The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

A Combinatorial-Bandit Algorithm for the Online Joint Bid / Budget Optimization of Pay-per-Click Advertising Campaigns

Alessandro Nuara, Francesco Trovò, Nicola Gatti, Marcello Restelli
Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano, Milano, Italy
{alessandro.nuara, francesco1.trovo, nicola.gatti, marcello.restelli}@polimi.it

Abstract

Pay-per-click advertising includes various formats (e.g., search, contextual, and social) with a total investment of more than 140 billion USD per year. An advertising campaign is composed of some subcampaigns—each with a different ad—and a cumulative daily budget. The allocation of the ads is ruled exploiting auction mechanisms. In this paper, we propose, for the first time to the best of our knowledge, an algorithm for the online joint bid/budget optimization of pay-per-click multi-channel advertising campaigns. We formulate the optimization problem as a *combinatorial bandit* problem, in which we use Gaussian Processes to estimate stochastic functions, Bayesian bandit techniques to address the exploration/exploitation problem, and a dynamic programming technique to solve a variation of the Multiple-Choice Knapsack problem. We experimentally evaluate our algorithm both in simulation—using a synthetic setting generated a Yahoo! dataset—and in a real-world application for two months.

Introduction

Online advertising has been given wide attention from the scientific world as well as from industry. in 2016 more than 72 billion USD has been spent in search advertising (IAB 2016), which represents about 50% of the total market. The development of automatic techniques is crucial both for the publishers and the advertisers, and artificial intelligence can play a prominent role in this context. In this paper, we focus on *pay-per-click* advertising—including different formats, e.g., *search*, *contextual*, and *social*—in which an advertiser pays only once a user has clicked her ad.

An *advertising campaign* is characterized by a set of *subcampaigns*—each with a potentially different pair ad/targeting—and by a cumulative daily budget. Remarkably, a campaign may include subcampaigns on different channels, e.g., Google, Bing, Facebook. In pay-per-click advertising, to get an ad impressed, the advertisers take part in an auction, specifying a bid and a daily budget for each subcampaign (Qin, Chen, and Liu 2015). The advertisers' goal is to select these variables to maximize the expected revenue they get from the advertising campaign. The *Generalized Second Price auction* (GSP) is used for search advertising (King, Atkins, and Schwarz 2007). This auction is not *truthful*—i.e., the best bid for an advertiser may be different from the actual value per click—, and learning algorithms are commonly used to learn the optimal bids. The *Vicrey-Clarke-Groves* mechanism (VCG) is instead used for contextual and social advertising (Varian and Harris 2014; Gatti et al. 2015). This auction is truthful only in the unrealistic case in which the daily budget is unlimited.[1] Thus, each advertiser needs to find the best bid and budget values in an online fashion, and this problem is currently open in the literature. In the present paper, we provide a novel algorithm, based on *combinatorial bandit* techniques (Chen, Wang, and Yuan 2013), capable of automating this task.

Related works. Only a few works in the algorithmic economic literature tackle the campaign advertising problem by combining learning and optimization techniques. More specifically, Zhang *et al.* (2012) study a scenario similar to the one we analyze in this paper. The authors take into account the problem of the joint bid/budget optimization of subcampaigns in an offline fashion. However, this algorithm suffers from some drawbacks. More precisely, the model of each subcampaign requires a huge number of parameters s.t. even achieving rough estimates requires a considerable amount of data, usually available only after the system has been running for several months. Furthermore, some parameters (e.g., the position of the ad for each impression and click) cannot be observed by an advertiser, not allowing the employment of the model in practice. Finally, the optimization problem formulated therein is nonlinear and finding a good-quality approximate solution requires long computational time. Thomaidou, Liakopoulos, and Vazirgiannis (2014) separate the optimization of the bid from that one of the budget, using a genetic algorithm to optimize the budget and subsequently applying some bidding strategies. Markakis and Telelis (2010) study the convergence of some bidding strategies in a single-subcampaign scenario.

Online learning results are known only for the restricted cases with a single subcampaign and a budget constraint over all the length of the campaign without temporal deadlines.[2] Ding *et al.* (2013) and Xia *et al.* (2015) work on a

[1] Notably, the value per click is unknown at the setup of the campaign, not allowing an advertiser to bid truthfully.

[2] This last assumption rarely holds in real-world applications where, instead, results are expected by a given deadline.

finite number of bid values and exploit a multi-armed bandit approach. Trovò *et al.* (2016) work on a continuous space of bids and show that assuring worst-case guarantees leads to the worsening of the average-case performance.

Less related works concern daily budget optimization (Xu et al. 2015; Italia et al. 2017), bidding strategies in display advertising (Wang, Zhang, and Yuan 2016; Weinan et al. 2016; Zhang, Yuan, and Wang 2014; Lee, Jalali, and Dasdan 2013) and video advertising (Geyik et al. 2016). Finally, some works deal with the attribution problem of the conversions in display advertising (Geyik, A-Saxena, and Dasdan 2014; Kireyev, Pauwels, and Gupta 2016).

Original contributions. We formulate the optimization problem as a combinatorial-bandit problem (Chen, Wang, and Yuan 2013), where the different arms are the bid/budget pairs. In the standard multi-armed bandit problem, we are given a set of options called *arms*, at each turn we choose a single arm, and we observe only its reward. Differently, in a combinatorial-bandit problem we are given a set of *superarms*, i.e., an element of the power set of the arms—here corresponding to a combination of bid/budget pairs for each subcampaign—, whose elements satisfy some set of combinatorial constraints—in our case, knapsack-like. At each round, we simultaneously play of all the arms contained in the selected superarm and, subsequently, observe their rewards. We use Gaussian Process (GP) regression models (Rasmussen and Williams 2006) to estimate, for each subcampaign, the expected daily *number of clicks* for each bid/budget pair and the *value per click*. We estimate the value per click of a subcampaign separately from the number of clicks since, while the number of daily clicks and, thus, of the observed samples is usually large allowing one to obtain accurate estimates in short time, the acquisitions are usually much more sporadic and the estimation of the value per click may require a longer time. As a result, at the very beginning of the learning phase our algorithm maximizes the number of clicks, whereas, subsequently, the objective function is gradually tuned by more accurate estimates of the values per click of each subcampaign. This rationale is the same one currently followed by human experts. We design two Bayesian bandit techniques to balance exploration and exploitation in the learning process, that return samples of the stochastic variables estimated by the GPs. Finally, we discretize the bid/budget space, and we formulate the optimization problem as a special case of Multiple-Choice Knapsack problem (Sinha and Zoltners 1979) in which we use the samples returned by the bandit algorithms, and we solve it in polynomial time by dynamic programming in a fashion similar to the approximation scheme for the knapsack problem. The optimization is repeated every day.

We experimentally evaluate, using a realistic simulator based on the *Yahoo!* Webscope $A3$ dataset, the convergence of our algorithm to the optimal (clairvoyant) solution and its regret as the size of the problem varies. Furthermore, we evaluate our algorithm in a real-world campaign with several subcampaigns in Google AdWords for two consecutive months, obtaining the same number of acquisitions obtained by human experts, but halving the cost per acquisition.

Problem Formulation

We are given an advertising campaign $\mathcal{C} = \{C_1, \dots, C_N\}$, with $N \in \mathbb{N}$, where C_j is the j-th subcampaign, a finite time horizon of $T \in \mathbb{N}$ days, and a *spending plan* $\mathcal{B} = \{\overline{y}_1, \dots, \overline{y}_T\}$, where $\overline{y}_t \in \mathbb{R}^+$ is the cumulative budget one is willing to spend at day $t \in \{1, \dots, T\}$.[3,4] While the proposed definition is general w.r.t. the channel we target, in the specific case of search engines (where the subcampaigns are commonly targeted to multiple keywords), we assume that each subcampaign has been set s.t. its keywords behave similarly. As a consequence, a single decision for each group of keywords is required. For day $t \in \{1, \dots, T\}$ and for every subcampaign C_j, the advertiser needs to specify the bid $x_{j,t} \in [\underline{x}_{j,t}, \overline{x}_{j,t}]$, where $\underline{x}_{j,t}$, $\overline{x}_{j,t} \in \mathbb{R}^+$ are the minimum and the maximum bid we can set, respectively, and the budget $y_{j,t} \in [\underline{y}_{j,t}, \overline{y}_{j,t}]$, where $\underline{y}_{j,t}, \overline{y}_{j,t} \in \mathbb{R}^+$ are the minimum and the maximum budget we can set, respectively. The goal is, for every day $t \in \{1, \dots, T\}$, to find the best values of bids and budgets, maximizing the subcampaigns cumulative expected revenue. These values can be found by solving the following optimization problem:

$$\max_{x_{j,t}, y_{j,t}} \sum_{j=1}^{N} v_j \, n_j(x_{j,t}, y_{j,t}) \tag{1a}$$

$$\text{s.t.} \sum_{j=1}^{N} y_{j,t} \leq \overline{y}_t \tag{1b}$$

$$\underline{x}_{j,t} \leq x_{j,t} \leq \overline{x}_{j,t} \qquad \forall j \tag{1c}$$

$$\underline{y}_{j,t} \leq y_{j,t} \leq \overline{y}_{j,t} \qquad \forall j \tag{1d}$$

where $n_j(x_{j,t}, y_{j,t})$ is the expected number of clicks given the bid $x_{j,t}$ and the budget $y_{j,t}$ for subcampaign C_j and v_j is the value per click for the subcampaign C_j. Basically, this is a special case of Multiple-Choice Knapsack problem (Kellerer, Pferschy, and Pisinger 2004) in which the objective function (1a)—corresponding to the value provided by knapsack—is the weighted sum of the expected number of clicks of all the subcampaigns, where the weights are the subcampaigns' value per click. Constraint (1b) is a budget constraint, forcing one not to spend more than the budget limit, while constraints (1c) and (1d) define the ranges of the variables. Similarly to the knapsack problem, here we have items–i.e., the subcampaigns—, each of which is characterized by a value and requires a portion of the budget. The differences are: the occupancy of an item is not a constant, being controllable by the assigned budget; we can decide on a further parameter, the bid, that does not have a corresponding parameter in the knapsack problem; the value of each item is not constant, but it depends on the decisions taken.

Since the function of the number of clicks $n_j(\cdot, \cdot)$ and the parameter specifying the value per click v_j need to be es-

[3] We assume that campaign $\mathcal{C}$ and spending plan $\mathcal{B}$ are given.

[4] For the sake of presentation, from now on we set the day as unitary temporal step of our algorithm. The use of shorter time units can be used to tackle situations in which the user's behaviour is non-stationary over the day. The application of the proposed algorithm to different time units is straightforward.

timated online, not being *a priori* known, the optimization problem can be naturally formulated in a sequential decision learning fashion (Cesa-Bianchi and Lugosi 2006), or, more precisely, as a combinatorial bandit problem (Chen, Wang, and Yuan 2013).[5] Here, we would like to gather as much information as possible about the stochastic functions during the operational life of the system and, at the same time, we do not want to lose too much revenue in doing so (a.k.a. exploration/exploitation dilemma). More precisely, the available options (a.k.a. arms) are all the values of bid $x_{j,t}$ and budget $y_{j,t}$ satisfying the combinatorial constraints of the optimization problem, while $n_j(\cdot,\cdot)$ and v_j are stochastic functions defined on the feasible region of the variables that we need to estimate during the time horizon T. A policy $\mathfrak{U}$ solving such a problem is an algorithm returning, for each day t and subcampaign C_j, a bid/budget pair $(\hat{x}_{j,t}, \hat{y}_{j,t})$. Given a policy $\mathfrak{U}$, we define the *pseudo regret* as:

$$R_T(\mathfrak{U}) := T\,G^* - \mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{N} v_j\; n_j(\hat{x}_{j,t}, \hat{y}_{j,t})\right],$$

where $G^* := \sum_{j=1}^{N} v_j\; n_j(x_j^*, y_j^*)$ is the expected value provided by a clairvoyant algorithm, the set of bid/budget pairs $\{(x_j^*, y_j^*)\}_{j=1}^{N}$ is the optimal clairvoyant solution to the problem in Equations (1a)–(1d), and the expectation $\mathbb{E}[\cdot]$ is taken w.r.t. the stochasticity of the policy $\mathfrak{U}$. Our goal is the design of algorithms minimizing the pseudo regret $R_T(\mathfrak{U})$.

Proposed Method

Initially, we provide an overview of our algorithm named AdComB—Advertising Combinatorial Bandit algorithm—, and, subsequently, we describe in detail the phases composing the algorithm.

The Main Algorithm

Algorithm 1 reports the high-level pseudocode of our method. For the sake of presentation, we distinguish three phases that are repeated each day t (see Fig. 1). For each subcampaign C_j, the parameters to the algorithm are: a finite set X_j of feasible bid values, a finite set Y_j of feasible budget values, a model $\mathcal{M}_j^{(0)}$ capturing a prior knowledge about the function $n_j(\cdot,\cdot)$ and of the parameter v_j, a spending plan $\mathcal{B}$ and a time horizon T.

In the first phase (Lines 4–8), named *Estimation* in Fig. 1, the algorithm learns, from the observations of days $\{1,\ldots,t-1\}$, the model $\mathcal{M}_j$ of the user behavior for each subcampaign C_j. More precisely, the model $\mathcal{M}_j$ provides a probability distribution over the number of clicks $n_j(x,y)$ as the bid x and the budget y vary and over the value per click v_j. The first day the algorithm is executed, no observation is available, and thus the model $\mathcal{M}_j$ is based on the prior $\mathcal{M}_j^{(0)}$ (Line 5). Conversely, the subsequent days, for each subcampaign C_j, the algorithm also gets the observations corresponding to day $t-1$ (Line 7) including: the actual number of clicks $\tilde{n}_{j,t-1}$, the actual total cost of the subcampaign $\tilde{c}_{j,t-1}$, the time when the daily budget $\overline{y}_t$ finished $\tilde{r}_{j,t-1}$, if so, and the actual value per click $\tilde{v}_{j,t-1}$. Subsequently, the model of each subcampaign $\mathcal{M}_j$ is updated using those observations (Line 8).

In the second phase (Lines 9–11), named *Bandit Choice* in Fig. 1, the algorithm chooses the values for the function $n_j(\cdot,\cdot)$ and the parameter v_j using the model $\mathcal{M}_j$ just updated. More precisely, for each subcampaign C_j, the algorithm initially selects the bids $X_{j,t} := X_j \cap [\underline{x}_{j,t}, \overline{x}_{j,t}]$ and budgets $Y_{j,t} := Y_j \cap [\underline{y}_{j,t}, \overline{y}_{j,t}]$ that are feasible according to the given ranges (Lines 9–10). Subsequently, the algorithm chooses, according to the probability distributions of $\mathcal{M}_j$, the samples of the function $n_j(\cdot,\cdot)$ for the feasible values of bid and budget in $X_{j,t}$ and $Y_{j,t}$ and for v_j (Line 11).

In the third phase (Lines 12–13), named *Optimization* in Fig. 1, the algorithm uses the values of $n_j(\cdot,\cdot)$ and of v_j as parameters of the problem in Equations (1a)–(1d) and solves this problem returning the bid/budget pairs to be set for the current day t (Line 13) in each different channel (denoted by *Search*, *Social* and *Contextual* in Fig. 1).[6]

In what follows, we provide a detailed description of the model $\mathcal{M}_j$ and of the subroutines Update$(\cdot)$, Sampling$(\cdot)$,

Algorithm 1 AdComB

1: **Parameters**: sets $\{X_j\}_{j=1}^N$ of bid values, sets $\{Y_j\}_{j=1}^N$ of budget values, prior model $\{\mathcal{M}_j^{(0)}\}_{j=1}^N$, spending plan $\mathcal{B}$, time horizon T
2: **for** $t \in \{1,\ldots,T\}$ **do**
3: **for** $j \in \{1,\ldots N\}$ **do**
4: **if** $t = 1$ **then**
5: $\mathcal{M}_j \leftarrow \mathcal{M}_j^{(0)}$
6: **else**
7: Get $(\tilde{n}_{j,t-1}, \tilde{c}_{j,t-1}, \tilde{r}_{j,t-1}, \tilde{v}_{j,t-1})$
8: $\mathcal{M}_j \leftarrow$ Update $(\mathcal{M}_j, (\hat{x}_{j,t-1}, \hat{y}_{j,t-1}, \tilde{n}_{j,t-1}, \tilde{c}_{j,t-1}, \tilde{r}_{j,t-1}, \tilde{v}_{j,t-1}))$
9: $X_{j,t} \leftarrow X_j \cap [\underline{x}_{j,t}, \overline{x}_{j,t}]$
10: $Y_{j,t} \leftarrow Y_j \cap [\underline{y}_{j,t}, \overline{y}_{j,t}]$
11: $(n_j(\cdot,\cdot), v_j) \leftarrow$ Sampling $(\mathcal{M}_j, X_{j,t}, Y_{j,t})$
12: $\{(\hat{x}_{j,t}, \hat{y}_{j,t})\}_{j\in N} \leftarrow$ Optimize $(\{n_j(\cdot,\cdot), v_j, X_{j,t}, Y_{j,t}\}_{j\in N}, \overline{y}_t)$
13: Set $(\{(\hat{x}_{j,t}, \hat{y}_{j,t})\}_{j\in N})$

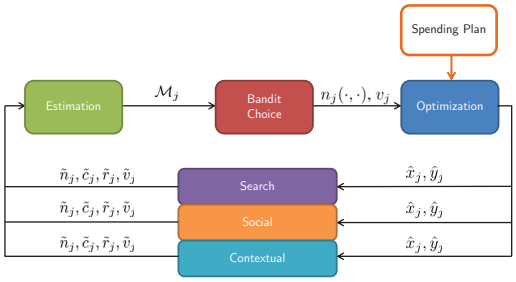

Figure 1: The information flow in the AdComB algorithm along the three phases.

[5] Another approach to solving the problem is to use a multistage method, e.g., backward induction, but it would require a huge computational effort that makes the problem intractable.

[6] Notice that, since the bid and the budget can assume a finite set of values, the problem in Equations (1a)–(1d) can be easily formulated as a Mixed Integer Linear Program (see the Supplemental Material for the mathematical programming formulation).

and Optimize$(\cdot)$ used in Algorithm 1.

Model and Update Subroutine

As mentioned before, the goal of this subroutine is the estimation of the functions $n_j(\cdot,\cdot)$ and the parameter v_j. The crucial issue, in this case, concerns the employment of a practical estimation model, providing a good tradeoff between accuracy and time needed for the learning process. For instance, let us observe that a straightforward approach employing independent estimates of $n_j(\cdot,\cdot)$ for every bid/budget pair (x, y) is not practical since it would require a huge amount of observations, and, thus, too many days to have accurate estimates. Suppose, for instance, to use 10 bid values and 10 budget values, with a total number of 100 bid/budget pairs. Such a discretization would require a period of 100 days only to have a single observation per estimates and years to have accurate estimates, thus making the algorithm useless in practice. Most of the methods for combinatorial bandits available in the state of the art (Chen, Wang, and Yuan 2013; Chen et al. 2016; Gai, Krishnamachari, and Jain 2010; Ontañón 2017) suffers from the same issue, not exploiting any correlation among the random variables corresponding to the arms rewards.

To address this issue, we assume that the function $n_j(\cdot,\cdot)$ presents some regularities and, in particular, that the values of the function at different points in the bid/budget space are correlated. We capture this regularity resorting to GPs (Rasmussen and Williams 2006). These models, developed in the statistical learning field, express the correlation of the nearby points in the input space exploiting a kernel function. Moreover, they provide a probability distribution over the output space—in our case the number of clicks—for each point of the input space—in our case the space of bid and budget—, thus giving information both on the expected values of the quantities to estimate as well as their uncertainty.

In particular, we propose two approaches for $n_j(\cdot,\cdot)$. A straightforward approach, which will be used as a baseline in our experimental activity, employs a single GP defined on a 2-dimension input space (details are provided in the Supplemental Material). Even if this method provides a flexible way of modeling the advertising phenomenon, it requires, due to the curse of dimensionality, a long initial phase before being effective. This issue is addressed by our second approach which exploits an assumption on the structure of the problem. More precisely, the dependency of $n_j(x, y)$ from bid x and budget y is modeled by two 1-dimensional GPs combined in a nonlinear fashion. Formally, we assume:

$$n_j(x,y) := n_j^{\mathsf{sat}}(x)\ \min\left\{1, \frac{y}{c_j^{\mathsf{sat}}(x)}\right\}, \tag{2}$$

where the two GPs employed for each subcampaign C_j are:
– the maximum number of clicks $n_j^{\mathsf{sat}} : \mathcal{X}_j \to \mathbb{R}^+$ that can be obtained with a given bid x without any budget constraint (or equivalently if we let $y \to +\infty$);
– the maximum cost incurred $c_j^{\mathsf{sat}} : \mathcal{X}_j \to \mathbb{R}^+$ with a given bid x without any budget constraint, as above;
where the bid space is defined as $\mathcal{X}_j := \cup_{t=1}^{T}[\underline{x}_{j,t}, \overline{x}_{j,t}]$. The rationale behind this decoupled model is that, given a bid x, the number of clicks increases linearly in the budget y where the coefficient is the average cost per click given x (i.e., $\frac{n_j^{\mathsf{sat}}(x)}{c_j^{\mathsf{sat}}(x)}$), until the maximum number of obtainable clicks is achieved. Notice that the values of $n_j^{\mathsf{sat}}(x)$ and $c_j^{\mathsf{sat}}(x)$ depend on the average position in which the ad is displayed when bid x is used and on the daily number of auctions. The larger x the larger $n_j^{\mathsf{sat}}(x)$ and $c_j^{\mathsf{sat}}(x)$.

Now we focus on the modeling of the maximum number of clicks $n_j^{\mathsf{sat}}(\cdot)$ with a GP regression model. The application of such techniques to estimate the maximum cost $c_j^{\mathsf{sat}}(\cdot)$ is analogous. We model $n_j^{\mathsf{sat}}(\cdot)$ in a subcampaign C_j with a GP over $\mathcal{X}_j$, i.e., we use a collection of random variables s.t. any finite subset has a joint Gaussian distribution. Following the definition provided in (Rasmussen and Williams 2006), a GP is completely specified by its mean $m : \mathcal{X}_j \to \mathbb{R}$ and covariance $k : \mathcal{X}_j \times \mathcal{X}_j \to \mathbb{R}$ functions. Hence, we denote the GP that models the maximum number of clicks in C_j as follows:

$$n_j^{\mathsf{sat}}(x) := \mathcal{GP}\left(m(x), k(x,\cdot)\right), \forall x \in \mathcal{X}_j.$$

If we have *a priori* information about the process, we can use it to design a function $m(x)$ over the input space $\mathcal{X}_j$ which specifies the initial mean value, e.g., if we have information about the maximum number of clicks we might reach θ for any bid, one might consider a linearly increasing function over the bid space as $m(x) = \frac{\theta}{\max_t \overline{x}_{j,t}} x$. If no *a priori* information is available, we set $m(x) = 0, \forall x \in \mathcal{X}_j$.

At the beginning of the optimization procedure ($t = 1$), we have the same predictive distribution at each point of the input space, i.e., $n_j^{\mathsf{sat}}(x) \sim \mathcal{N}(m(x), k(x,x))$, where we denote with $\mathcal{N}(\mu, \sigma^2)$ the Gaussian distribution with mean μ and variance σ^2. For each day t we obtain a value for the maximum number of clicks by relying on:

$$\tilde{n}_{j,t}^{\mathsf{sat}} := d(\tilde{r}_{j,t}, \tilde{n}_{j,t}),$$

where $d(\cdot,\cdot)$ is a function specifying the distribution of the clicks over the day. The function $d(\cdot,\cdot)$ can be estimated from historical data coming from past advertising campaigns of products belonging to the same category (e.g., toys, insurances, beauty products). The vector of the bid set so far $\hat{\boldsymbol{x}}_{j,t-1} := (\hat{x}_{j,1}, \ldots, \hat{x}_{j,t-1})^T$ and the vector of maximum number of clicks $\tilde{\boldsymbol{n}}_{j,t-1}^{\mathsf{sat}} := \left(\tilde{n}_{j,1}^{\mathsf{sat}}, \ldots, \tilde{n}_{j,t-1}^{\mathsf{sat}}\right)^T$ are used by the algorithm to refine the initial prior model $\mathcal{M}_j^{(0)}$. From the definition of GP, its restriction over a finite number of points is a multivariate Gaussian random variable, which can be used, for every $x \in \mathcal{X}_j$, to predict the expected value $\mu_{j,t-1}(x)$ and variance $\sigma_{j,t-1}^2(x)$ of the maximum number of clicks in the following way:

$$\mu_{j,t-1}(x) = m(x) + K(x, \hat{\boldsymbol{x}}_{j,t-1})\Phi^{-1}\left(\tilde{\boldsymbol{n}}_{j,t-1}^{\mathsf{sat}} - (m(\hat{x}_{j,1}), \ldots, m(\hat{x}_{j,t-1}))^T\right),$$

$$\sigma_{j,t-1}^2(x) = k(x,x) - K(x, \hat{\boldsymbol{x}}_{j,t-1})\,\Phi^{-1}\,K(x, \hat{\boldsymbol{x}}_{j,t-1})^T,$$

where we define $\Phi := K(\hat{\boldsymbol{x}}_{j,t-1}, \hat{\boldsymbol{x}}_{j,t-1}) + \sigma_n^2 I$, I is the identity matrix of order $t-1$, and the (i,h)-element of the matrix $K(\boldsymbol{x}, \boldsymbol{x'})$ is the value of the kernel computed over

the i-th element of the generic vector $\boldsymbol{x}$ and the h-th element of the generic vector $\boldsymbol{x'}$. Hence, the maximum number of clicks for the bid x is distributed as the Gaussian $\mathcal{N}(\mu_{j,t-1}(x), \sigma^2_{j,t-1}(x))$.

Regarding the value per click v_j, at day t we estimate it by exploiting the data $\tilde{v}_{j,h}$ recorded during the days $h < t$. Resorting to the Central Limit theorem, we have that the mean value per click v_j is asymptotically Gaussian distributed, thus, at each day t, it is sufficient to estimate its mean $\nu_{j,t}$ and variance $\phi^2_{j,t}$ as follows:

$$\nu_{j,t-1} := \frac{\sum_{h=1}^{t-1} \tilde{v}_{j,h}}{t-1}, \quad \phi^2_{j,t-1} := \frac{\sum_{h=1}^{t-1} \left(\tilde{v}_{j,h} - \nu_{j,t-1}\right)^2}{(t-1)(t-2)}.$$

Overall, the model $\mathcal{M}_j$ corresponding to a subcampaign C_j at a day t consists of the following vectors: the values per click $\tilde{\boldsymbol{v}}_{j,t-1} := (\tilde{v}_{j,1}, \ldots, \tilde{v}_{j,t-1})^T$, the selected bids $\hat{\boldsymbol{x}}_{j,t-1}$, the maximum number of clicks $\boldsymbol{n}^{\mathsf{sat}}_{j,t-1}$, and the maximum costs $\boldsymbol{c}^{\mathsf{sat}}_{j,t-1}$ (defined similarly to $\boldsymbol{n}^{\mathsf{sat}}_{j,t-1}$). Therefore, the Update subroutine of Algorithm 1 includes the incoming data $(\tilde{n}_{j,t-1}, \tilde{c}_{j,t-1}, \tilde{r}_{j,t-1}, \tilde{v}_{j,t-1})$, properly transformed, in the aforementioned vectors.[7]

Sampling Subroutine

The models $\mathcal{M}_j$ we estimate for each subcampaign provide a probability distribution over the function $n_j(\cdot,\cdot)$ and of the values v_j and, therefore, over the possible instances of the optimization problem in Equations (1a)–(1d). The Sampling subroutine generates, from $\mathcal{M}_j$, a single instance of the optimization problem, assigning a value to $n_j(x,y)$ for every $x \in X_j, y \in Y_j$ and a value to v_j. In the present paper, we propose two novel Bayesian approaches, namely AdComB-TS and AdComB-BUCB, taking inspiration from the Thompson Sampling (TS) algorithm (Thompson 1933) and the BayesUCB algorithm (Kaufmann, Cappé, and Garivier 2012), respectively. We resort to the Bayesian approach since, in most of the bandit scenarios, it leads to better performance than that one of their frequentist counterparts, see, e.g., (Chapelle and Li 2011; Granmo 2010; May et al. 2012; Paladino et al. 2017).

The AdComB-TS algorithm generates the values for $n^{\mathsf{sat}}_j(x)$ and $c^{\mathsf{sat}}_j(x)$ by drawing samples from the posterior distributions provided by the GPs. More formally, at a given day t for each bid in $x \in X_{j,t}$, we draw a sample for $n^{\mathsf{sat}}_j(x)$ and a sample for $c^{\mathsf{sat}}_j(x)$ as follows:

$$n^{\mathsf{sat}}_j(x) \sim \mathcal{N}(\mu_{j,t-1}(x), \sigma^2_{j,t-1}(x)),$$
$$c^{\mathsf{sat}}_j(x) \sim \mathcal{N}(\eta_{j,t-1}(x), s^2_{j,t-1}(x)),$$

where $\eta_{j,t-1}(x)$ and $s^2_{j,t-1}(x)$ are the mean and the variance for bid x, respectively, estimated by the GP modeling $c^{\mathsf{sat}}_j(x)$ (we recall that the definitions of $\mu_{j,t-1}(x)$ and $\sigma^2_{j,t-1}(x)$ are provided in the previous section). Similarly, for the value per click we draw a new sample:

$$\hat{v}_j \sim \mathcal{N}(\nu_{j,t-1}, \phi^2_{j,t-1}).$$

Conversely, the AdComB-BUCB algorithm generates samples for $n^{\mathsf{sat}}_j(x)$ and $c^{\mathsf{sat}}_j(x)$ exploiting different time-varying quantiles of the posterior distributions. More precisely, we use a high quantile (of order $1 - \frac{1}{t}$) for $n^{\mathsf{sat}}_j(x)$ and v_j and a low quantile (of order $\frac{1}{t}$) for $c^{\mathsf{sat}}_j(x)$. This assures us to generate optimistic bounds, that are necessary for the convergence of the algorithm to the optimal solution. Let us denote with $q(\mu, \sigma^2, p)$ the quantile of order p of a Gaussian distribution with mean μ and variance σ^2. At day t, for each bid $x \in X_{j,t}$, we generate samples for $n^{\mathsf{sat}}_j(x)$ and $c^{\mathsf{sat}}_j(x)$ as:

$$n^{\mathsf{sat}}_j(x) = q\left(\mu_{j,t-1}(x), \sigma^2_{j,t-1}(x), 1 - \frac{1}{t}\right),$$
$$c^{\mathsf{sat}}_j(x) = q\left(\eta_{j,t-1}(x), s^2_{j,t-1}(x), \frac{1}{t}\right),$$

assuring that $n^{\mathsf{sat}}_j(x)$ is a high-probability upper bound for the maximum number of clicks and $c^{\mathsf{sat}}_j(x)$ is a high-probability lower bound for the maximum costs. Similarly, for the value per click v_j we assign:

$$\hat{v}_j = q\left(\nu_{j,t-1}, \phi^2_{j,t-1}, 1 - \frac{1}{t}\right).$$

Finally, given the values for $n^{\mathsf{sat}}_j(x)$ and $c^{\mathsf{sat}}_j(x)$ generated by one of the two aforementioned methods, we compute $n_j(x,y)$ as prescribed by Equation (2) for each $x \in X_{j,t}$ and for each $y \in Y_{j,t}$.

Optimize Subroutine

Finally, we need to decide a single bid/budget pair to set at day t for each subcampaign C_j. We resort to a modified version of the algorithm in (Kellerer, Pferschy, and Pisinger 2004) used for the solution of the knapsack problem. For the sake of simplicity, let us assume we set an evenly spaced discretization Y of the daily cumulative budget $\overline{y}_t$ and that the feasible values for the budget are a subset of such a discretization, i.e., $Y_{j,t} \subseteq Y,\ \forall j,t$. At first, for each value of budget $y \in Y_{j,t}$ we define $z_j(y) \in X_{j,t}$ as the bid maximizing the number of clicks, formally:

$$z_j(y) := \arg\max_{x \in X_{j,t}} n_j(x,y).$$

The value $z_j(y)$ is easily found by enumeration. Then, for each value of budget $y \in Y$ we define $w_j(y)$ as the value we expect to receive by setting the budget of subcampaign C_j equal to y and the bid equal to $z_j(y)$, formally:

$$w_j(y) := \begin{cases} \hat{v}_j\, n_j(z_j(y), y) & \underline{y}_{j,t} \le y \le \overline{y}_{j,t} \\ 0 & y < \underline{y}_{j,t} \vee y > \overline{y}_{j,t} \end{cases}.$$

This allows one to discard x from the set of the variables of the optimization problem defined in Equations (1a)–(1d), letting variables y the only variables to deal with.

[7] The computation cost of the proposed solution can be dramatically reduced by using an alternative, but much more involved, solution where the inverse of the Gram matrix $K(\hat{\boldsymbol{x}}_{j,t-1}, \hat{\boldsymbol{x}}_{j,t-1})^{-1}$ is stored and updated iteratively at each day; see (Bishop 2006).

Finally, the optimization problem is solved in dynamic programming fashion. We use a matrix $M(j, y)$ with $j \in \{1, \ldots, N\}$ and $y \in Y$. We fill iteratively the matrix as follows. Each row is initialized as $M(j, y) = 0$ for every j and $y \in Y$. For $j = 1$, we set $M(1, y) = w_1(y)$ for every $y \in Y$, corresponding to the best budget assignment for every value of y if the subcampaign C_j were the only subcampaign in the problem. For $j > 1$, we set for every $y \in Y$:

$$M(j, y) = \max_{y' \in Y, y' \leq y} \Big\{ M(j-1, y') + w_j(y - y') \Big\}.$$

That is, the value in each cell $M(j, y)$ is found by scanning all the elements $M(j-1, y')$ for $y' \leq y$, taking the corresponding value, adding the value given by assigning a budget of $y - y'$ to subcampaign C_j and, finally, taking the maximum among all these combinations. At the end of the recursion, the optimal value of the optimization problem can be found in the cell corresponding to $\max_{y \in Y} M(N, y)$. To find the optimal assignment of the budget, it is sufficient to also store the partial assignments of the budget corresponding to the optimal value. The complexity of the aforementioned algorithm is $O(NH^2)$, i.e., it is linear in the number of subcampaigns N and quadratic in the number of different values of the budget $H := |Y|$, where $|\cdot|$ is the cardinality operator. (Let us observe that, although the complexity is polynomial in H, it is pseudopolynomial in $\overline{y}_{j,t}$.) When H is huge, the algorithm could require a long time. In that case, it is sufficient to reduce H by rounding the values of the budget as in the FPTAS of the knapsack problem. This produces a $(1 - \varepsilon)$-approximation of the optimal solution.

Experimental Evaluation

We experimentally evaluate our algorithm both in a synthetic setting, necessary to evaluate the convergence and the regret of our algorithm, and in a real-world setting, necessary to assess its effectiveness when compared with the performance of human experts.

Evaluation in the Synthetic Setting

The synthetic setting we use has been generated according to data of the *Yahoo!* Webscope $A3$ dataset using the simulator developed by Farina and Gatti (2017).

Experiment 1. We consider $N = 4$ subcampaigns, each with a variable number of daily auctions drawn from $\lfloor z \rfloor$ with $z \sim \mathcal{N}(1000, 10)$. Each auction presents 5 slots and 10 advertisers. Each subcampaign is associated with a different truncated Gaussian distribution that is used at every day t to draw the bid of each ad, while, in the same way, the ads' click probability is drawn from a Beta distribution; see (Farina and Gatti 2017) for details. We set a constant cumulative budget per day $\overline{y}_t = 100$ over a time horizon of $T = 100$ days, with limits $\underline{y}_{j,t} = 0$, $\overline{y}_{j,t} = 100$ for every t, j, and we set bid limits to $\underline{x}_{j,t} = 0$ and $\overline{x}_{j,t} = 1$ for every t, j. Furthermore, we use an evenly spaced discretization of $|X_j| = 5$ bids and $|Y_j| = 10$ budgets over the aforementioned intervals. The values per click of each subcampaign are constant—thus letting the clicks to be the only source of randomness—and drawn uniformly from $[0, 1]$. We assume a uniform distribution of the clicks over the day, thus the function $d(\cdot, \cdot)$ has the following expression ($\tilde{r}_{j,h}$ is expressed in hours):

$$d(\tilde{r}_{j,h}, \tilde{n}_{j,h}) := \frac{24}{\tilde{r}_{j,h}} \tilde{n}_{j,h}.$$

We compare the AdComB-TS and AdComB-BUCB algorithms with:[8]

- AdComB-2D-TS, a version of AdComB using a single GP over the two dimensional bid/budget space to estimate the number of clicks (see the Supplemental Material);
- AdComB-Mean, a version of AdComB selecting at each day the average values $\mu_{j,t-1}(x)$ and $\eta_{j,t-1}(x)$ for bid x and $\nu_{j,t-1}$ to be used in the optimization procedure.

For the GPs used in the algorithms, we adopt a squared exponential kernel of the form:

$$k(z, z') := \sigma_f^2 \exp\left\{ -\frac{(z - z')^2}{l} \right\},$$

where $\sigma_f, l \in \mathbb{R}^+$ are kernel parameters, whose values are chosen as prescribed by the GP literature, see (Rasmussen and Williams 2006) for details.

In addition to the pseudo regret $R_t(\mathfrak{U})$, we also evaluate the *instantaneous reward* which is defined as:

$$P_t(\mathfrak{U}) := \sum_{j=1}^{N} v_j n_j(\hat{x}_{j,t}, \hat{y}_{j,t}).$$

We average the results over 100 independent runs.

In Fig. 2a, we report $P_t(\mathfrak{U})$ of the 4 algorithms, while, in Fig. 2b, we report their average $R_t(\mathfrak{U})$. By inspecting the instantaneous reward, we can see that all the algorithms but AdComB-Mean present a slightly varying reward even at the end of the of the time horizon since they incorporate the variance of the GP as information to select the budget over time. This variance is larger at the beginning of the process, thus incentivising exploration, and it fades as the number of observations increases, allowing the algorithm to reach the optimum asymptotically. Moving to Fig. 2b, we observe that the 2 Bayesian algorithms, namely AdComB-TS and AdComB-BUCB, present essentially the same performance, providing the best regret for every $t \leq T$. For $t \leq 20$ the regret of AdComB-Mean is essentially the same one of the 2 Bayesian algorithms. Instead, for larger values of t its relative performance worsens reaching, at $t = 100$, a regret about 35% larger than the one of the AdComB-BUCB algorithm. This is because the exploration performed by AdComB-Mean is not sufficient. As a result, in all the runs, AdComB-Mean does not change the policy for, approximately, $t \geq 40$ and, in some of the 100 independent runs, it gets stuck in a suboptimal solution, as we can observe in Fig. 2a. Conversely, the 2 Bayesian algorithms,

[8]The comparison with the algorithm by Chen, Wang, and Yuan (2013)—in a version accounting for Gaussian distributions—is unfair. Indeed, this algorithm requires 50 days to have a single sample per random variable, and, for all $t \leq T$, it purely explores the space of arms without any form of exploitation.

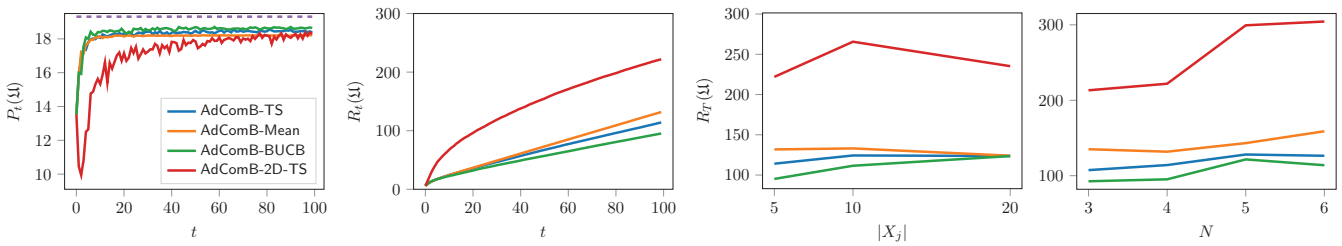

Figure 2: Results for the synthetic setting: instantaneous reward of Experiment 1 (a), pseudo regret over time for Experiment 1 (b), pseudo regret at the end of the time horizon for Experiment 2 (c) and Experiment 3 (d). The optimum value of the instantaneous reward G^* is represented with a dashed line in (a).

thanks to a wider exploration, converge to the optimal solution asymptotically in all the runs (the phenomenon is more evident on a longer time horizon, see the Supplemental Material). Finally, we observe that AdComB-2D-TS suffers from a larger regret than that one of the other 3 algorithms—more than 100% compared with the regret of AdComB-TS and AdComB-BUCB at $t = 100$—and this is mainly accumulated over the first half of the time horizon.

Experiment 2. The experimental setting is the same used above, except that we use $|X_j| \in \{5, 10, 20\}$, s.t. the set of bids we used X_j with $|X_j| = 20$ includes X_j with $|X_j| = 10$ that, in its turn, includes X_j with $|X_j| = 5$. In Fig. 2c we report the average $R_T(\mathfrak{U})$ of the 4 algorithms. Even if increasing the number of bid values may allow one to increase the value of the optimal solution, we observe that the regret slightly increases as $|X_j|$ increases for the AdComB-TS and AdComB-2D-TS algorithms. This is because the algorithms pays a larger exploration cost. Remarkably, the extra cost from $|X_j| = 10$ to $|X_j| = 20$ is small. This result shows that the performance of the algorithms is robust to an increase of the number of possible bids.

Experiment 3. The experimental setting is the same used in the Experiment 1, except that $N \in \{3, 4, 5, 6\}$. In Fig. 2d, we report the average regret $R_T(\mathfrak{U})$ of the 4 algorithms. All the algorithms (except AdComB-2D-TS) do not suffer from a significant increase in the regret as the number of the subcampaigns increases, showing that they scale well as the number of subcampaigns increases.

Evaluation in a Real-world Setting

We advertised a campaign for a loan product of a large Italian finance company with the AdComB-TS algorithm (the names of the product and the company, and other details omitted below are not provided due to reasons of industrial secrecy). The advertising campaign was composed of $N = 13$ subcampaigns. The campaign has been advertised for $T = 120$ days (4 months) in 2017 during which no further advertising campaigns (e.g., video, radio, television) were conducted to avoid mutual effects between the campaigns. The 4 months during which the experiments have been conducted were chosen in such a way, according to past observations, the click behaviour of the users was as homogeneous over time as possible. During the first 2 months, the bid/budget optimization has been performed by human experts, leading to an average of 350 acquisitions per month with an average cost per acquisition of about 83 €. After the first 2 months, the optimization has been performed by the AdComB-TS algorithm in a completely automated fashion. The goal of the company was to reduce the cost per acquisition, given that the cost of 104 € was considered excessively large, keeping the same number of acquisitions per month obtained during the first 2 months. We used a discretization of 5 € for the budget values and 0.10 € for the bid values. The algorithm, implemented in Python 2.7.12 and executed on Ubuntu 16.04.1 LTS with an Intel(R) Xeon(R) CPU E5-2620 v3 2.40GHz, was used at the midnight of each day to decide the bid/budget pairs for the next day. The maximum computation time of the algorithm during the 2 months was less than 1 minute. During the 2 months AdComB-TS was executed, it obtained 353 conversions with an average cost per acquisition of about 56 €. More precisely, in the first month the algorithm was used, the cost per acquisition was about 62 €, while, in the second one, about 50 €. Thus, the average reduction of the cost per acquisition during the first month of execution of the algorithm has been about 25%, while during the second one about 40%.

Conclusions and Future Works

In the current paper, we present AdComB, an algorithm capable of deciding automatically the values of the bid and the budget to set in an advertising campaign to maximize in online fashion the value of the campaign given a spending plan. The algorithm exploits Gaussian Processes to estimate the users' model, combinatorial bandit techniques to address the exploration/exploitation dilemma, and optimization techniques to solve a knapsack-like problem. We propose two flavours of the algorithm, namely AdComB-TS and AdComB-BUCB, differing for the criterion used for the bandit choice. Experiments on both a realistic synthetic setting and real-world setting show that our algorithms tackle the problem properly, outperforming other naive algorithms based on existing solutions and the human expert.

As future work, we plan to study the theoretical properties of the pseudo regret of our algorithm, as well as the study of

techniques to provide a proper setup of the subcampaigns. While in the present work we assume that the environment, including the users and the other advertisers, is stationary over time, we will investigate non-stationary environments, e.g., including in the model the option that there exists periodicity in the user behaviour, as well as some sudden change due the modification of the competitors marketing policy. Moreover, another interesting line of research is to design methods to set up the subcampaigns and possibly modify their targeting over time, basing on their performance.

Acknowledgments. This research has been funded by the Mediamatic company, part of the MMM group. We sincerely thank Roberto Coronel Da Silva, Enrico Dellavalle, and Paola Corbani for their valuable support.

References

Bishop, C. M. 2006. *Pattern recognition and machine learning*. Springer.

Cesa-Bianchi, N., and Lugosi, G. 2006. *Prediction, learning, and games*. Cambridge University Press.

Chapelle, O., and Li, L. 2011. An empirical evaluation of thompson sampling. In *NIPS*, 2249–2257.

Chen, W.; Wang, Y.; Yuan, Y.; and Wang, Q. 2016. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. *J MACH LEARN RES* 17(1):1746–1778.

Chen, W.; Wang, Y.; and Yuan, Y. 2013. Combinatorial multi-armed bandit: General framework and applications. In *ICML*, 151–159.

Ding, W.; Qin, T.; Zhang, X.-D.; and Liu, T. 2013. Multi-armed bandit with budget constraint and variable costs. In *AAAI*, 232–238.

Farina, G., and Gatti, N. 2017. Adopting the cascade model in ad auctions: Efficiency bounds and truthful algorithmic mechanisms. *J ARTIF INTELL RES* 59:265–310.

Gai, Y.; Krishnamachari, B.; and Jain, R. 2010. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In *DySPAN*, 1–9. IEEE.

Gatti, N.; Lazaric, A.; Rocco, M.; and Trovò, F. 2015. Truthful learning mechanisms for multi-slot sponsored search auctions with externalities. *ARTIF INTELL* 227:93–139.

Geyik, S. C.; A-Saxena; and Dasdan, A. 2014. Multi-touch attribution based budget allocation in online advertising. In *ADKDD*, 1–9.

Geyik, S. C.; Faleev, S.; Shen, J.; O'Donnell, S.; and Kolay, S. 2016. Joint optimization of multiple performance metrics in online video advertising. In *SIGKDD*, 471–480.

Granmo, O.-C. 2010. Solving two-armed bernoulli bandit problems using a bayesian learning automaton. *IJICC* 3(2):207–234.

IAB. 2016. Iab internet advertising revenue report 2016, full year results. https://www.iab.com. Online; accessed 21 July 2017.

Italia, E. M.; Nuara, A.; Trovò, F.; Restelli, M.; Gatti, N.; and Dellavalle, E. 2017. Internet advertising for non-stationary environments. In *AMEC*, 1–15.

Kaufmann, E.; Cappé, O.; and Garivier, A. 2012. On bayesian upper confidence bounds for bandit problems. In *AISTATS*, 592–600.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *The Multiple-Choice Knapsack Problem*. Springer. 317–347.

King, M.; Atkins, J.; and Schwarz, M. 2007. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *AM ECON REV* 97(1):242–259.

Kireyev, P.; Pauwels, K.; and Gupta, S. 2016. Do display ads influence search? attribution and dynamics in online advertising. *INT J RES MARK* 33(3):475–490.

Lee, K.-C.; Jalali, A.; and Dasdan, A. 2013. Real time bid optimization with smooth budget delivery in online advertising. In *ADKDD*, 1–9.

Markakis, E., and Telelis, O. 2010. Discrete strategies in keyword auctions and their inefficiency for locally aware bidders. In *WINE*, 523–530.

May, B. C.; Korda, N.; Lee, A.; and Leslie, D. S. 2012. Optimistic bayesian sampling in contextual-bandit problems. *J MACH LEARN RES* 13(Jun):2069–2106.

Ontañón, S. 2017. Combinatorial multi-armed bandits for real-time strategy games. *J ARTIF INTELL RES* 58:665–702.

Paladino, S.; Trovò, F.; Restelli, M.; and Gatti, N. 2017. Unimodal thompson sampling for graph-structured arms. In *AAAI*.

Qin, T.; Chen, W.; and Liu, T.-Y. 2015. Sponsored search auctions: Recent advances and future directions. *ACM T INTEL SYST TEC* 5(4):60:1–60:34.

Rasmussen, C. E., and Williams, C. K. 2006. *Gaussian processes for machine learning*, volume 1. MIT Press.

Sinha, P., and Zoltners, A. A. 1979. The multiple-choice knapsack problem. *OPER RES* 27(3):503–515.

Thomaidou, S.; Liakopoulos, K.; and Vazirgiannis, M. 2014. Toward an integrated framework for automated development and optimization of online advertising campaigns. *INTELL DATA ANAL* 18(6):1199–1227.

Thompson, W. R. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *BIOMETRIKA* 25(3/4):285–294.

Trovò, F.; Paladino, S.; Restelli, M.; and Gatti, N. 2016. Budgeted multi-armed bandit in continuous action space. In *ECAI*, 560–568.

Varian, H. R., and Harris, C. 2014. The VCG auction in theory and practice. *AM ECON REV* 104(5):442–445.

Wang, J.; Zhang, W.; and Yuan, S. 2016. Display advertising with real-time bidding (RTB) and behavioural targeting. *CoRR* abs/1610.03013.

Weinan, W.; Rong, Y.; Wang, J.; Zhu, T.; and Wang, X. 2016. Feedback control of real-time display advertising. In *WSDM*, 407–416.

Xia, Y.; Li, H.; Qin, T.; Yu, N.; and Liu, T.-Y. 2015. Thompson sampling for budgeted multi-armed bandits. In *IJCAI*, 3960–3966.

Xu, J.; Lee, K.-C.; Li, W.; Qi, H.; and Lu, Q. 2015. Smart pacing for effective online ad campaign optimization. In *SIGKDD*, 2217–2226.

Zhang, W.; Zhang, Y.; Gao, B.; Yu, Y.; Yuan, X.; and Liu, T.-Y. 2012. Joint optimization of bid and budget allocation in sponsored search. In *SIGKDD*, 1177–1185.

Zhang, W.; Yuan, S.; and Wang, J. 2014. Optimal real-time bidding for display advertising. In *SIGKDD*, 1077–1086.