits required by the advertiser, i.e., $\min(rows(D'), T)$. The T represents the set threshold, and it is used to avoid converging to solutions with an excessive number of visits. We should bear in mind that we are looking for configurations with a sufficient number to cover the number required by the advertiser rather than configurations with the highest possible number of visits.

$$Quality\,Score = Profitability_{Average} \times \min(rows(D'), T) \tag{6}$$

The following lines give an example of how the process of calculating the *Quality Scores* for a combination of attributes works. Let's imagine that we are testing the combination of columns (2, 5, 8). First, we select the second, fifth, and eighth attribute from the dataset. The result will be a subset with three selected attributes from the original dataset. From this subset, we select the unique records (i.e., rows) and calculate the *Quality Scores* for each record. As shown in Fig. 3, the first unique values of the subset are: 9,312,274, 582,437 and 9,312,274 respectively. Then, we proceed with the subset called D', where the second, the fifth and the eighth attribute have values 9,312,274, 582,437 and 9,312,274 respectively, are extracted from the original dataset. For this subset D', where D' $\subseteq$ D, the *Quality Score* is calculated using Eq. 6.

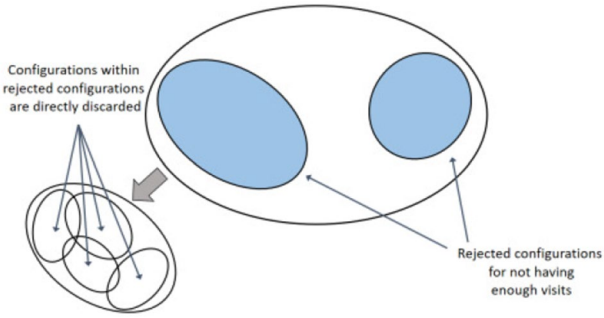

Fig. 4 Using a dynamic programming approach, if a set is inserted in the list of configurations with not enough visits, all the subsets inside of that set will be directly discarded

To speed up the algorithm, we implemented a dynamic programming approach creating a list of all the configurations that do not have enough visits. Then, we added the following condition: if a configuration of a campaign is a subset of a previously discarded configuration, then this new configuration gets rejected as well. As can be seen in Fig. 4 more intuitively, such action was taken on the basis that a subset cannot be greater than the set that contains it. For example, for attributes (1, 3, 4, 7) with values (458, 47, 58, 58) respectively, we check if the list of configurations with not enough visits contains any combination without repetition of these attributes with precisely these same values. To do so, we first check in the list, combinations of one element (attribute 1 with value 458, attribute 3 with value 47, and so on).

Later, we look in the list for combinations of two elements (attribute 1 with value 458 and attribute 3 with value 47, attribute 3 with value 47, attribute 4 with value 58, and so on), and the same technique is applied for the combinations of three elements. This simple improvement significantly reduces the running time to find out the optimal solution and reduces the time to explore all the configurations from 75 h when this improvement is not applied to 193.82 s when it is. Having got to this point, we define algorithm 2 to select the optimal configuration given the requirements of the advertisers.

Algorithm 2 Calculating the best solution.

```
Require: Data, Comb, Limit
Ensure: Tuple
RejectSolution ← { }
SolList ← { }
for all c_i ∈ Comb do                          ▷ Select only columns from combination of the Data
    Data1 ← Subset(Data[, c_i])                ▷ The value of Data1 is a subset deep copy of Data
    Data1 ← Unique (Data1)   ▷ If a configuration is repeated we do not have to calculate the value again
    for all row ∈ Data1 do
        for i ∈ 1:Length(row) do
            Data2 ← Subset(Data1[,Comb[i]] = row[i])
            if ! RejectSolution.exists( c_i, row) then
                if Rows(Data2) ¿= Limit then
                    Profit_Avg ← mean(Data2[,Prof])
                    Size ← Rows(Data2)
                    Fitness ← Profit_Avg × min(Size, Limit)
                    Sol ← Tuple(Fitness, Profit_Avg, Size, c_i, row)
                    SolList ← SolList + Sol
                else
                    RejectSolution.append(Sol[1])          ▷ Add rejected solution to the list
                end if
            end if
        end for
    end for
end for
BestSol ← OrderbyFitness(SolList)                                  ▷ Order by Fitness
return BestSol
```

5 Experiments

In this section, we carry out some experiments to evaluate the yield of the proposed methodology. First, we preprocess the dataset. Secondly, we build the model to estimate the CVR value, then we evaluate the built model using well-known metrics by predicting the CVR of the RTB impressions. Next, we apply the algorithm to find the optimal configurations. Finally, we discuss each of the five experiments to demonstrate the different approaches in the context of attribute selection. All the proposed experiments have in common the following steps: (1) Building the CVR estimation model, (2) Predicting the CVR for all impressions, (3) Generating all possible campaign configurations and calculating the Quality Score, (4) Return the best solution for the given dataset. The steps "Calculate the Quality Score" and "Build the CVR estimation model" have been explained in detail in the previous Sects. 3 and 4).

To carry out the experiments, we used Python 3.6.1 and Anaconda runtime (x86_64). We performed all experiments using a MacBook computer having macOS High Sierra (2.3 GHz Intel Core i5, 8GB 2133 MHz, L2 Cache:256kb, L3: 4MB).

5.1 Data set description

The experiments were conducted using a dataset from the Criteo company [21]. The dataset consists of real data collected from internet traffic for 30 days. The dataset is a subset of the total visits from that period. Each row of the dataset contains information about a displayed advert and the website where it has been displayed. Due to privacy concerns, the dataset has been anonymized.

The dataset is composed of the following fields:

- *Timestamp* Starts at 0 and sorts rows according to the displayed time.
- *UID* Unique code to identify the user.
- *Campaign* Unique code to identify the campaign.
- *Conversion* Uses "1" if a conversion takes place in the following 30 days and "0" in the opposite case.
- *Conversion_timestamp* The exact time at which the conversion derived from the impression took place. If there is no conversion the value is set to "-1".
- *Conversion_id* Represents the code of the conversion derived from the impression. It enables the creation of the timeline (in case they are required).
- *Attribution* A binary field to indicate if the conversion was connected to Criteo ("1" means affirmative).
- *Click* A binary field to indicate if the user clicked the impression ("1" is affirmative)
- *Click_pos* When there are several clicks from a user before a conversion, this value represents the position of the click in the current impression.
- *Click_nb* Represents the number of clicks before the conversion. The same user can click several times in adverts of the same campaign.
- *Cost* A transformed version of the price paid by Criteo.
- *Cost-per-order(CPO)* The amount paid by the advertiser when the conversion is assigned to Criteo.
- *Time_since_last_click* Indicates in seconds the amount of time elapsed from the last click.
- *Cat[1–9]* These nine fields represent features related to the user and the publisher. These fields are encrypted and have been hashed by applying the well-known hashing trick to reduce the dimensions [19]. In addition, these values are used to create a supervised model for predicting the conversion probability.

The original dataset had a total of 16,468,027 instances from different campaigns. First, we split the dataset into the training set (first 6,468,027 rows) and the testing set (last 10 million rows). Then, we built the CVR model using the training set, and we predicted the probability of generating a conversion for the instances of the testing set. In the testing set, there are four campaign identifiers with more than 200 thousand visits (the identifiers of the campaigns are: 10,341,182, 15,398,570, 17,686,799, 30,801,593). We further split each of the four campaigns into two datasets of 100 thousand visits each. There are also nine other campaign identifiers with more than 100 thousand visits: 5061834, 15,184,511, 29,427,842, 18,975,823, 6,686,701, 2,8351,001, 26,852,339, 31,772,643, 497,593. We discarded the rows

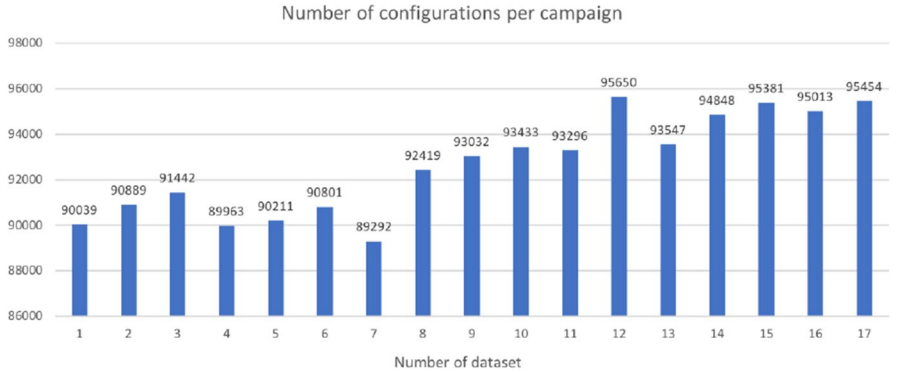

Fig. 5 As we can see, there are a lot of possible combinations for each of the 17 datasets, where each combination is a unique set of values for a group of parameters

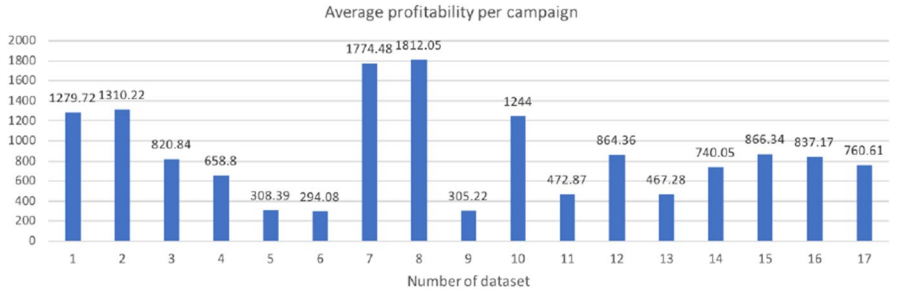

Fig. 6 Average profitability (calculated with the proposed formula) for each of the 17 advertising datasets

above 100 thousand for each of the datasets so that all of them had the same dimensions. Finally, we got a total of 17 datasets of 100 thousand visits each (eight as a result of dividing each of the four campaigns with 200 k into two campaigns of 100 k and the other nine campaigns with 100 k visits). All the experiments were carried out using the 17 datasets. Figure 5 shows the total number of possible configurations in each dataset and Fig. 6 shows the average profitability of each campaign.

5.2 Implementation of the conversion rate Model

To build the CVR estimator module, we selected a dataset from Criteo [21]. The dataset is ordered by time and represents 30 days (see Sect. 5.1 for more details). We used the first 6.46 M rows of the dataset as the training set, and calculated the conversion probability over the testing set that has 10 M rows. For evaluating the performance of our methodology, we included the predictions of the created model as a new column of the testing set and stored it in the advertising campaign dataset module.

Table 1 Metrics of the conversion prediction models using standard metrics

	LR	RF	DT	LR OL	LR HT	LR HT*
RMSE	0.3846	0.4023	0.4201	0.4236	0.4001	0.2027
MAE	0.3079	0.3383	0.3596	0.3824	0.3429	0.0807
Sensitivity	0.6904	0.7225	0.6269	0.6964	0.7189	0.9523
Specificity	0.8014	0.7739	0.8268	0.7838	0.7914	0.5821
Precision	0.8014	0.7739	0.8268	0.7838	0.7914	0.9985
False pos rate	0.1986	0.2261	0.1732	0.2162	0.2086	0.4179
False neg rate	0.3096	0.2775	0.3731	0.3036	0.2811	0.0477
Accuracy	0.7960	0.7714	0.8171	0.7796	0.7878	0.9510
F1 score	0.7418	0.7473	0.7131	0.7375	0.7534	0.9748

*This model was trained with 6.46 M rows and is the one with better performance

But before selecting a classifier we performed an experiment using some of the well-known methods in the literature. Since the number of visits that ended up in conversions was 4.8% and we did not want our models to be biased, we randomly selected 25.000 visits with conversions and 25.000 visits without conversions for creating the first five models. In Table 1 can be seen that the performance of the models according to the RMSE for the models trained with the reduce dataset (Logistic regression, Random Forest, Decision Trees and Logistic regression with Online Learning, and Logistic Regression using the hashing trick) was not very high. The models were implemented using the python library "scikit-learn 0.24.2" with the following parameters: For logistic regression we used a Stochastic Gradient Descent with the function SGDClassifier (loss='log', penalty=None, fit_intercept=True, learning_rate='constant', eta0=0.01), for Random Forests we used the function RandomForestClassifier (n_estimators=100, criterion='gini', min_samples_split=30) and for the Decision trees we used the function DecisionTreeClassifier (criterion='gini', min_samples_split =30).

Those methods were not very accurate and they require a lot of time and computer resources to be trained, hence, we tried a more suitable solution called logistic regression using adapting learning and the hashing trick in the literature but using the whole dataset [18–20]. In this investigation, we are mainly interested in predicting the probability of generating a conversion rather than predicting whether or not a conversion will take place. Therefore we find the last algorithm (LG HT) trained with the complete training set (around 6.4 M) much more suitable for performing the experiments.

As described in Fig. 2, we need a predictive CVR model to estimate the probability of getting a conversion from a visit. To implement the logistic regression model using the hashing trick, we converted all features from strings to integer values and applied modulus 2^{20} to calculate the hashing table. Later, we applied the logistic regression method to build the prediction conversion model with a *learning rate* $\alpha =$

0.1 and $D = 2^{20}$ as the length of arrays n and w. This methodology is inspired by the popular method Vowpal Wabbit[1] widely used for training accurate models without needing powerful computing resources. The Vowpal Wabbit method has multiple version and has inspired several influential papers [22].

5.3 Implementation of the algorithm to find the best solution

The main goal of the algorithm is to explore all the possible attribute combinations and select the unique values for each combination. For example, if the attributes (1, 5) have the values (85, 58), (7714, 424), (596, 3458), (85, 58). Then, the last one will not be evaluated since it is the same as the first one. For each unique value, we select from the dataset all the rows with the value 85 in attribute 1 and the value 58 in attribute 5. For that subset of the dataset, we calculate the average profitability and the number of rows from the dataset that match that configuration. The idea is to evaluate all the possible combinations and store the ones with higher values.

5.4 Parameters of the experiments

We gather the following information for each of the configurations:

- *Average profitability* Indicates the average profitability of the selected impressions in the configuration campaign.
- *Data set size* Indicates the number of rows of the dataset.
- *No Rows* Indicates the number of rows of the best configuration, which is the configuration with the highest value.
- *Selected columns* Indicates the selected attributes by the best configuration.
- *Time (Sec)* required time to find the best configuration.
- *Values for selected columns* In addition to which columns/attributes have to be selected, we are also interested in the values that optimise those parameters.
- *Quality Score* As indicated previously, this metric is used to show how good a configuration is, given the number of visits required by the user.
- *Data set size* The remaining size of the dataset after subtracting the selected configurations.

6 Results and discussion

To measure the improvement of the proposed method, we have conducted five different experiments based on parameter optimisation combined with other ideas as discussed in the previous section. We have performed the experiments using 17 datasets, and the obtained results are the summation of the average profitability of

[1] Vowpal Wabbit is a popular open-source library developed by Yahoo and later acquired by Microsoft research. For more information: https://github.com/VowpalWabbit/vowpal_wabbit/wiki.

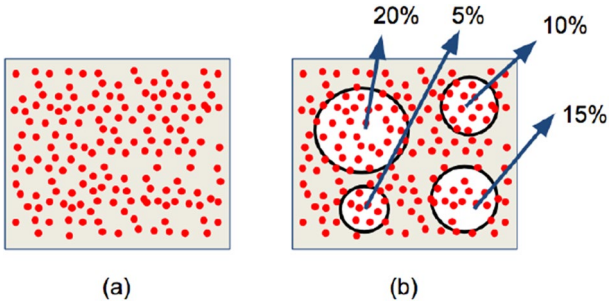

Fig. 7 Each configuration has two parameters: the average profitability and the number of visits matching the parameter configuration. The first picture **a** shows all the visits and the second, **b** the required visits by the advertisers

all the datasets. The designed algorithm collects some parameters that indicate the performance of the campaign. This table shows the collected information in experiment II for the first of the 17 datasets. Other variables such as *Quality Score* or the remaining size of the dataset can be calculated from these values.

6.1 Experiment I

As shown in Fig. 7, the original datasets have 100 K samples but we are trying to select small groups with higher profitability. In this first experiment, we selected the best configuration from the dataset with at least 5 k (5% of 100 k) visits. Then, we repeated the same process increasing the limit of the selected visits by adding 5 k until reaching 50 k. If advertisers require a small number of visits, the profitability will be very high, but, to perform a successful campaign it is generally required to display a sufficient number of adverts. The key aspect of this experiment is to compare our approach with some baselines and to visualise how profitability decreases as the number of demanded visits increases.

In this first experiment, we compare the performance of our proposed system with two other methods in the literature, "Optimal bid price" and "Reinforcement Learning". We also compare it with the best possible solution called the "Optimal profitability". The "Optimal Profitability" is the summation of the average profitability of the visits with the highest profitability for each of the 17 campaigns for the required number of visits. They represent the limit of how good a campaign can be in terms of profitability. As can be seen in Fig. 7, the higher the required number of visits, the lower the profitability. In other words, there is a negative correlation between the number of required visits and the profitability.

The method called "Optimal bid price" is a common strategy used as a baseline in which the bidding value is constant throughout the whole campaign [16]. The shown results are obtained by calculating the price of the bid that maximizes the average profitability for each campaign. If the number of visits is lower than the required number by the advertiser then we have to increase the price. If the number of visits is higher then we select randomly the value set by the advertiser.

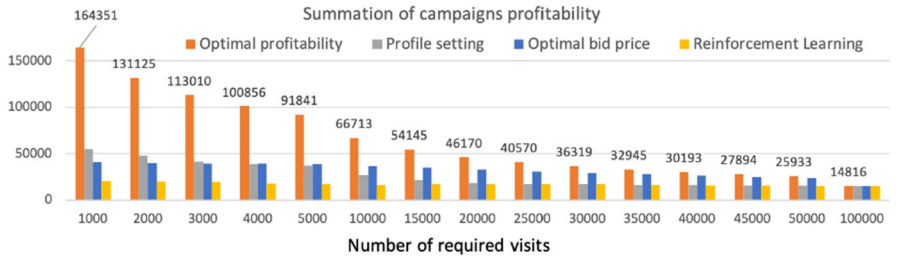

Fig. 8 Experiment I confirms that as the number of required visits increases, the average profitability declines. The graph shows the average profitability of the 17 campaigns as well as the average percentage increment compared to the average of all the visits

We started bidding from the lowest price and we increased the value using small intervals, where each interval was the result of dividing the range between the highest and the lowest campaign by 1000, until the highest price. Then, we selected the value that maximized the profitability for that campaign.

The method called "Reinforcement Learning" consists of an agent that decides whether or not to buy depending on the price. The agent is rewarded using the profitability in such a way that it tries to maximize the average reward the agent obtains covering the number of visits required by the advertiser [23]. The agent has two possible actions: buying and not buying. We used the Q-Learning algorithm with the following parameters: gamma = 0.95, epsilon = 1, epsilon decay = 0.995. To approximate the Q-Table we used a Deep Learning Network based in a densely connected sequential multi-layer perceptron [24] with four layers with 64, 128, 128, and 8 nodes respectively. The ANN utilizes a linear function and an Adam optimizer with 0.001 with mean-squared error as the loss function. First, we trained the model with 100 k samples from the training for 200 episodes. Then we evaluated it for the selected campaigns. Reinforcement learning works well for some environments but one of the downsides is that it is very difficult to find the optimal parameters for making the agent converge to the optimal solution.

Our approach improves the campaign' performance by optimising the parameter configuration. Generally, the more specific the campaigns the higher the yield, but, on the other hand, the number of visits that match the configuration is lower, so we have fewer visits. As seen in Fig. 8, the experiment confirms that it is possible to enhance campaigns performance through parameter configuration. As you can see our approach "Profile settings" works better than the "Optimal bidding price" for a small number of visits than 4000. Nevertheless, it is worthwhile noticing that it is not possible to calculate the better price in real-time as it constantly fluctuates and also the advertiser will have to spend a lot of money trying different prices to calculate the optimal price for a given time.

We can also see in Fig. 8, that the performance of "Profile Setting" and "Reinforcement Learning" are similar after 20 K visits. The reason why the performance of "Profile Settings" is low after such a number of visits is that it is very difficult to find configurations (combinations of parameters and their possible

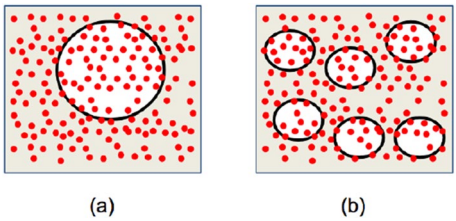

Fig. 9 Selecting a large group as in **a** is computationally less expensive than selecting small groups as in **b**, but the small-groups approach gives higher profitability

values) for such a high number of users. Bear in mind that 20 k represents 20% of the users and that the bigger the size of the group, the more difficult it becomes to find configurations with high performance.

On the other hand, our implementation of "Reinforcement Learning" based on the algorithm Q-Learning is more regular across the different number of visits but the performance is low compared to other methods. RL had to consider the required number of visits which made the problem more challenging for the agent. This could also be due to changes in the environment (the price and features of the adverts) between the 17 different campaigns. As aforementioned, the main drawback of RL methods is that because the agent learns from its own experience, it can take hundreds of simulations to converge to the optional policy. Additionally, RL has high sensitivity towards parameter values and defining the reward function has to be customized for each specific problem. That is why RL generally requires very powerful computational resources compared to the "Profile Settings" approach. Recent implementations use more information like the actions of other users bidding in the platform, and implement more elaborated approaches that require training two models such as Multi-Objective Actor-Critic [25] or using multiple agents such as Distributed Coordinated Multi-Agent Bidding (DCMAB) [26].

6.2 Experiment II

The motivation of the second experiment is to discover whether it is better to optimise multiple campaigns with a small number of visits, or to optimise one single campaign with a big number of visits, where the summation of the visits of the small campaigns is similar to that of the big campaign. In Fig. 9 the two approaches can be seen graphically. Optimising more campaigns has a higher computational cost, but a better solution may pay off these costs. To avoid the problem of the intersection of visits, which occurs when two or more campaigns share the same target, we perform the experiments sequentially so that when the first campaign configuration is selected, all the visits that match with such configuration are removed from the dataset. The same process is repeated until all the campaigns are chosen.

The experiment is performed for the following number of visits required by the advertiser: 5 k, 10 k, 15 k, 20 k, 25 k and 30 k. Finally, we compare the performance of selecting only one campaign with multiple campaigns with configurations with at least 1 k and 2.5 k visits respectively. For instance, we compare a configuration that

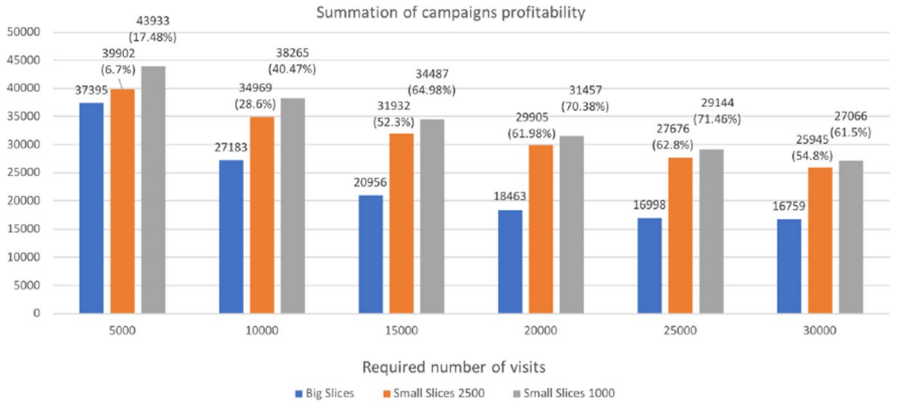

Fig. 10 The technique used in Experiment II consists of selecting multiple configurations with small visits. It performs better since it is easier to find small groups with high performance than a single large group

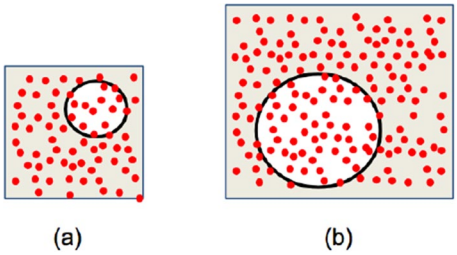

Fig. 11 If it is possible to extrapolate a good solution from a slice of the campaign (**a**) instead of using the whole dataset (**b**), then profitable configurations can be obtained at a lower computational cost

matches at least 5 k visits with the summation of two configurations with at least 2.5 k, and with five configurations with at least 1k visits.

This experiment is conducted to find out if the technique of selecting a group of configurations with a small number of visits has a higher performance than that of selecting a single configuration with a large number of visits, where the combination of the visits of the group has a similar number of visits to the single configuration. Figure 10 clearly shows that when the slides are of 1000 visits, the increase in profitability is minimal. Additionally, it is shown that, as the number of required visits increases, the performance when applying this technique has much better results. However, when the slides are of 2500 visits, the improvement is very small and as the number of required visits increases, the improvement decreases. The drawback of this strategy is that it requires a higher processing cost. For example, when searching for sets of 10,000 items, the proposed strategy will search for four sets of 2500, which requires executing the algorithm four times instead of one. However, the improvement of the profitability confirms that it is worthwhile applying it.

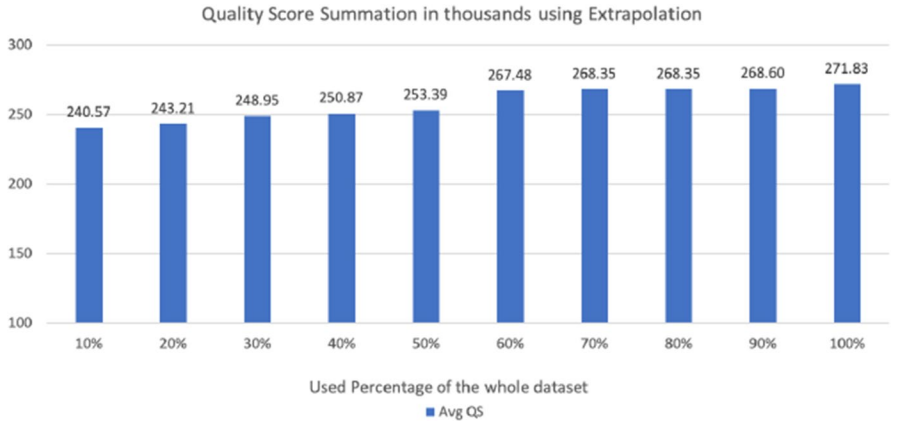

Fig. 12 Extrapolating configurations from a small sample, as shown in Experiment III, is a very effective technique to find good configurations reducing computational costs

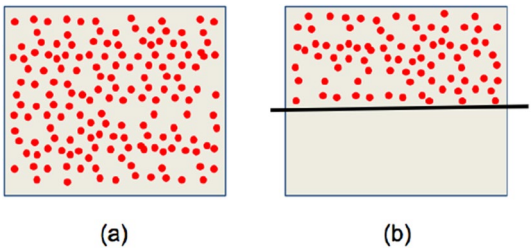

Fig. 13 Discarding visits over a certain price limit or below a certain profitability value as in **b**, can produce better results than when there are no discarded visits as in **a**

6.3 Experiment III

In the third experiment, we investigate the effect of finding a profitable configuration from a subset of the dataset and observe if the same configuration is still performing well over the entire dataset. If the finding is positive, it will have a substantial advantage in terms of computational costs (time and memory). Figure 11 shows the impact of configurations extracted from the subset and extrapolated over the whole dataset. In this experiment, we extrapolate from 10%, 20%, 30%, and so on until we reach 100 % of the dataset.

As shown in Fig. 12, extrapolating is an interesting way to save campaign expenses (as discussed in the previous section). The experiments show that by extrapolating from the first 10% of the campaigns visits, it is possible to find out very profitable configurations. In particular, these configurations have an average fitness value only 8% lower than the optimal solution. This finding is relevant in terms of reducing economic resources since a configuration that performs well today will probably do so in the following days.

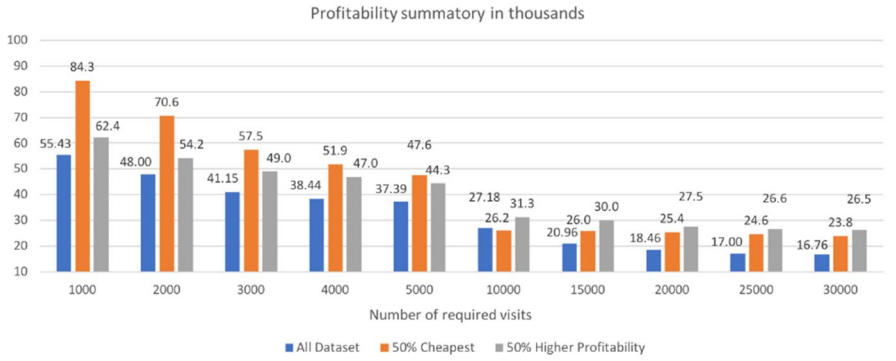

Fig. 14 Experiment IV confirms that discarding visits below a certain price or profitability significantly improves the average performance of the campaigns with the 17 datasets

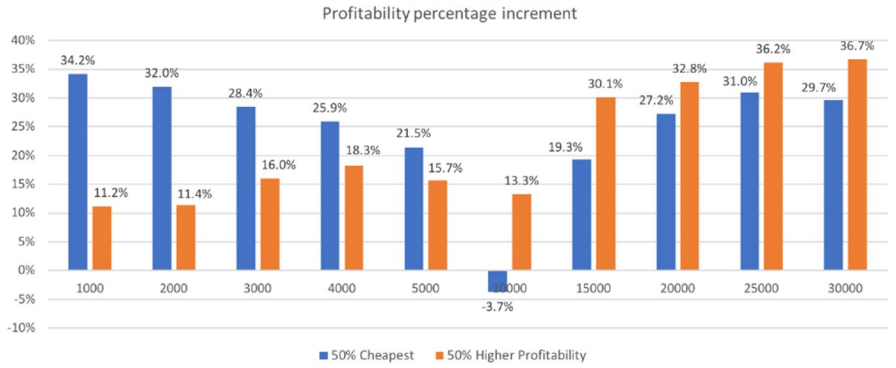

Fig. 15 Comparison of increments improvements with each kind of threshold. The performance of the approach based on price is better until reaching 10 k visits. From that point on, it is better to use the threshold based on profitability

6.4 Experiment IV

In the fourth experiment, we evaluate a strategy that combines the optimisation of parameters with an approach based on setting an economic threshold. In the proposed approach, we discard all visits that are above a particular economic value and then, we apply a parameter optimisation over the remaining visits as indicated in Fig. 13. The threshold is calculated as the value that divides the dataset into two equivalent halves. Then, we discard the half of the dataset below the threshold. In the experiment, we use two different kinds of thresholds, the first one based on the economic cost of the visits, and the second one on the expected profitability of the visits. We conducted the experiment with groups of visits as 5 k, 10 k, 15 k,...,50 k.

The results of the experiment confirm that combining the selection of parameters with the application of a threshold (either by profitability or by price) increases the

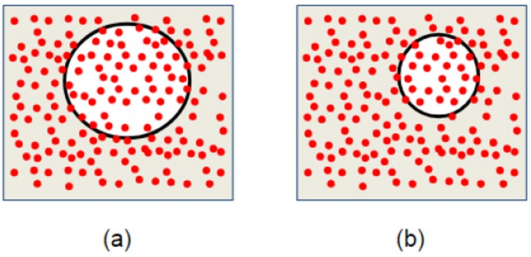

Fig. 16 Increasing the search space by allowing configurations with a smaller number of visits (**a**) compared instead of having the restriction (**b**), allows detecting configurations with higher Quality Score

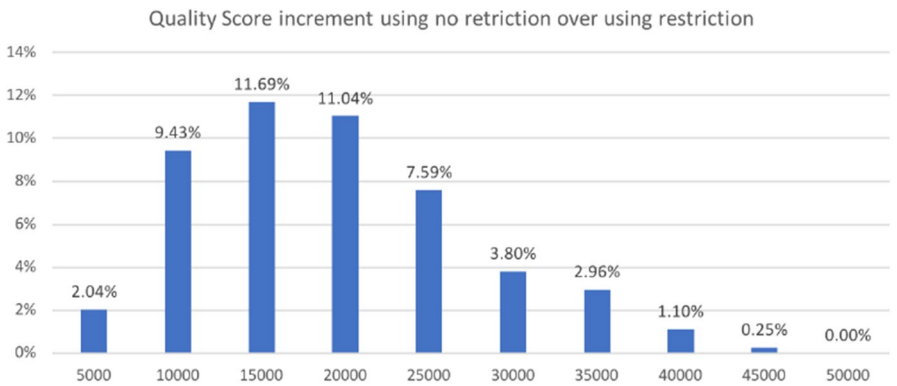

Fig. 17 Increment of not having restriction versus having a restriction. It can be seen that configurations in the search space below the threshold can be very cost efficient, low cost, and with a number of visits close to the required number

efficiency of the campaigns. Figure 14 shows that using the price as a threshold is better than using a threshold based on profitability until reaching the point of the 10 k visits. On the other hand, Fig. 15 shows the increments of improvement with each threshold. It is worthwhile highlighting that when the number of required visits is higher than 10 k, it is better to use the threshold based on profitability. We also see that as the number of required visits increases, the performance of the three approaches tends to equalise.

6.5 Experiment V

One of the restrictions in the previous experiments was that all configurations with a number of visits lower than the required number were rejected. But, it could be the case that a campaign is offered with a slightly lower number of visits than requested but with higher average profitability, so that it satisfies the advertiser. Figure 16 shows a motivating example, where the advertiser wants 15 K visits (a), and, we propose a solution of 13 K (b), but, with very high performance. It is quite likely that the advertiser would accept it. Bear in mind that configurations with a low number

of visits will still be penalised (but not rejected) as the number of visits is a factor to calculate how good a configuration is.

Increasing the search space by allowing the algorithm to explore configurations with a number of visits below the limit improvement of the campaign performance and decreases the economic costs of the campaign. However, it requires a higher computational cost since the search space for the solutions is larger. As can be seen in Fig. 17, this technique brings a substantial improvement within the range of 10 k to 25 k, but after 40 k the improvement becomes marginal.

6.6 Performance comparison with state-of-the-art methods

Next, we compare the performance of our methodology with some state-of-the-art methods. Although we have not found other research focused on increasing the campaign performance through parameter optimisation, we would like to highlight some recent publications aimed at improving the performance of RTB campaigns.

Zhang et al. [27] affirm that there are not many publications related to RTB because until 2013 (the year in which the Chinese RTB company iPinyou decided to make public some of its campaigns) there were no databases related to RTB campaigns. This author performed a comparative study on the performance of bidding strategies applied to RTB. The results are shown according to a key performance indicator (KPI) defined as the summation of clicks and visits. Results indicate that the algorithm called LIN (Linear-form bidding of predicted CTR) improves the performance of the bidding model below max (MCPC) by 204.13% when using 1/32 of the budget, 24% when using 1/8 of the budget, and 8.5% when using 1/2 of the budget.

Similarly, Zhang et al. [27] developed an algorithm to increase the number of clicks in the campaigns. In their research, they compare the performance of a new method called ORTB (Optimal real-time bidding) with LIN. The average increment of the summation of clicks for the nine campaigns using different budgets is 84.3% when the budget is 1/64, 28.61% when the budget is 1/32, 16.14% for 1/16, 9.19% for 1/8, 4.43% for 1/4, and 1.94% for 1/2 [16].

Another interesting publication along these lines is that of Lee et al. [8] in which an adaptive algorithm to select quality impressions is presented. The algorithm takes into account the performance of previously displayed impressions while it distributes the budget evenly overtime to reach the widest possible audience. In CPM flat campaigns, the average CTR increment for seven campaigns was 123.7%. The performance of their methodology was also evaluated using ten dynamic CPM campaigns and the increment in performance concerning conversions (CPA) and the number of clicks (CPC) was 30.9% and 19.0% respectively.

Also relevant in this context is the publication of Do et al. [28] in which they improved the performance of the RTB through a Constrained Markov Decision Process (CMDP) based on a reinforcement learning framework. A distributed representation model is used to estimate the CTR value where the estimated CTR is the state, and the price of the action and the clicks are the reward. We see that the CMDP performance in terms of the number of clicks for the sum of ten campaigns is 12.6%

better and the expected cost-per-click (eCPC) is 9.13% lower than in *Sparse Binary* which was considered a baseline.

Finally, Shioji et al. [29] used neural embedding strategies like word2vec to improve the estimation of the users' response to the displayed adverts (a.k.a. CTR). The Word2vec technique is used to learn distributed representations from the internet browser history of the users. Their approach can improve the accuracy of the CTR estimations results as follows: 4.90% for 0.3 k, 3.39% for 1 k, 2.87% for 10 k, and 1.38% for 100 k, where the first number of the tuple indicates the side of the training data and the second the AUC improvement.

7 Conclusions and future work

RTB platforms are becoming a very beneficial advertising model for publishers and advertisers. It is no wonder that the estimated volume of impressions managed by RTB networks will continue to increase in the coming years. It does not seem unreasonable to say that shortly RTB can replace advertising network platforms or at least take significant market share.

In this paper, we propose a novel methodology based on parameter configuration to find profitable campaigns for advertisers in an automatic fashion. In this sense, we think that the proposed approach is interesting because, to our knowledge, it covers a gap in RTB campaign optimisation research.

The developed experiments prove that the presented methodology improves the results of RTB campaigns in a substantial way. Moreover, the combination of parameter optimisation with other approaches such as small campaign selection, setting a threshold, configuration extrapolation, or increasing the solution search space, improves the obtained results even more. However, these results may vary depending on several variables of a campaign such as a target audience, the moment when it is launched, or the behaviour of the rest of the competitors.

Implementing the methodology would enable RTB platforms as well as advertising networks that manage third-party campaigns to provide advertisers with better configurations for their campaigns. Profitable campaigns will eventually boost the performance of advertising companies, making RTB a more attractive platform, which in turn will make RTB advertisers more willing to launch more campaigns.

It could seem that, as the selected number of configurations increases, the gain of the average profit by campaign becomes trivial. But, here, we argue that, first, in RTB many ad networks coexist with their advertisers, therefore, when a platform decides not to bid on a particular impression, it does not imply that it is lost, but instead, that it will be disputed among the rest of the advertisers. Additionally, it may be the case that some impressions may have a low probability conversion for an advertiser but a high probability for other advertisers as it depends on the nature of the advert. Secondly, there are two types of campaigns: those based on branding and others based on performance. Impressions not valuable from a performance-based perspective could be valuable for branding-based campaigns; where the goal is to increase the brand value instead of looking for profits in the short term.

For future work, it could be a good idea to combine several techniques. For example, setting an economic threshold with small campaign selection and increasing the search space. The new methodology could be tested in different scenarios with other payment methods such as pay-per-click or pay-per-acquisition. Using algorithms to find a suboptimal solution but in less time could also be a good starting point. To this end, multi-objective evolutionary algorithms such as NSGA-II (Non-dominated Sorting Genetic Algorithm) could be a good solution.

Funding Open Access funding provided by the IReL Consortium.

References

1. Kshetri, N., & Voas, J. (2019). Online advertising fraud. *Computer, 52*(1), 58–61.
2. Ren, K., Qin, J., Zheng, L., Yang, Z., Zhang, W., & Yu, Y. (2019). Deep landscape forecasting for real-time bidding advertising. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, (pp. 363–372).
3. Wang, J., Zhang, W., Yuan, S., et al. (2017). Display advertising with real-time bidding (rtb) and behavioural targeting. *Foundations and Trends® in Information Retrieval,11*(4–5), 297–435.
4. Yuan, Y., Wang, F., Li, J., & Qin, R. (2014). A survey on real time bidding advertising. In *2014 IEEE international conference on service operations and logistics, and informatics (SOLI)*, (pp. 418–423). IEEE.
5. Li, Z., Zhang, L., & Ma, W.-Y. (2008). Delivering online advertisements inside images. In *Proceedings of the 16th ACM International conference on multimedia*, (pp. 1051–1060).
6. Chen, X., Nguyen, T. V., Shen, Z., & Kankanhalli, M. (2019). Livesense: Contextual advertising in live streaming videos. In *Proceedings of the 27th ACM international conference on multimedia*, (pp. 392–400).
7. Yuan, S., Wang, J., Chen, B., Mason, P., & Seljan, S. (2014). An empirical study of reserve price optimisation in real-time bidding. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 1897–1906). ACM.
8. Lee, K.-C., Jalali, A., & Dasdan, A. (2013). Real time bid optimization with smooth budget delivery in online advertising. In *Proceedings of the seventh international workshop on data mining for online advertising*, (p. 1). ACM.
9. Goldfarb, A., & Tucker, C. (2011). Online display advertising: Targeting and obtrusiveness. *Marketing Science, 30*(3), 389–404.
10. Miralles-Pechuán, L., Ponce, H., & Martínez-Villaseñor, L. (2018). A novel methodology for optimizing display advertising campaigns using genetic algorithms. *Electronic Commerce Research and Applications, 27,* 39–51.
11. Liu, Y., Pandey, S., Agarwal, D., & Josifovski, V. (2012). Finding the right consumer: optimizing for conversion in display advertising campaigns. In *Proceedings of the fifth ACM international conference on web search and data mining*, (pp. 473–482). ACM.
12. Miralles-Pechuán, L., Ponce, H., & Martínez-Villaseñor, L. (2020). A 2020 perspective on a novel methodology for optimizing display advertising campaigns using genetic algorithms. *Electronic Commerce Research and Applications, 40,* 100953.

13. Yuan, S., Wang, J., & Zhao, X. (2013). Real-time bidding for online advertising: Measurement and analysis. In *Proceedings of the seventh international workshop on data mining for online advertising*, (p. 3). ACM.
14. Lee, K.-c., Orten, B., Dasdan, A., & Li, W. (2012). Estimating conversion rate in display advertising from past erformance data. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 768–776). ACM.
15. Miralles-Pechuán, L., Rosso, D., Jiménez, F., & García, J. M. (2017). A methodology based on deep learning for advert value calculation in cpm, cpc and cpa networks. *Soft Computing, 21*(3), 651–665.
16. Zhang, W., Yuan, S., & Wang, J. (2014). Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 1077–1086). ACM.
17. Ahmed, A., Das, A., & Smola, A. J. (2014). Scalable hierarchical multitask learning algorithms for conversion optimization in display advertising. In *Proceedings of the 7th ACM international conference on web search and data mining*, (pp. 153–162). ACM.
18. Chapelle, O., Manavoglu, E., & Rosales, R. (2015). Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST), 5*(4), 61.
19. Li, P., Shrivastava, A., Moore, J. L., & König, A. C. (2011). Hashing algorithms for large-scale learning. In *Advances in neural information processing systems*, (pp. 2672–2680).
20. McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., & Golovin, D., et al. (2013). Ad click prediction: A view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining*, (pp. 1222–1230). ACM.
21. Diemert, E., Meynet, J., Galland, P., & Lefortier, D. (2017). Attribution modeling increases efficiency of bidding in display advertising. In *Proceedings of the ADKDD'17*, (p. 2). ACM.
22. Juan, Y., Zhuang, Y., Chin, W.-S., & Lin, C.-J. (2016). Field-aware factorization machines for ctr prediction. In *Proceedings of the 10th ACM conference on recommender systems*, (pp. 43–50).
23. Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., & Guo, D. (2017). Real-time bidding by reinforcement learning in display advertising. In *Proceedings of the tenth ACM international conference on web search and data mining*, (pp. 661–670). ACM.
24. Miralles-Pechuán, L., Jiménez, F., Ponce, H., & Martínez-Villaseñor, L. (2020). A methodology based on deep q-learning/genetic algorithms for optimizing covid-19 pandemic government actions. In *Proceedings of the 29th ACM international conference on information & knowledge management*, (pp. 1135–1144).
25. Yang, C., Lu, J., Gao, X., Liu, H., Chen, Q., Liu, G., & Chen, G. (2020). Motiac: Multi-objective actor-critics for real-time bidding. Preprint arXiv:2002.07408
26. Jin, J., Song, C., Li, H., Gai, K., Wang, J., & Zhang, W. (2018). Real-time bidding with multi-agent reinforcement learning in display advertising. In *Proceedings of the 27th ACM international conference on information and knowledge management*, (pp. 2193–2201).
27. Zhang, W., Yuan, S., Wang, J., & Shen, X. (2014). Real-time bidding benchmarking with ipinyou dataset. Preprint arXiv:1407.7073
28. Du, M., Sassioui, R., Varisteas, G., Brorsson, M., & Cherkaoui, O., et al. (2017). Improving real-time bidding using a constrained markov decision process. In *International conference on advanced data mining and applications*, (pp. 711–726). Springer.
29. Shioji, E., & Arai, M. (2017). Neural feature embedding for user response prediction in real-time bidding (rtb). Preprint arXiv:1702.00855

Authors and Affiliations

Luis Miralles-Pechuán[1] · M. Atif Qureshi[2] · Brian Mac Namee[3]

M. Atif Qureshi
muhammadatif.qureshi@adaptcentre.ie

Brian Mac Namee
brian.macnamee@ucd.ie

[1] School of Computer Science, Technological University Dublin, Central Quad, Grangegorman Lower, Dublin D07 ADY7, Ireland

[2] ADAPT Centre, School of Marketing, Technological University Dublin, Aungier Street, Dublin D02 HW71, Ireland

[3] Insight Centre for Data Analytics, University College Dublin, Belfield, Dublin D04 V1W8, Ireland