Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Cesar Nogueira Coutinho

Online Advertiser-Centric Budget Allocation

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática, do Departamento de Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Marco Serpa Molinaro

Rio de Janeiro
July 2020

Pontifícia Universidade Católica
do Rio de Janeiro

Eduardo Cesar Nogueira Coutinho

Online Advertiser-Centric Budget Allocation

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Marco Serpa Molinaro
Advisor
Departamento de Informática – PUC-Rio

Prof. Eduardo Sany Laber
Departamento de Informática – PUC-Rio

Prof. Marcus Vinícius Soledade Poggi de Aragão
Departamento de Informática – PUC-Rio

Rio de Janeiro, July 9, 2020

Eduardo Cesar Nogueira Coutinho

Bachelor in Computer Engineering (2017) at the Instituto Militar de Engenharia (IME).

Bibliographic data

Coutinho, Eduardo Cesar

Online Advertiser-Centric Budget Allocation / Eduardo Cesar Nogueira Coutinho; advisor: Marco Serpa Molinaro. – 2020.

52 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2020.

Inclui bibliografia

1. Otimização Online. 2. Teoria da Computação. 3. Projeto e Análise de Algoritmos. 4. Anúncios em Display. I. M Molinaro. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Firstly, I would like to thank my advisor, Marco Molinaro, for all the support during my master's and for his exemplary professionalism in conducting the guidance of this work. I would also like to thank the examination committee, Eduardo Laber and Marcus Poggi, for their comments and suggestions.

My appreciation also extends to my family, that always supported me in each stage of my life, especially to my mother Marcia Nogueira, my aunts Arlete and Gloria and my brother Ricardo. My sincere gratitude to my wife Marina for the love and companionship in the last years that always helped me to insist on my goals.

Last but not least, I must thank the friends Yúri Forain, João Gris, Victor Feitosa and Pedro Telles, that inspired me at different times to write this dissertation.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Abstract

Coutinho, Eduardo Cesar; M Molinaro (Advisor). **Online Advertiser-Centric Budget Allocation**. Rio de Janeiro, 2020. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

In this work, we propose the problem AdInvest, which models the decision-making process for allocating investment in digital marketing from the advertiser' perspective. For the proposed problem, we define an algorithm called balGreedy, and we prove its guarantees in deterministic and stochastic instances of the AdInvest. The proven theorems assure to our algorithm worst-case results relatively close to OPT, in several types of instances raised during the work. In particular, we focus on the instances that model the audience saturation effect, which is present in the dynamics of online advertisements. As shown in the computational experiments, the balGreedy algorithm had been consistently efficient compared to the alternative policies adopted, both in the instances generated by simulation and in real instances built from the data of a certain Facebook Ads advertiser.

Keywords

Online Optimization; Theory of Computation; Design and Analysis of Algorithms; Display Advertisemen.

Resumo

Coutinho, Eduardo Cesar; M Molinaro. **Alocação de Recursos Online da Perspectiva de Anunciantes**. Rio de Janeiro, 2020. 52p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Nesse trabalho, propomos o problema AdInvest, que modela o processo decisiório de alocação de investimento em marketing digital do ponto de vista do anunciante. Para o problema proposto, definimos um algoritmo chamado balGreedy, e provamos suas garantias para instâncias determísticas e estocásticas do AdInvest. Os teoremas provados garantem ao nosso algoritmo resultados de pior caso relativamente próximos ao OPT, em diversos tipos de instâncias levantadas ao decorrer do trabalho. Em especial, focamos nas instâncias que modelam o efeito de saturação das audiências, que se faz presente na dinâmica de anúncios online. Como mostrado nos experimentos computacionais, o algoritmo balGreedy se mostrou consistentemente eficiente em comparação com as políticas alternativas adotadas, tanto nas instâncias que foram geradas por simulação, quanto em instâncias reais obtidas a partir de dados de um anunciante do Facebook Ads.

Palavras-chave

Otimização Online; Teoria da Computação; Projeto e Análise de Algoritmos; Anúncios em Display.

Table of contents

List of figures

List of tables

1
Introduction

There are many marketing channels where the advertisers can invest money to display online banners, sponsored links, and other types of digital advertisement. In this type of market, one of the investors' intents is to maximize a user action, for example clicks in their ad banners.

Marketing channels often offer some features to allow advertisers to focus their efforts on targeted audiences. For instance, a wine company can choose to show their ads only to middle-aged people, and a female cosmetics company is able to invest only in female audiences. However, the advertisers' task of choosing which audiences they should invest in may not be simple because of the following practical issues: 1) The advertisers do not know in advance which are the more profitable audiences; 2) The profitability of an audience tends to decrease as it receives impressions of an ad (*saturation effect*).

This motivates the following online *advertiser-centric* budget allocation problem that captures how the advertisers should decide in which audiences and channels to invest (see Section 1.1 for a more formal definition):

- There is a fixed number K of options (audiences in the different marketing channels) to be invested.
- Each option gives the agent a unit reward (a banner click, for example) after he has invested a certain amount of money in it. The advertiser gets the feedback when such reward happens.
- The amount of money needed for obtaining each reward is unknown, and it depends on the option as well as on how many rewards the option has already given thus far (allowing to model the audience saturation)
- The goal is to minimize the total money spent on obtaining a set number of rewards.

See Figure 1.1 for an illustration. We note that while we focus on the budget allocation decisions, there are several other important components that appear in practice. For example, in many channels, the display of an ad is subject to a *real-time bidding* process, in which the advertiser needs to decide its bids on the different audiences (for example, by bidding on keywords in a search engine) and compete

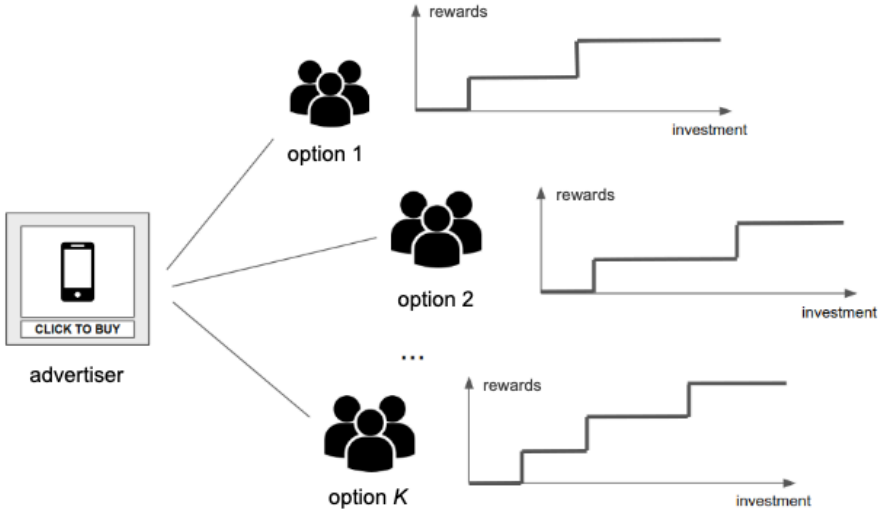

Figure 1.1: Online advertiser-centric budget allocation problem. Each option has an unknown reward function that describes how many (cumulatively) rewards are obtained depending on the agent's investment in that option.

with other bidders for the display. In addition, often the advertiser has flexibility in setting some definition of an audience (for example, deciding in which keywords to bid in an audience). Here we are abstracting out options and the different rules of interaction with them via their reward function, thus focusing on making higher-level budget allocation decisions. Besides, although the time variable may be relevant in certain types of marketing channels, we are not directly modeling it in this problem.

1.1 Problem definition

We now formally define the problem AdInvest. For an integer t, we use the notation $[t] := \{1, \ldots, t\}$. There are k fixed options that can be invested in by the agent. For each option $i \in [k]$, there is an **unknown** sequence of non-negative *marginal conversion costs* $(x_i^j)_{j \in \mathbb{N}}$, where x_i^j is the additional investment necessary for obtaining its jth conversion after its $(j-1)$th conversion has been obtained (see Figure 1.2). That is, after investing x_i^1 in option i the player gets a conversion; after investing an additional x_i^2 in this option (for a total of $x_i^1 + x_i^2$) the player gets the second conversion from this option, etc. The player continuously decides how much to invest in each option. After each investment, the player learns which options yielded new conversions; this is the only feedback the player receives about the marginal conversion costs. Given a target value S, the player's goal is to obtain S conversions while minimizing its total investment over all the options.

As mentioned before, an important class of instances are those that have a

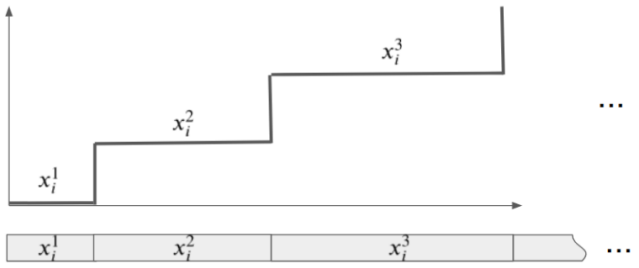

Figure 1.2: The marginal conversion costs for option i. Each x_i^j represents the additional investment necessary for obtaining the jth conversion after the $(j-1)$th conversion has been obtained.

saturation effect, that is when the marginal conversion costs are non-decreasing within each option, namely $x_i^1 \leq x_i^2 \leq \ldots$ for each i. We call these instances *monotone*.

Given an instance I and a target S, we use $\text{OPT}_S(I)$ to denote the offline optimal investment cost for obtaining S conversions, namely the optimal cost of a policy that knows all marginal conversion costs x_i^j's upfront. Our goal is to design online algorithms whose costs are competitive against this offline OPT, namely that for some values α, β satisfy

$$\textsc{Alg}_S(I) \leq \alpha \cdot \text{OPT}_S(I) + \beta, \qquad \text{for every instance } I.$$

Example 1. *Consider the instance I where for a fixed value L and multiplier $\alpha \geq 1$, we have the last option with all marginal values equal to L, and the marginal values of the other options are equal to αL:*

$$x_i^j = \begin{cases} \alpha L & \text{if } i \in [k-1] \\ L & \text{if } i = k \end{cases}.$$

Let the target value S be a multiple of k. See Figure 1.3 for a depiction of this instance.

One of the simplest investment policy is roundRobin*: the algorithm starts in option 1, and invest on it continuously until a reward is obtained. At this point it gets the feedback that a reward was obtained, and moves to investing in the next option. The process is repeated until S rewards are obtained (after obtaining the reward from option k, go back to option 1). On the instance above,* $\textsc{roundRobin}_S$ *obtains precisely $\frac{S}{k}$ conversions in each option, at a cost*

$$\textsc{roundRobin}_S(I) \;=\; (k-1)\alpha L\,\frac{S}{k} + L\,\frac{S}{k} \;=\; \left[(1-\tfrac{1}{k})\alpha + \tfrac{1}{k}\right] \cdot LS.$$

On the other hand, the optimal policy is to invest only in the last option until it yields S rewards. Thus, we have $\text{OPT}_S(I) = LS$. *As expected,* $\textsc{roundRobin}_S$ *is a bad strategy: taking α arbitrarily large, this simple strategy can be arbitrarily bad compared to* OPT_S.

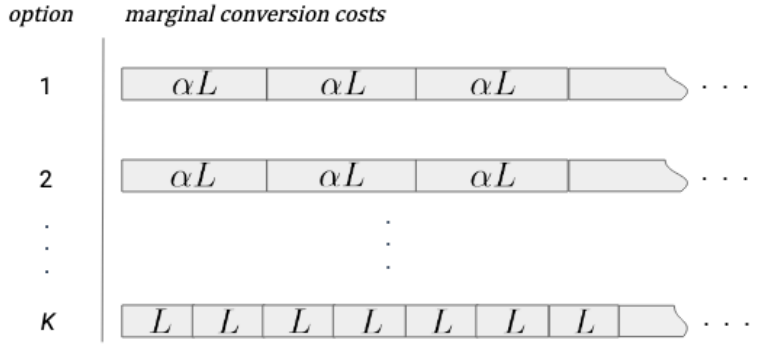

Figure 1.3: Instance of the example, where the last option has the smallest marginal costs.

1.2 Our results

Our main result is to show that a single algorithm balGreedy has a provably good performance for several natural classes of instances. In particular, this illustrates that this is a *robust* algorithm, which is a good candidate for performing well in practice. We also provide (almost matching) lower bounds, as well as computational results to validate the practical performance of the algorithm.

Monotone instances. We first show that algorithm balGreedy gives a good approximation for monotone instances.

Theorem 1. *Consider the problem AdInvest. The algorithm* balGreedy *has the following guarantee: for every monotone instance I and integer S, the cost of the algorithm for obtaining S conversions is bounded as*

$$\textsc{balGreedy}_S(I) \leq \text{OPT}_S(I) + k \cdot C^*,$$

where C^* is the highest marginal cost paid by an optimal solution $\text{OPT}_S(I)$.

An important observation is that this guarantee does not depend on the target number S of conversions, which is desirable since the value S is typically large in many situations. In addition, in principle C^* can be as large as $\text{OPT}_S(I)$, in which case the guarantee above only gives a multiplicative $(k+1)$-approximation. However,

typically $\text{OPT}_S(I)$ grows as a function of the target S; since the S is usually much larger than the number of options k, in many situations the above guarantee actually gives a $(1+\varepsilon)$-approximation for a small constant ε.

Moreover, we show that no algorithm can have a much better performance guarantee.

Theorem 2. *With the notation from Theorem 1, for every deterministic algorithm* ALG *there is a monotone instance I and target S such that*

$$\textit{ALG}_S(I) \geq \text{OPT}_S(I) + (k-2)\cdot C^*,$$

and for every randomized algorithm there is I and S such that

$$\textit{ALG}_S(I) \geq \text{OPT}_S(I) + \frac{k-3}{2}\cdot C^*.$$

Approximately monotone instances. Next, we consider instances where the saturation/monotonicity property is only satisfied approximately. Inspired by *robust optimization* [4], it is useful to think that there is a "nominal" instance $\bar{I}$ that is monotone, but the player actually faces a perturbed version I of this instance that may not be monotone. We show that the same algorithm balGreedy still works well for these instances, with the guarantee depending on how close the nominal and actual instances are.

Theorem 3. *Consider a monotone nominal instance $\bar{I} = (\bar{x}_i^j)_{i,j}$ and a (not necessarily monotone) actual instance $I = (x_i^j)_{i,j}$. Let α be such that $x_i^j \leq \alpha \bar{x}_i^j$ for all i, j. Then for all targets S, the algorithm* balGreedy *satisfies*

$$\textsc{balGreedy}_S(I) \leq \alpha(\text{OPT}_S(\bar{I}) + k\cdot C^*),$$

where C^* is the highest nominal marginal cost paid by $\text{OPT}_S(\bar{I})$.

Notice that when the actual instance I is monotone Theorem 3 recovers Theorem 1, since we can take $\bar{I} = I$. We remark that such online algorithms whose guarantees degrade nicely when the assumptions of the model are only partially satisfied is a broad area of active interest [19, 14, 5, 28, 23, 35, 17].

Stochastic instances. Another important situation is when the instance is a noisy version of a monotone instance, that is, the marginal conversion costs are random variables whose expected values are non-decreasing in each option.

Definição 1.1 (Random instances)**.** *A* random instance that is monotone in expectation $I = (X_i^j)_{i,j}$ *is one where the marginal conversion costs X_i^j are non-negative*

independent random variables, and the expected instance $\bar{I} = (\mathbb{E}X_i^j)_{i,j}$ *is monotone.*

We first consider the case where the random costs X_i^j's have exponential tails. The following is a direct consequence of Theorem 3.

Theorem 4. *Consider a random instance* $I = (X_i^j)_{i,j}$ *that is monotone in expectation. Also, assume that the costs have sub-exponential tails, namely there is* β *such that*

$$\Pr(X_i^j \geq \lambda \mathbb{E}X_i^j) \leq e^{-\lambda\beta} \quad \forall \lambda \geq 0,\ \forall i, j.$$

Then balGreedy *has expected cost upper bounded as*

$$\mathbb{E}\,\textsc{balGreedy}_S(I) \leq \frac{1}{\beta}(1 + \log kS)(\text{OPT}_S(\bar{I}) + k \cdot C^*),$$

where $\bar{C}^*$ *is the highest (mean) marginal cost paid by* $\text{OPT}_S(\bar{I})$.

This result covers, in particular, the case when the marginal costs are exponentially distributed, which could model real instances, as described in more detail in the section on our computational experiments (Section 6). Notice that while this guarantee now depends on the target number of conversions S, this dependence is only logarithmic, and hence still within a meaningful range in many situations.

Finally, we consider another class of stochastic instances with bounded deviation from the mean.

Theorem 5. *Consider a random instance* $I = (X_i^j)_{i,j}$ *that is monotone in expectation, and define* $\mu_i^j = \mathbb{E}X_i^j$. *Let*

$$\Delta := \min_{i,i',j,j' : \mu_i^j \neq \mu_{i'}^{j'}} |\mu_i^j - \mu_{i'}^{j'}|$$

be the smallest gap between the means. Let L *be an upper bound such that* $|X_i^j - \mu_i^j| \leq L$ *for all* i, j, *and define* $top := \min_{i,j} \Pr(X_i^j \geq \mu_i^j + L - \Delta)$. *Then we have*

$$\mathbb{E}\,\textsc{balGreedy}_S(I) \leq \text{OPT}_S(\bar{I}) + kC^* + kL\left(1 + \frac{1}{top}\right),$$

where $\bar{C}^*$ *is the highest (mean) marginal cost paid by* $\text{OPT}_S(\bar{I})$.

The main advantage of this guarantee compared to Theorem 4 is that it is independent of the target number of conversions S. Instead, this guarantee has a new term $Lk(1 + \frac{1}{top})$ that depends on the range $\pm L$ of the deviation of the marginal costs, and also on the probability top that the deviation is close to its maximum

value L. Notice that when all "noises" $X_i^j - \mu_i^j$ are the same bounded symmetric distribution (e.g., truncated Normal) then $\top$ is a non-zero quantity that only depends on this distribution (in particular being independent of k and S). Finally, we remark that the term Δ involved in this bound is a generalization of the standard "winner gap" present in the standard bounds for *multi-armed bandit* problems [6].

Computational experiments. We computed experiments for both simulated and real instances. For the simulated data, we generated synthetic data for marginal conversion costs from some stochastic processes to model different types of instance contemplated by the above theorems above. In addition, we built 3 real instances from extracting campaign data from a *Facebook Ads* advertiser. The algorithm balGreedy was the best online policy in almost all instances. In some instances, in which the options were very similar, other algorithms had similar, or even better, results than balGreedy. In the simulated instances, the balGreedy performed better than all the online alternative algorithms and achieved good results even compared to offline alternatives. Besides, the balGreedy obtained the best result in all except one real instance.

1.3 Related work

Despite a great wealth of works in the broad area of online advertisement, we are not aware of any model that tackles the advertiser's investment problem as proposed here.

Arguably most of the work in the literature focuses on the optimization problem from the perspective of the marketing channel. For example, there are several results on choosing which ads are displayed to a given user (e.g., upon a search on a search engine) aiming at maximizing the channel's revenue. A very important example is the *AdWords* problem introduced by [25]: users come one-by-one in an online fashion, and when a new user comes the marketing channel decides on witch ad to show him/her. Each advertiser reports a bid for this user, which is the money the channel earns if it shows an ad from the advertiser. However, the advertisers also set (at the beginning of the time horizon) the total budget that they are willing to spend on the channel over all time steps. The channel's goal is to allocate ads to users in an online fashion so as to maximize its revenue while respecting the advertisers' budgets. In [25], Mehta et al. give an online algorithm that is within a multiplicative $(1 - 1/e)$-factor of the optimal offline solution (under mild conditions). For other works in the area see [1, 22, 15, 30, 13, 11, 29, 21, 10]

There are several other papers that consider optimization problems from the perspective of the advertisers. Many of them focus on optimizing an advertiser's bids

on channels that display ads based on real-time bidding. For example, one relevant task is predicting the "value" (e.g., probability of a click) that an advertiser gets from displaying an ad to a given user, see for example [8, 3, 2, 24, 7], an important information when choosing which bid to give. In other areas, [16, 12, 31, 34] aim at learning in an online fashion the bidding strategy of the other advertisers competing for ad displays. Yet another set of papers focus on optimizing the actual bidding strategy of an advertiser in a given marketing channel [27, 33, 32, 18] Compared to these works, the model we introduce here is not restricted to real-time bidding-based channels and is less fine-grained, abstracting several components of the problem into the reward function of each option.

Our model is also somewhat related to the *multi-armed bandit problem* [6]. In one classic version of this problem, there are k options/arms that can be "pulled" in each time step, and upon a pull, it gives a random reward sampled from an unknown arm-dependent (but usually time-invariant) distribution. Given a limited number of pulls, the goal is to chose which arms to pull in each time step in order to maximize the expected reward. Here, the performance of the algorithm is typically compared to the expected reward of the best arm, i.e., OPT must play the same arm (the best one) over all the time steps. In contrast, our model considers time-varying reward (intervals) and the algorithms are compared against a fully dynamic OPT.

2
Algorithm balGreedy and OPT

In this section, we describe our algorithm, then give a more combinatorial view of the process, and finally show that when the instance is monotone the optimal offline solution is given by the greedy algorithm. This structure of OPT will be crucial for the development of guarantees on balGreedy in the remaining sections.

In this section we consider deterministic instances $I = (x_i^j)_{i,j}$ that may not be monotone; we will explicitly state when and where monotonicity is used.

2.1
Algorithm balGreedy

We will present an online algorithm balGreedy for this problem and the respective guarantees. We consider that the process occurs in continuous time, with the algorithm investing with infinitesimal increments.

At each point in time, the *stash* of an option i denotes the total amount invested by balGreedy in option i since the last conversion of this option, or equivalently, the amount invested that has not yet yielded a conversion (see Figure 1.1).

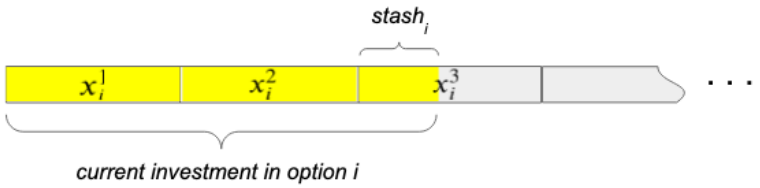

Figure 2.1: Example of stash for option i. The colored part represents the current total investment in that option.

The algorithm balGreedy$_S$ is then the following investment policy:

Algorithm 1: balGreedy$_S$

At every point in time, invest in the option with the smallest stash; if there is more than one such option, invest equally in all options with minimum stash.

Stop after S conversions are obtained

Notice that the stash of an option becomes 0 when a new conversion is obtained from this option (all stashes start at 0).

Example 2. *consider again the instance defined in Example 1. Executing the algorithm* balGreedy *on this instance, we see that this policy starts investing equally in all options until the first conversion occurs (Figure 2.2). From this point on, the algorithm will only invest in the last option, until it gets S conversions in total (Figure 2.3). Thus,* balGreedy *spends a total of $SL + (k-1)L$ in this instance (recall $\text{OPT}_S(I) = SL$).*

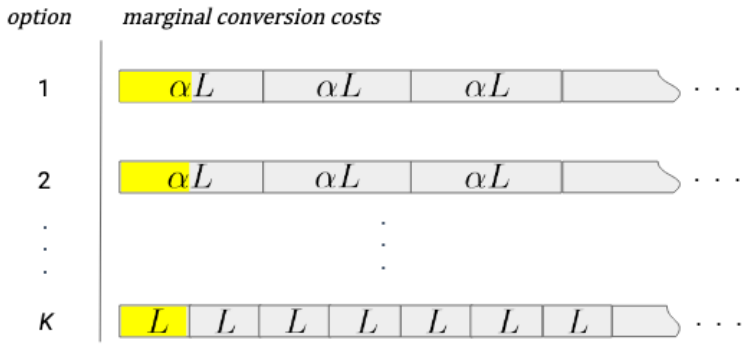

Figure 2.2: balGreedy running on the instance from Example 1. Shown here is the moment right before the first conversion.

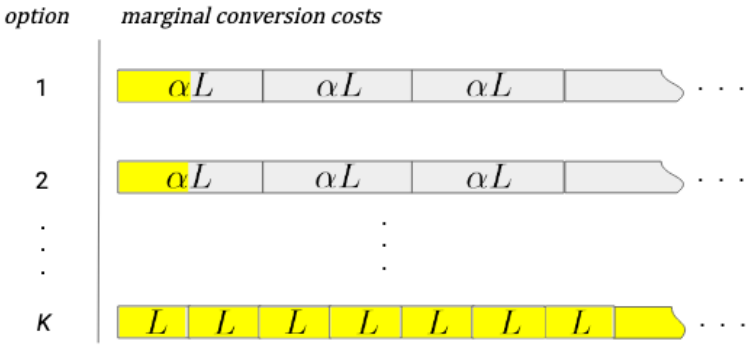

Figure 2.3: balGreedy running on the instance from Example 1. Shown here is the "endgame" when the algorithm obtains a total of S conversions.

Water-filling perspective. balGreedy can also be described more visually as a *water-filling* algorithm. Figure 2.4 depicts the execution of balGreedy at three points in time, with the leftmost image happening right after a conversion (at option 3). The vertical bars represent the marginal costs $x_i^{j_i+1}$ for the next conversion in each of the options i (so there have been j_i conversions in option i). The yellow bars represent the stash in each option. Then balGreedy can be thought of as continuously raising the "water level" λ, investing in the options to ensure that all

stash levels are at least λ. Notice that the next conversion happens when the level λ reaches the smallest next-conversion cost $x_i^{j_i+1}$.

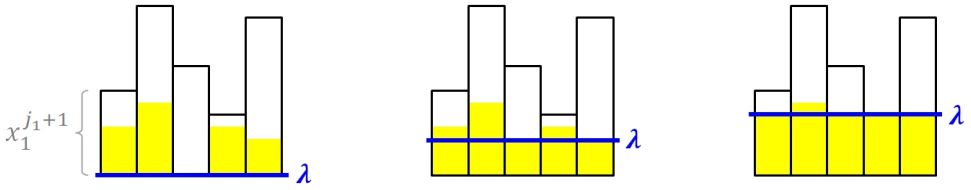

Figure 2.4: Image depicting the water-filling perspective of algorithm balGreedy. The leftmost image depicts the moment right after a conversion (at option 3), and the rightmost image depicts the moment right before the next conversion (at option 4).

2.2 A more combinatorial view of the problem

To compare the execution of the different algorithms, it will be useful to think of the problem in a more combinatorial way. We think of each option i as composed of *slabs* $(i,1),(i,2),\ldots$ that can be collected. Each slab corresponds to a possible conversion, with the cost of the slab (i,j) being the marginal cost x_i^j of the corresponding conversion. To collect a slab, the algorithm needs to pay its cost. In addition, in any feasible solution, the jth slab of option i can only be collected after all the previous slabs of that option have been collected. So a *feasible solution* selects a prefix $\{(i,1),\ldots,(i,j_i)\}$ of the slabs of each option i, see Figure 2.5. The goal of the algorithm is to collect a total S slabs using the different options. Finally, in our online problem, the cost of a slab is only known after the algorithm fully pays for it.

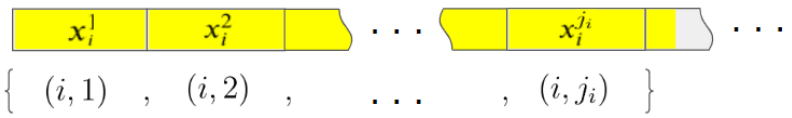

Figure 2.5: Image describing the combinatorial view for the picked slabs for an option.

One remark in terms of the total cost of the algorithms: while the total cost of OPT_S is the just sum of the costs of the slabs that it picks, this is **not true** for balGreedy_S, since it may have made investments that were not enough to generate a conversion. But notice that these "leftover" investments are exactly what compose the stashes of the options. Thus we have the following.

Observation 1 (Decomposition into slab and stash costs)**.** *The total cost of* $\textsc{balGreedy}_S(I)$ *is the cost of the slabs that it picks (*slab cost*) plus the sum of the stashes of the options at the end of its execution (*stash cost*). We use* $\mathit{stash}(\textsc{balGreedy}_S(I))$ *and* $\mathit{slab}(\textsc{balGreedy}_S(I))$ *to denote these costs respectively.*

This combinatorial view allows us to abstract the two main properties of feasible solutions for our problem:

Lemma 1 (Feasible sets of slabs)**.** *We have the following properties:*

- *(Augmentation) If* $\mathcal{S}$ *and* $\mathcal{S}'$ *are two feasible sets of slabs and* $\mathcal{S}'$ *has more slabs than* $\mathcal{S}$*, then there is a slab* $s \in \mathcal{S}' \setminus \mathcal{S}$ *such that* $\mathcal{S} \cup \{s\}$ *is feasible.*
- *(Compatibility) Consider a monotone instance* $I = (x_i^j)_{i,j}$ *and a feasible set of slabs* $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ *ordered by costs, namely* $x(s_j) \leq x(s_{j+1})$ *for all* j*. Then for every* t*, the set of the* t *cheapest slabs* $\{s_1, s_2, \ldots, s_t\}$ *of the solution is also feasible.*

The first property holds because for any such solutions $\mathcal{S}, \mathcal{S}'$ there is an option i where the solution $\mathcal{S}'$ picks more slabs than $\mathcal{S}$; then the next slab $s = (i, j_i + 1)$ in this option that $\mathcal{S}$ did not pick belongs to $\mathcal{S}'$ and can be added to $\mathcal{S}$ while maintaining feasibility.

For the second property, by feasibility, the solution $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ picks out a prefix of slabs $(i, 1), \ldots, (i, j_i)$ for each option i. Then the instance is monotone, the cost of the slab increases as we traverse such prefix, and hence truncating the solution $\mathcal{S}$ to its t cheapest slabs simply selects a sub-prefix $(i, 1), \ldots, (i, \bar{j}_i)$ for each option, thus giving a feasible solution.

(We remark that these properties show that the feasible sets form a *greedoid*, which is a generalization of matroids, and that monotone instances are *R-compatible*, which is an important property for the optimality of greedy algorithms, see [20].)

2.3 Greedy and the offline optimum OPT

Consider the (offline) greedy algorithm for selecting slabs that always picks the "next slab" in one of the options that has smallest cost.

Algorithm 2: $\textsc{Greedy}_S$

Input: Costs $I = (x_i^j)_{i,j}$, target number of slabs S

While has not picked S slabs

Let j_i be the last slab picked in option i

Among the next slabs $\{(i, j_i + 1)\}_i$, pick the one with smallest cost $x((i, j_i + 1))$

Notice that if there are multiple slabs with the same cost then there are multiple executions of $\textsc{Greedy}_S$ depending on tie breaking, but it will not be important to differentiate these executions.

We show next that for monotone instances, this greedy procedure is optimal. (As indicated above, this follows from the general theory of greedoids, but we provide a short self-contained proof.)

Lemma 2. *For monotone instances I, any execution of $\textsc{Greedy}_S$ returns an optimal solution, namely $\textsc{Greedy}_S(I) = \text{OPT}_S(I)$.*

Proof. Let $\mathcal{S}^* = \{s_1^*, \ldots, s_S^*\}$ be the slabs of an optimal solution and let $\mathcal{S} = \{s_1, \ldots, s_S\}$ be the slabs picked by $\textsc{Greedy}_S(I)$, both with items sorted in non-decreasing cost (so $x(s_i^*) \leq x(s_{i+1}^*)$ for all i and similarly for the s_i's). Let $\mathcal{S}_t^* = \{s_1^*, \ldots, s_t^*\}$ be the t cheapest slabs of the optimal solution, and define $\mathcal{S}_t$ analogously (which are feasible due to Lemma 1).

If for all t we have $x(s_t) \leq x(s_t^*)$ then clearly $\textsc{Greedy}_S(I) = \text{OPT}_S(I)$. Then assume by contradiction that there is t such that $x(s_t) > x(s_t^*)$.

We claim that $x(s_t) \leq x(s_t^*)$ for all t, which then shows $\textsc{Greedy}_S(I) = \text{OPT}_S(I)$. To see this, consider the solution $\mathcal{S}_{t-1}$ obtained by $\textsc{Greedy}_S(I)$ right before selecting s_t. By the augmentation property of Lemma 1 there is $\bar{s} \in \mathcal{S}_t^* \setminus \mathcal{S}_{t-1}$ such that $\mathcal{S}_{t-1} \cup \{\bar{s}\}$ is feasible, so $\bar{s}$ was a possible candidate when $\textsc{Greedy}_S(I)$ selected s_t. Then by the greedy criterion and the fact s_t^* is the most expensive slab in $\mathcal{S}_t^* \ni \bar{s}$, we have $x(s_t) \leq x(\bar{s}) \leq x(s_t^*)$ as claimed. This concludes the proof. □

3
Monotone instances

In this section, we consider monotone instances and prove the guarantee for balGreedy over (Theorem 1) as well as an almost matching lower bound for every online algorithm (Theorem 2).

3.1
Guarantee for balGreedy

We state again the guarantee that we will prove for balGreedy.

Theorem 1. *Consider the problem AdInvest. The algorithm* balGreedy *has the following guarantee: for every monotone instance I and integer S, the cost of the algorithm for obtaining S conversions is bounded as*

$$\textsc{balGreedy}_S(I) \leq \text{OPT}_S(I) + k \cdot C^*,$$

where C^* is the highest marginal cost paid by an optimal solution $\text{OPT}_S(I)$.

To prove this theorem, first recall from Observation 1 that the cost $\textsc{balGreedy}_S(I)$ incurred by the algorithm can is the sum of its stash $\text{stash}(\textsc{balGreedy}_S(I))$ and slab costs $\text{slab}(\textsc{balGreedy}_S(I))$. To upper bound the slab cost, notice that the water-filling perspective of balGreedy (Figure 2.4) shows that the next slab it picks is always that of minimum cost. That is:

Lemma 3. *For every instance (even non-monotone ones) I, the slabs picked by $\textsc{balGreedy}_S(I)$ are exactly the same as those picked by some execution of the greedy procedure $\textsc{Greedy}_S(I)$. In particular,*

$$\textit{slab}(\textsc{balGreedy}_S(I)) = \textsc{Greedy}_S(I).$$

Since for monotone instances Lemma 2 guarantees that the greedy procedure selects an optimal solution, we directly have the following.

Corollary 1 (Slab cost)**.** *If I is a monotone instance, then $\textsc{balGreedy}_S(I)$ picks the same slabs as some optimal solution $\mathcal{S}^*$. In particular we have*

$$\textit{slab}(\textsc{balGreedy}_S(I)) = \textsc{Greedy}_S(I) = \text{OPT}_S(I).$$

Next, we upper bound the stash cost of the algorithm.

Lemma 4. *Consider a (possibly non-monotone) instance $I = (x_i^j)_{i,j}$, and let $\mathcal{S}$ be the set of slabs picked by* balGreedy$_S(I)$. *Then the stash cost of the algorithm can be upper bounded using the maximum slab cost:*

$$\textit{stash}(\textsc{balGreedy}_S(I)) \leq k \cdot \max_{s \in \mathcal{S}} x(s).$$

Proof. Let stash_i denote the stash at option i at the end of the execution of balGreedy$_S(I)$. We claim that for all i,

$$\text{stash}_i \leq \max_{s \in \mathcal{S}} x(s). \tag{3-1}$$

For that, fix an option i. Using the water-filling perspective of balGreedy$_S$ (Figure 2.4), we see that during the execution the stash cost of option i only increases if the "water level" λ reaches its current stash cost. This implies that at all points in time, the stash cost of option i is at most the highest water level seen thus far. Moreover, is $\mathcal{S} = \{s_1, \ldots, s_S\}$ are the slabs picked by balGreedy$_S(I)$ in the order in which they are picked, we see that the water level raises from 0 to $x(s_1)$ (causing the algorithm to pick this slab), then drops to 0 and raises to $x(s_2)$, etc. Thus, the highest water level reached during the execution is precisely the max slab cost $\max_{s \in \mathcal{S}} x(s)$. This proves inequality (3-1), and concludes the proof of the lemma. □

Since Corollary 1 guarantees that for monotone instances there is an optimal solution with the same slabs as balGreedy, its max slab cost is the same as that of this optimal solution, giving the following.

Corollary 2 (Stash cost)**.** *If $I = (x_i^j)_{i,j}$ is a monotone instance, then there is*[1] *an optimal solution $\mathcal{S}^*$ such that*

$$\textit{stash}(\textsc{balGreedy}_S(I)) \leq k \cdot \max_{s^* \in \mathcal{S}^*} x(s^*).$$

Adding the bounds from Corollaries 1 and 2 concludes the proof of Theorem 1.

3.2 Lower bound for monotone instances

Now we prove a lower bound for online algorithms over monotone instances (again restated for convenience).

[1] Actually using the same exchange argument as in the proof of Lemma 2 it is easy to see that the max slab cost is the same for all optimal solution.

Theorem 2. *With the notation from Theorem 1, for every deterministic algorithm* ALG *there is a monotone instance I and target S such that*

$$\textsc{Alg}_S(I) \geq \text{OPT}_S(I) + (k-2) \cdot C^*,$$

and for every randomized algorithm there is I and S such that

$$\textsc{Alg}_S(I) \geq \text{OPT}_S(I) + \frac{k-3}{2} \cdot C^*.$$

Proof. This is a reduction from the following promise version of the UNORDERED-SEARCH problem: The input is an array of integers A of size k, and an integer x that appears in one of the entries of A. The goal is to locate the position of x in the array while minimizing the number of lookups to the entries of the array. It is a folklore result that to solve this problem every deterministic algorithm requires in the worst-case at least $k-1$ probes, and for every randomized algorithm there is an instance that makes the expected number of probes to be at least $\frac{k-1}{2}$.[2]

Given an instance (A, x) of this problem, we can encode it into an instance $I = (\mu_i^j)_{i,j}$ of our problem as follows: The first slab on all options cost 1, namely $x_i^1 = 1$ for all $i \in [k]$. For the option i^* such that $A[i^*] = x$, all other slabs cost 0; for all other options, all other slabs cost ∞. The target number of slabs S is set to be $k+1$.

The offline optimal solution of this instance has cost

$$\text{OPT}_{k+1}(I) = 1,$$

just picks $k+1$ slabs in the option i^*. But notice that every online algorithm that solves this instance with finite cost needs to locate the position i^* whose **second** slab has cost 0. But to start seeing the second slab of any option it needs to "probe" it, namely to buy its first slab at cost 1. Then from the above lower bound, any deterministic algorithm needs to pay at least

$$\textsc{Alg}_{k+1}(I) \geq k-1 = \text{OPT}_{k+1}(I) + (k-2) \cdot C^*.$$

The lower bound for randomized algorithms follows by just using the randomized lower bound for UNORDEREDSEARCH instead. □

[2]The lower bound for deterministic algorithms is immediate, and we outline the proof for randomized algorithms: By Yao's Minimax Theorem [26], it suffices to consider how deterministic algorithms do in expectation over the instance that plants x in a uniformly random position of A. It is not hard to see that, without loss of generality, we can assume the algorithm to be non-adaptive and that it probes the entries $A[1], A[2], \ldots, A[k-1]$ in order (it does not need to probe $A[k]$ because of the promise that x belongs to A). For all $i = 1, \ldots, k-1$, when x appears in position i the algorithm pays i probes, and this happens with probability $\frac{1}{k}$, giving $\frac{k(k-1)}{2k} = \frac{k-1}{2}$ probes in expectation.

4
Approximately monotone instances

In this section, we prove that balGreedy still works well when the instance is approximately monotone.

Theorem 3. *Consider a monotone nominal instance $\bar{I} = (\bar{x}_i^j)_{i,j}$ and a (not necessarily monotone) actual instance $I = (x_i^j)_{i,j}$. Let α be such that $x_i^j \leq \alpha \bar{x}_i^j$ for all i, j. Then for all targets S, the algorithm* balGreedy *satisfies*

$$\textsc{balGreedy}_S(I) \leq \alpha(\text{OPT}_S(\bar{I}) + k \cdot C^*),$$

where C^* is the highest nominal marginal cost paid by $\text{OPT}_S(\bar{I})$.

Going back to the guarantees for monotone instances given in Section 3, we see that the monotonicity assumption was mostly used to guarantee that greedy procedure $\textsc{Greedy}_S(I)$ (which we know picks the same slabs as $\textsc{balGreedy}_S(I)$) gives an optimal solution $\text{OPT}_S(I)$. Since now our actual instance I is not monotone, it is easy to see that greedy may not give an optimal solution. However, because of the bounds connecting the actual and nominal instances $I, \bar{I}$, in some sense running greedy over the actual instance should give an approximately optimal solution to the nominal instance. The next lemma makes this precise.

Lemma 5. *With the notation from Theorem 3, for every execution of the procedure $\textsc{Greedy}_S(I)$ we have*

$$\textit{Greedy}_S(I) \leq \alpha\, \text{OPT}_S(\bar{I})$$

and also

Proof. Since the nominal instance $\bar{I}$ is monotone, Lemma 2 shows that running $\textsc{Greedy}_S$ over this instance produces an optimal nominal solution $\text{OPT}_S(\bar{I})$. So consider such greedy/optimal solution $\mathcal{S}^* = \{s_1^*, \ldots, s_S^*\}$, sorted in non-decreasing order of nominal cost, so $\bar{x}(s_t^*) \leq \bar{x}(s_{t+1}^*)$ for all t. Also, define the set of its t cheapest slabs $\mathcal{S}_t^* = \{s_1^*, \ldots, s_t^*\}$.

Also let $\mathcal{S} = \{s_1, \ldots, s_S\}$ be the slabs picked by running $\textsc{Greedy}_S$ over the actual instance, sorted in non-decreasing order of actual cost, so $X(s_t) \leq X(s_{t+1})$ for all t. Define $\mathcal{S}_t$ similarly.

We claim that for all t we have $X(s_t) \leq \alpha \bar{x}(s_t^*)$, which then shows that $\textsc{Greedy}_S(I) \leq \alpha\, \text{OPT}_S(\bar{I})$. For that, fix t. By the augmentation property of Lemma

1, there is a slab $s^* \in \mathcal{S}_t^* \setminus \mathcal{S}_{t-1}$ such that $\mathcal{S}_{t-1} \cup \{s^*\}$ is feasible, so s^* was a possible option for $\textsc{Greedy}_S(I)$ on the tth round. But its greedy choice minimizes actual cost, so $X(s_t) \leq X(s^*)$. But using the assumption on I and $\bar{I}$, this is at most α times the nominal cost $\bar{x}(s^*)$, which in turn is at most $\bar{x}^*(s_t^*)$ (since s_t^* is the most expensive slab in the set $\mathcal{S}_t^* \ni s^*$). Chaining these inequalities gives $X(s_t) \leq \alpha \bar{x}(s_t^*)$ as claimed. This concludes the proof. □

This proof also proves the following control over the max slab cost of $\textsc{Greedy}_S(I)$.

Lemma 6. *Let $\mathcal{S}$ be the slabs picked by an execution of $\textsc{Greedy}_S(I)$. There is*[1] *an optimal solution $\mathcal{S}^*$ for the nominal instance $\bar{I}$ (i.e. achieving $\text{OPT}_S(\bar{I})$) such that*

$$\max_{s \in \mathcal{S}} X(s) \leq \alpha \cdot \max_{s^* \in \mathcal{S}^*} \bar{x}(s^*).$$

Proof of Theorem 3. Let $\mathcal{S}$ be the slabs picked by $\textsc{balGreedy}_S(I)$, which by Lemma 3 are the same picked by an execution of $\textsc{Greedy}_S(I)$. Applying the decomposition of the cost of the algorithm into slab plus stash costs of Observation 1, then the control of slab and stash costs for non-monotone instances from Lemmas 3 and 4, and then Lemmas 5 and 6 above, we have

$$\begin{aligned}
\textsc{balGreedy}_S(I) &= \text{slab}(\textsc{balGreedy}_S(I)) + \text{stash}(\textsc{balGreedy}_S(I)) \\
&\leq \textsc{Greedy}_S(I) + k \cdot \max_{s \in \mathcal{S}} X(s) \\
&\leq \alpha \left(\text{OPT}_S(\bar{I}) + k \cdot \max_{s^* \in \mathcal{S}^*} \bar{x}(s^*) \right),
\end{aligned}$$

where $\mathcal{S}^*$ is an optimal solution for the nominal instance $\bar{I}$. This concludes the proof. □

Observation 2. *Notice that in the statement of Theorem 3 it suffices to require the bound $x_i^j \leq \alpha \bar{x}_i^j$ to hold for the first S slabs of each option, namely for $j \leq S$, since these are the only slabs that can appear in the proofs of Lemmas 5 and 6.*

[1] As before, one can show that all optimal solutions have the same max slab cost.

5
Stochastic instances

In this section, we consider two types of stochastic instances, i.e. the marginal conversion costs are now independent random variables, whose expected values form a monotone instance. The first class of instances is when the random costs have sub-exponential tails, for which our guarantee will depend on a parameter β that is related to the sub-exponential decay. Next, we consider instances where the marginal costs have bounded variation, for which we prove guarantees that are again independent of S, but depend on some parameters of the cost distributions.

5.1
Sub-exponential instances

We start by proving Theorem 4, which will be a direct consequence of the guarantee for balGreedy$_S$ for approximately monotone instances.

Theorem 4. *Consider a random instance $I = (X_i^j)_{i,j}$ that is monotone in expectation. Also, assume that the costs have sub-exponential tails, namely there is β such that*

$$\Pr(X_i^j \geq \lambda \mathbb{E} X_i^j) \leq e^{-\lambda\beta} \quad \forall \lambda \geq 0,\ \forall i, j.$$

Then balGreedy *has expected cost upper bounded as*

$$\mathbb{E}\,\text{balGreedy}_S(I) \leq \frac{1}{\beta}(1 + \log kS)(\text{OPT}_S(\bar{I}) + k \cdot C^*),$$

where $\bar{C}^*$ is the highest (mean) marginal cost paid by $\text{OPT}_S(\bar{I})$.

Proof. To simplify the notation let $\mu_i^j := \mathbb{E} X_i^j$. Applying Theorem 3 to each scenario of the random instance I we get that

$$\text{balGreedy}_S(I) \leq R \cdot (\text{OPT}_S(\bar{I}) + k \cdot C^*), \tag{5-1}$$

where R is now the **random** variable that gives the multiplicative gap between the actual instance I and the expected instance in each scenario, namely

$$R := \max\left\{ \max_{i \in [k],\, j \in [S]} \frac{X_i^j}{\mu_i^j},\ 1 \right\}.$$

Here we have used Observation 2, so the ratio R is only defined based on the S first slabs of each option.

To bound the expected value of R we use the concentration properties of the random costs X_i^j via standard arguments. Applying a union bound and then the subexponentiality assumption on the X_i^j's we have

$$\Pr\left(\max_{i\in[k],\, j\in[S]} \frac{X_i^j}{\mu_i^j} \;\geq\; \lambda\right) \leq \sum_{i\in[k], j\in[S]} \Pr\left(\frac{X_i^j}{\mu_i^j} \geq \lambda\right) \;\leq\; kSe^{-\lambda\beta}.$$

Since R is non-negative, we can obtain its expected value by integrating its tails:

$$\begin{aligned}\mathbb{E}R = \int_0^\infty \Pr(R \geq \lambda)\,\mathrm{d}\lambda \;&\leq\; \int_0^{\frac{\log kS}{\beta}} 1\,\mathrm{d}\lambda \;+\; \int_{\frac{\log kS}{\beta}}^\infty kSe^{-\lambda\beta}\,\mathrm{d}\lambda \\ &=\; \frac{\log kS}{\beta} + \frac{1}{\beta}.\end{aligned}$$

Then taking expectation over (5-1) and noticing that on the right-hand side only R is random, we have

$$\mathbb{E}\,\textsc{balGreedy}_S(I) \leq \frac{1}{\beta}(1+\log kS)(\mathrm{OPT}_S(\bar{I}) + k\cdot C^*). \tag{5-2}$$

This concludes the proof of the theorem. □

5.2 Instances with bounded variation

Here we consider stochastic instances with bounded variation, proving Theorem 5 stated in the introduction; we restate here for convenience.

Theorem 5. *Consider a random instance $I = (X_i^j)_{i,j}$ that is monotone in expectation, and define $\mu_i^j = \mathbb{E}X_i^j$. Let*

$$\Delta := \min_{i,i',j,j' : \mu_i^j \neq \mu_{i'}^{j'}} |\mu_i^j - \mu_{i'}^{j'}|$$

be the smallest gap between the means. Let L be an upper bound such that $|X_i^j - \mu_i^j| \leq L$ for all i, j, and define top $:= \min_{i,j} \Pr(X_i^j \geq \mu_i^j + L - \Delta)$. Then we have

$$\mathbb{E}\,\textsc{balGreedy}_S(I) \leq \mathrm{OPT}_S(\bar{I}) + kC^* + kL\left(1 + \frac{1}{top}\right),$$

where $\bar{C}^*$ is the highest (mean) marginal cost paid by $\mathrm{OPT}_S(\bar{I})$.

Notice that while $\textsc{balGreedy}_S(I)$ pays the cost of the actual (random) instance I, we compare here against the optimal value of the *expected* instance, namely $\text{OPT}_S(\bar{I})$.

For the rest of this section, we prove this theorem. Since the expected instance $\bar{I}$ is monotone, using Lemma 2 we consider throughout an optimal solution $\text{OPT}_S(\bar{I})$ that is given by an execution of the greedy procedure.

As in Figure 2.3, we can depict $\text{OPT}_S(\bar{I})$ as the yellow region of the slabs it picks. The main step in analyzing the cost of balGreedy is understanding how much it "overshoots" this region, namely how many additional slabs it picks in each option compared to $\text{OPT}_S(\bar{I})$. Notice that while the cost/size of the slabs is different in the actual instance I (which is what the algorithm pays) and in the expected instance $\bar{I}$ (which is what $\text{OPT}_S(\bar{I})$ pays), it may be useful to try to imagine that these instances have the same slabs, albeit with different costs.

To make this idea precise, let N_i be the number of slabs $\textsc{balGreedy}_S(I)$ picks on option i, and let n_i^* be the number of slabs that $\text{OPT}_S(\bar{I})$ picks on option i. We use $(x)_+ := \max\{x, 0\}$ to denote the positive part of a number.

Lemma 7.

$$\mathbb{E}\, \textit{slab}(\textsc{balGreedy}_S(I)) \leq \text{OPT}_S(\bar{I}) + L \cdot \sum_i \mathbb{E}(N_i - n_i^*)_+.$$

Proof. Let $\mathcal{S} = \{s_1, \ldots, s_S\}$ be the slabs obtained by $\textsc{balGreedy}_S(I)$, and let $\mathcal{S}^* = \{s_1^*, \ldots, s_S^*\}$ be those obtained by $\text{OPT}_S(\bar{I})$. Since both differences $\mathcal{S} \setminus \mathcal{S}^*$ and $\mathcal{S}^* \setminus \mathcal{S}$ have the same number of elements, the previous lemma we have

$$\text{slab}(\textsc{balGreedy}_S(I)) - \text{OPT}_S(\bar{I}) = \sum_{s \in \mathcal{S} \setminus \mathcal{S}^*} X(s) - \sum_{s^* \in \mathcal{S} \setminus \mathcal{S}^*} \mu(s^*) \leq L\, |\mathcal{S} \setminus \mathcal{S}^*|$$

Further notice that the size of the set $\mathcal{S} \setminus \mathcal{S}^*$ can be counted by looking, for each option, at how many slabs the algorithm picked after OPT's slabs, namely $|\mathcal{S} \setminus \mathcal{S}^*| = \sum_i (N_i - n_i^*)_+$. Taking expectations we get

$$\mathbb{E}\, \text{slab}(\textsc{balGreedy}_S(I)) - \text{OPT}_S(\bar{I}) \;\leq\; L \cdot \sum_i \mathbb{E}(N_i - n_i^*)_+,$$

concluding the proof of the lemma. □

The heart of the proof is controlling the expected amount of "overshooting" $\mathbb{E}(N_i - n_i^*)_+$ in each option, which is done in the next lemma.

Lemma 8. *For every option i we have $\mathbb{E}(N_i - n_i^*)_+ \leq \frac{1}{top}$.*

Proof. Fix i. Notice that $N_i \geq n_i^* + b$ if and only if the slab $X_i^{n_i^*+a}$ is picked. The main idea of the proof is that for this to happen, *all* slabs $X_i^{n_i^*}, X_i^{n_i^*+1}, \ldots, X_i^{n_i^*+a}$

must have "small" cost; as a increases, the likelihood of all these events happening should be small. To make this precise, define the $big_{i,b}$ as the event that the cost $X_i^{n_i^*+b}$ is quite bigger than the last expected cost $\mu_{n_\ell^*}^{\ell}$ of all other options $\ell \neq i$:

$$(big_{i,b}) \qquad X_i^{n_i^*+b} > \max_{\ell}\{\mu_\ell^{n_\ell^*} : \ell \neq i\} + L.$$

Claim 1. *Consider $a \geq 1$. If any of the events $\{big_{i,b}\}_{1\leq b\leq a}$ happen, then* balGreedy$_S(I)$ *does not pick the slab $X_i^{n_i^*+a}$.*

Proof. First notice that the event $big_{i,b}$ implies that

$$X_i^{n_i^*+b} > \mu_\ell^{n_\ell^*} + L \geq \mu_\ell^j + L \geq X_\ell^j, \quad \text{for all } j \leq n_\ell^* \text{ and } \ell \neq i, \tag{5-3}$$

where the second inequality follows from the monotonicity of the means $(\mu_\ell^j)_j$, and the last inequality follows from the definition of L.

So by contradiction, assume that one of these events, say $big_{i,b}$ (for $1 \leq b \leq a$), happens and balGreedy$_S(I)$ picks the slab $X_i^{n_i^*+a}$. In particular, it picks the earlier slab $X_i^{n_i^*+b}$. But from Lemma 3, we know that balGreedy$_S(I)$ picks slabs according to an execution of the greedy procedure. But then (5-3) gives that balGreedy$_S(I)$ must have picked all the cheaper slabs X_ℓ^j for all $j \leq n_\ell^*$ and $\ell \neq i$ as well (i.e., $X_i^{n_i^*+b}$ can only be picked after all these slabs). But this means that balGreedy$_S(I)$ has picked a total of at least

$$(n_i^* + b) + \sum_{\ell\neq i} n_\ell^* = (n_1^* + \ldots + n_k^*) + b = S + b \geq S + 1$$

slabs, which is a contradiction. This concludes the proof of the claim. ■

Therefore, since the events $big_{i,b}$ are independent we can upper bound the probability that $N_i \geq n_i^* + a$ as

$$\Pr(N_i \geq n_i^* + a) \leq \Pr\left(\text{none of the events } \{big_{i,b}\}_{b\leq a} \text{ happen}\right) = \prod_{b\leq a}(1 - \Pr(big_{i,b})). \tag{5-4}$$

We now lower bound $\Pr(big_{i,b})$ for $b \geq 1$. Again, since the optimal solution $\text{OPT}_S(\bar{I})$ is given by the greedy procedure (using monotonicity of the expected instance $\bar{I} = (\mu_i^j)_{i,j}$ and Lemma 2), the last slab $\mu_{n_\ell^*}^{\ell}$ it picks at any option ℓ costs at most the slab $\mu_i^{n_i^*+1}$ right after the last one picked at option i (otherwise it would have picked $\mu_i^{n_i^*+1}$ instead of $\mu_\ell^{n_\ell^*}$). Since Δ is the smallest gap between means, we

actually have that $\mu_i^{n_i^*+1} \geq \mu_\ell^{n_\ell^*} + \Delta$ for all $\ell \neq i$, and thus

$$\begin{aligned}
\Pr(big_{i,b}) &= \Pr(X_i^{n_i^*+b} \geq \max\{\mu_{n_\ell^*}^{\ell} : \ell \neq i\} + L) \\
&= \Pr(X_i^{n_i^*+b} - \mu_i^{n_i^*+b} \geq \max\{\mu_\ell^{n_\ell^*} : \ell \neq i\} - \mu_i^{n_i^*+b} + L) \\
&\geq \Pr(X_i^{n_i^*+b} - \mu_i^{n_i^*+b} \geq \max\{\mu_\ell^{n_\ell^*} : \ell \neq i\} - \mu_i^{n_i^*+1} + L) \quad \text{(by monotonicity)} \\
&\geq \Pr(X_i^{n_i^*+b} - \mu_i^{n_i^*+b} \geq -\Delta + L) \\
&\geq top \qquad \text{(by def. of } top\text{)}.
\end{aligned}$$

Plugging this onto the inequality (5-4) gives the exponential bound $\Pr(N_i \geq n_i^* + a) \leq (1 - top)^a$. Using non-negativity and integrality, "integrating" these tails gives the expected value of $(N_i - n_i^*)_+$ and so we get

$$\mathbb{E}(N_i - n_i^*)_+ = \sum_{a \in \mathbb{N}} \Pr(N_i \geq n_i^* + a) \leq \sum_{a \in \mathbb{N}} (1 - top)^a \leq \frac{1}{top}.$$

This concludes the proof of the lemma. □

Putting Lemmas 7 and 8 together gives a clean upper bound on the slab cost of the algorithm.

Corollary 3.

$$\mathbb{E}\, slab(\textsc{balGreedy}_S(I)) \leq \text{OPT}_S(\bar{I}) + \frac{kL}{min}.$$

Finally, we just need to control the max actual slab cost paid by $\textsc{balGreedy}_S(I)$. The same proof of Lemma 6 also proves this result.

Lemma 9. *Let $\mathcal{S}$ be the slabs picked by an execution of $\textsc{Greedy}_S(I)$. There is an optimal solution $\mathcal{S}^*$ for the expected instance $\bar{I}$ (i.e. achieving $\text{OPT}_S(\bar{I})$) such that in every scenario*

$$\max_{s \in \mathcal{S}} X(s) \leq \max_{s^* \in \mathcal{S}^*} \mu(s^*) + L.$$

Proof of Theorem 5. Let $\mathcal{S}$ be the slabs picked by $\textsc{balGreedy}_S(I)$, which by Lemma 3 are the same picked by an execution of $\textsc{Greedy}_S(I)$. Applying the decomposition of the cost of the algorithm into slab plus stash costs of Observation 1, then the control of the (expected) slab and stash costs from Corollary 3 and Lemma 4, and then using Lemma 9, we have

$$\begin{aligned}
\mathbb{E}\, \textsc{balGreedy}_S(I) &= \mathbb{E}\, \text{slab}(\textsc{balGreedy}_S(I)) + \mathbb{E}\, \text{stash}(\textsc{balGreedy}_S(I)) \\
&\leq \text{OPT}_S(\bar{I}) + \frac{Lk}{top} + k \cdot \mathbb{E} \max_{s \in \mathcal{S}} X(s) \\
&\leq \text{OPT}_S(\bar{I}) + \frac{Lk}{top} + k \cdot \max_{s^* \in \mathcal{S}^*} \mu(s^*) + kL,
\end{aligned}$$

where $\mathcal{S}^*$ is an optimal solution for the expected instance $\bar{I}$. This concludes the proof of the theorem. □

6
Computational experiments

We have conducted preliminary experiments to validate the performance of our algorithm balGreedy. Both simulated and real instances of the problem are considered, which are described in detail below. We have compared balGreedy against OPT, roundRobin (defined in Example 1) as well as with the following other baseline algorithms:

randomArm: Randomly choose an option, invest in it until obtaining S conversions.

uniformInvest: Invest equally in all options until obtaining S conversions.

offBestArm: This algorithm chooses the (offline) best option, and invest in it until obtaining S conversions. Notice that this is not an online strategy since it depends on the prior knowledge of which option is the best one.

6.1
Exponential marginal costs in real instances

Consider the following simple modeling for the process of displaying an advertisement: A company wants to receive web traffic to its website. Then, this company decides to invest money in an online marketing channel in banner advertisements to obtain clicks of the users. So the banner is displayed to many different users. When any of them receives the advertisement impression, he has a constant p probability of converting (click in the banner in this case) after the advertisement is displayed to him. If each impression in this channel costs an amount of money c, we have that the investment X necessary for a conversion to occur is defined by a geometric distribution

$$X \sim c \cdot geometric(p).$$

We define the conversion rate as

$$\lambda = \left(\frac{p}{c}\right).$$

In the limit case, when p tends to zero, the random variable X tends to an exponential distribution

$$X \sim exp(\lambda).$$

This limit case fits in many online marketing channels because usually each unique ad impression has a small probability to result in a conversion. This type of modeling was already used before (for example, in [9]) to describe user behavior when interacting with advertisements.

6.2 Simulated data

We generated synthetic instances with 5 options and 50 marginal costs in each option. In order to generate these sequences, we started from probability distributions with a linearly increasing mean in the order of marginal conversion in each option, as shown in Figure 6.1. Since the expected values of marginal conversion cost are modeled from linear functions, we used the *slope* parameter (saturation rate of the options) and *intercept* (initial marginal conversion cost values) to model the expected values of marginal conversion in the options. Then, each expected marginal conversion cost could be model as

$$\mathbb{E}X_i^j = A_i \cdot j + B_i,$$

where A and B are the slope and intercept vectors, respectively.

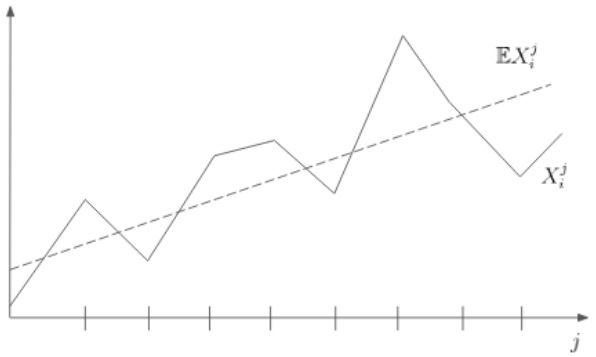

Figure 6.1: Depiction of how the marginal costs are generated in the synthetic instances.

Besides the expected values, we need to define the distributions to generate the marginal costs values. The chosen probability distributions were the *uniform* (with interval size equals to 60), the *exponential* and the *constant*, that is a degeneration that models deterministic instances. The exponential distribution was chosen because it should better model the users' behavior when interacting with the advertisement, as we said before.

In order to simplify the comparison among the results for each instance category, we considered only 2 sets of possibilities for each of the parameters intercept

Instance Category	Distribution	Intercept	Slope
1	CONSTANT	SIMILAR	SIMILAR
2	CONSTANT	SIMILAR	DIFFERENT
3	CONSTANT	DIFFERENT	SIMILAR
4	CONSTANT	DIFFERENT	DIFFERENT
5	UNIFORM	SIMILAR	SIMILAR
6	UNIFORM	SIMILAR	DIFFERENT
7	UNIFORM	DIFFERENT	SIMILAR
8	UNIFORM	DIFFERENT	DIFFERENT
9	EXPONENTIAL	SIMILAR	SIMILAR
10	EXPONENTIAL	DIFFERENT	SIMILAR
11	EXPONENTIAL	SIMILAR	DIFFERENT
12	EXPONENTIAL	DIFFERENT	DIFFERENT

Table 6.1: Simulated instance categories

Parameter	Label	Values
Intercept	SIMILAR	{150, 175, 200, 225, 250}
Intercept	DIFFERENT	{50, 200, 350, 500, 650}
Slope	SIMILAR	{2, 2, 2, 2, 2}
Slope	DIFFERENT	{10, 20, 30, 175, 200}

Table 6.2: Parameters for each instance category

and slope. We labeled the possibilities as SIMILAR, in which all the options have similar values, and DIFFERENT, where the variability among the options parameters is higher. Table 6.2 defines the 12 instance categories that were used to generate the synthetic data. In addition, Table 6.2 shows the exact values used for the labels SIMILAR and DIFFERENT in intercept and slope parameters.

The result for each algorithm and instance category was obtained from computing the average of the result got by the algorithm in 20 simulated instances generated from the instance category. For each generated instance, we attributed the slope and intercept values randomly among the options, without hard defining the values for each option.

Table 6.2 shows the average results obtained by each of the policies in the simulated instances. Among the online policies, BALGREEDY was the best in all the instance categories. In instances 1, 5 and 9, the algorithms UNIFORMINVEST and ROUNDROBIN obtained good results, which should be expected as all the options have similar expected marginal costs functions in these instances, what favors the policies that distribute regularly the conversions between the options, avoiding saturated conversions. On the other hand, the algorithm RANDOMARM had the worst performance in all of the instance categories.

Instance Category	OPT	BALGREEDY	OFFBESTARM	ROUNDROBIN	UNIFORMINVEST	RANDOMARM
1	9221	10049	**9950**	10512	10485	13262
2	15769	**17542**	22500	31209	20688	119100
3	4950	5542	**4950**	18002	10650	18075
4	18919	**21203**	26162	36783	25667	106281
5	9173	10108	**9949**	10431	10477	10502
6	15618	**17381**	22229	29495	20534	104590
7	4915	5579	**4915**	17755	10566	17656
8	19871	**22229**	29572	39796	25900	104147
9	6922	9851	**9392**	10588	10180	12893
10	13358	**19623**	23112	28866	20566	136170
11	4603	6624	**4953**	18217	10141	24048
12	15206	**21844**	25729	44025	25317	136285

Table 6.3: Algorithms comparison in simulated instances

In instances 1, 3, 5, 7, 9 and 11, the offBestArm algorithm obtained a better result than balGreedy, obtaining an average result lower than balGreedy in 1.0%, 10.7%, 1.6%, 11.9%, 4.7%, 25.2% for each of these instance categories. The difference between them was higher in instances with slope similar and intercept different. These instances favor the offBestArm policy because one of the options, precisely the one with intercept component equals to 50, will be very profitable. Then, the strategy that always invests in it will obtain a good result (for example, in instance categories 3 and 7, the offBestArm performed equal to OPT).

6.3 Real data

We obtained data from Facebook Ads for a certain Brazilian advertiser company over a certain period to build the real instances. This advertiser was investing in marketing campaigns to obtain traffic to its website. Thus, the company has invested in some Facebook audiences to impact the social network users with images to possibly get clicks on its ads.

The campaign performance data is available to the advertiser, but there are some important observations to be done about how this data is available on Facebook Ads and how we have processed it to build the instances:

- We considered the ad impression number as the investment costs to build the instances, instead of the spent cost in Brazilian currency, because there are some problems when we use it to define the marginal conversion costs, for example: auction quality metrics[1], exchange fluctuation[2] and auction inflation[3].
- The data provided by Facebook Ads to the advertiser are obtained in hourly blocks of grouped metrics. Then, for each hour we have access to the respective total click value and ad impression for the audiences in which the advertiser was investing. When the ad clicks are greater than 1 in an hourly grouped data block, we can not know what was the exact marginal conversion cost for each of these clicks. In this case, we are considering all the clicks having the same marginal cost, dividing the accumulated cost among the number of clicks. See Figure 6.2.

We built 3 different instances from the private Facebook Ads data provided by the advertiser. Each instance is composed of the marginal costs (measured in

[1]The Facebook Ads continuously defines quality metrics to the campaigns, that directly impact the cost per ad impression in the auctions (If an advertiser has worse metrics that the competitors, he will pay more per ad impression). One of these metrics depends on the user experience after clicking on the ads, which is related to website health.

[2]The Brazilian currency is not so stable, then there is a constant change in intrinsic currency value.

[3]The advertisement is exposed in Facebook Ads in blind auctions, that are very dynamic. Thus, the cost per ad impression changes very drastically when new competitors enter or leave the auction.

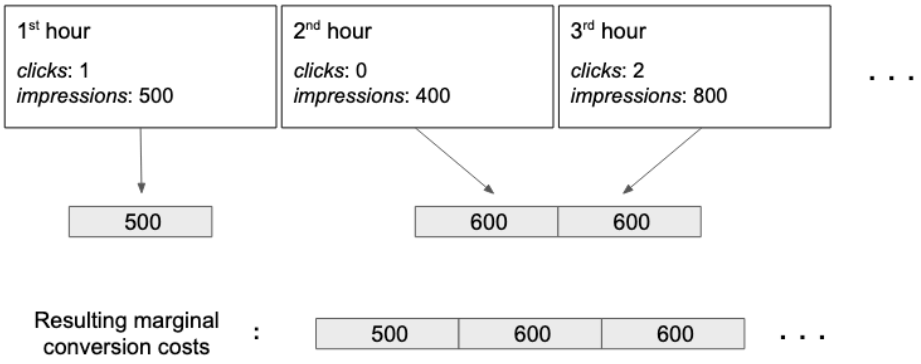

Figure 6.2: Instance building example - 1) in the first hour, we do not know exactly when the first conversion (click) happens, but we consider that marginal conversion cost is 500 in this situation; 2) the impressions from the second hour only sums for the next conversion, as there is no conversion in the second hour; 3) we divide the 1200 accumulated marginal cost between the 2 conversions that happen in the third hour.

ad impressions) extracted from different Facebook audiences in different periods, following the method shown in Figure 6.2. We ran the algorithms for S equals the number of slabs in the options (in the same instance, all the options have the same number of slabs). Below, we describe in more detail the instances.

- The first real instance is composed of 500 marginal costs for 5 different audiences, as shown in Figure 6.3.
- Similarly, the second instance is formed by 400 marginal conversions for 5 different audiences. See Figure 6.4.
- Finally, the third instance is composed of 300 marginal conversion for 3 different audiences. See Figure 6.5.

We would like to know how similar to exponential distributions the real marginal conversions should be. Naturally, as there are some limitations in the data extraction process to obtain the real instances, and because of the saturation effect mentioned before, the empirical distributions are not expected to be fully adherent to exponential distributions. We compared the *empirical cumulative distributions* (CDF) of the marginal conversion costs for each real instance with an exponential distribution having the same mean value. We could observe a reasonable exponential behavior for some options and instances (for example, option 5 in real instance 1 and option 3 in real instance 3). See figures 6.6, 6.7 and 6.8.

We compared the same algorithms adopted in the experiments with simulated instances. We ran the algorithms 20 times for each instance and we computed the average result for each of them.

Instance	OPT	BALGREEDY	**OFFBESTARM**	**ROUNDROBIN**	**UNIFORMINVEST**	**RANDOMARM**
1	26183	**28528**	33887	30422	29278	42072
2	17901	**18532**	22406	20367	20383	24936
3	71653	90681	**86645**	87675	91118	91443

Table 6.4: Algorithms comparison in real instances

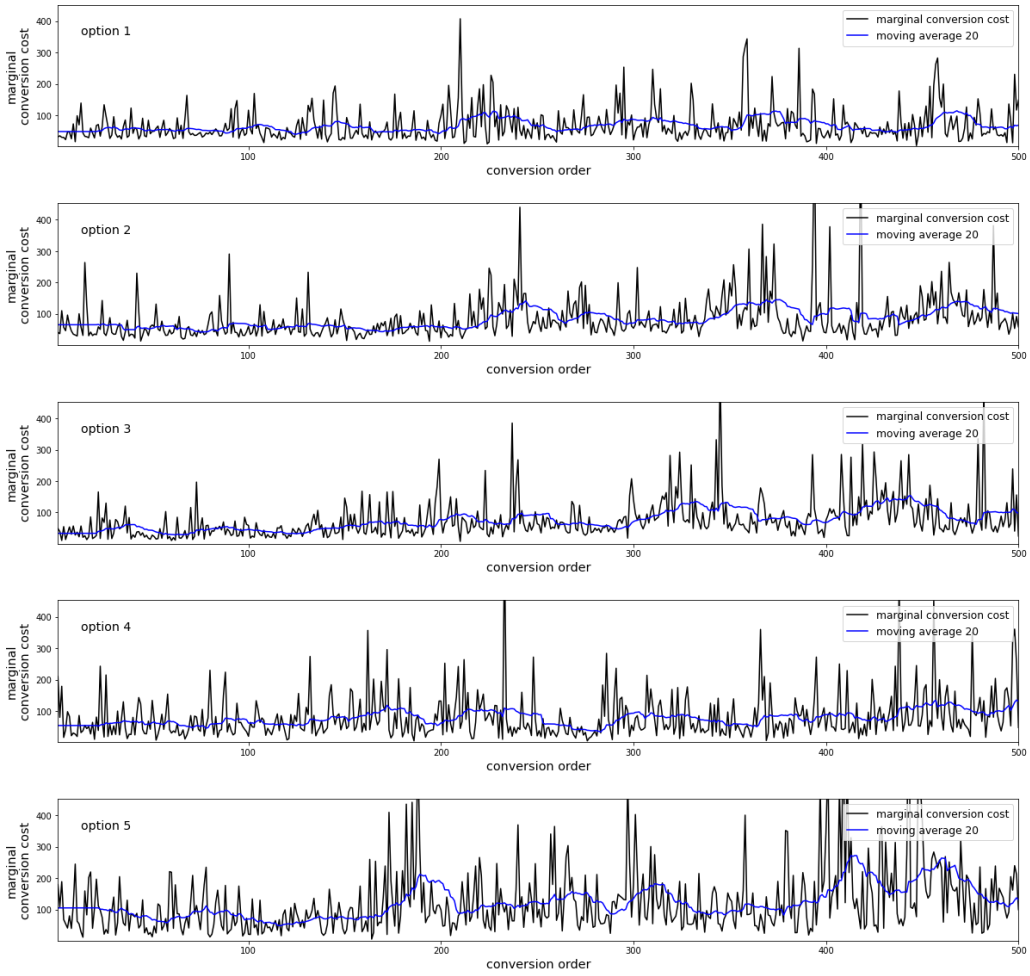

Figure 6.3: Real instance 1 - Sequence of marginal conversions for each of the 5 options. The blue line is the moving average of these values for a window of 20 data points.

Table 6.3 shows that balGreedy was the best in real instances 1 and 2. However, both roundRobin and offBestArm performed slightly better than balGreedy in real instance 3. In this same real instance, all the algorithms performed similarly (the cost for the best algorithm in this instance category was 5.25% lower than the worst one).

In instances 1 and 2, the saturation effect is more evident than in the instance 3, it is one of the reasons why the balGreedy performed better in the first 2 instances. Besides, the algorithms roundRobin and uniformInvest obtained good results in the first 2 instances too, because these algorithms regularly explore all the options, taking advantage of the cheap first conversions in all options.

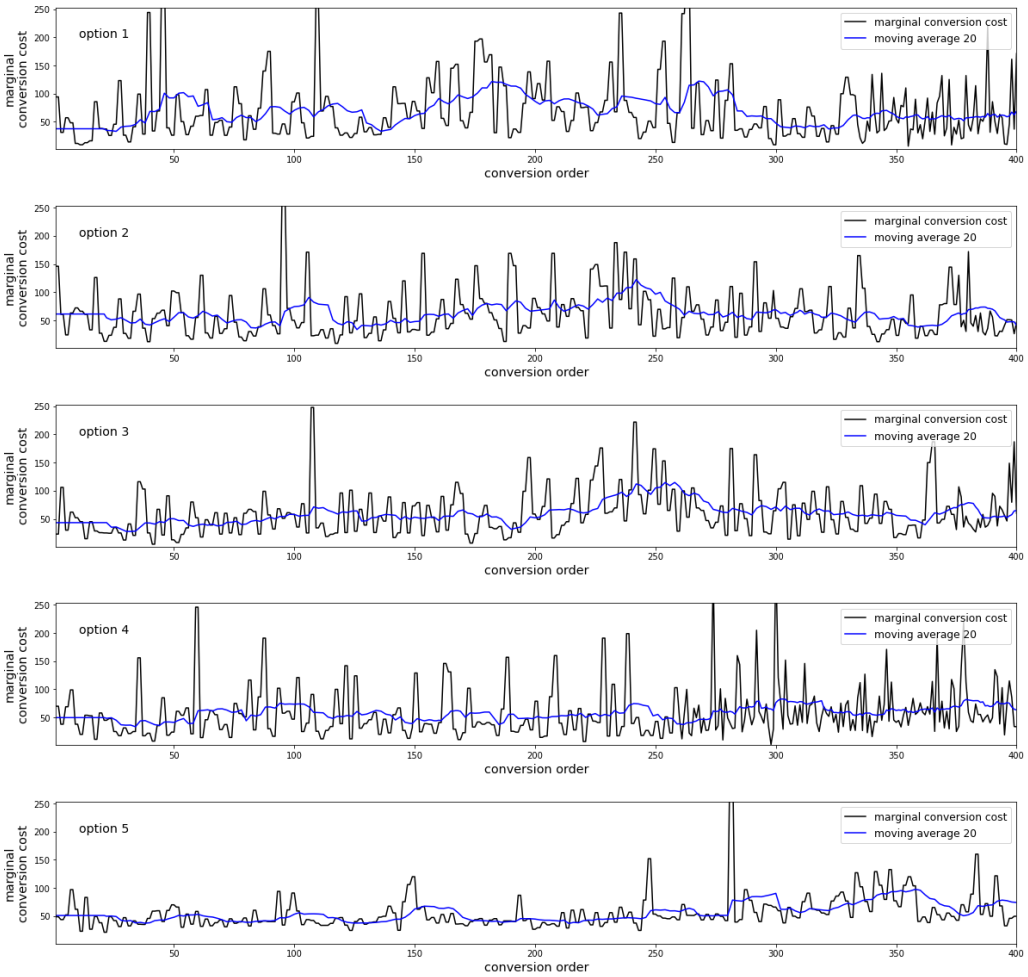

Figure 6.4: Real instance 2 - Sequence of marginal conversions for each of the 5 options. The blue line is the moving average of these values for a window of 20 data points.

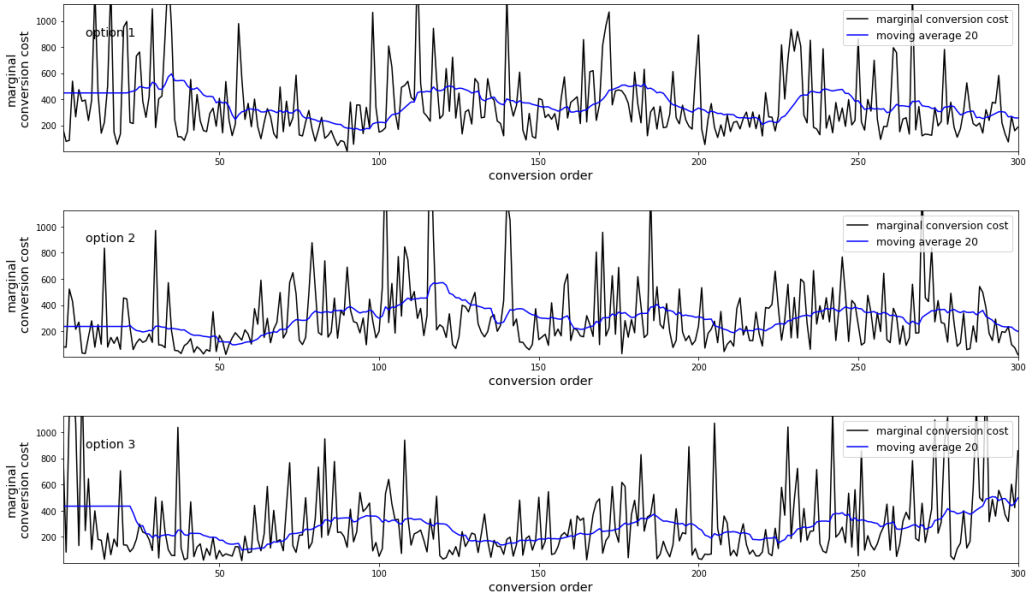

Figure 6.5: Real instance 3 - Sequence of marginal conversions for each of the 3 options. The blue line is the moving average of these values for a window of 20 data points.

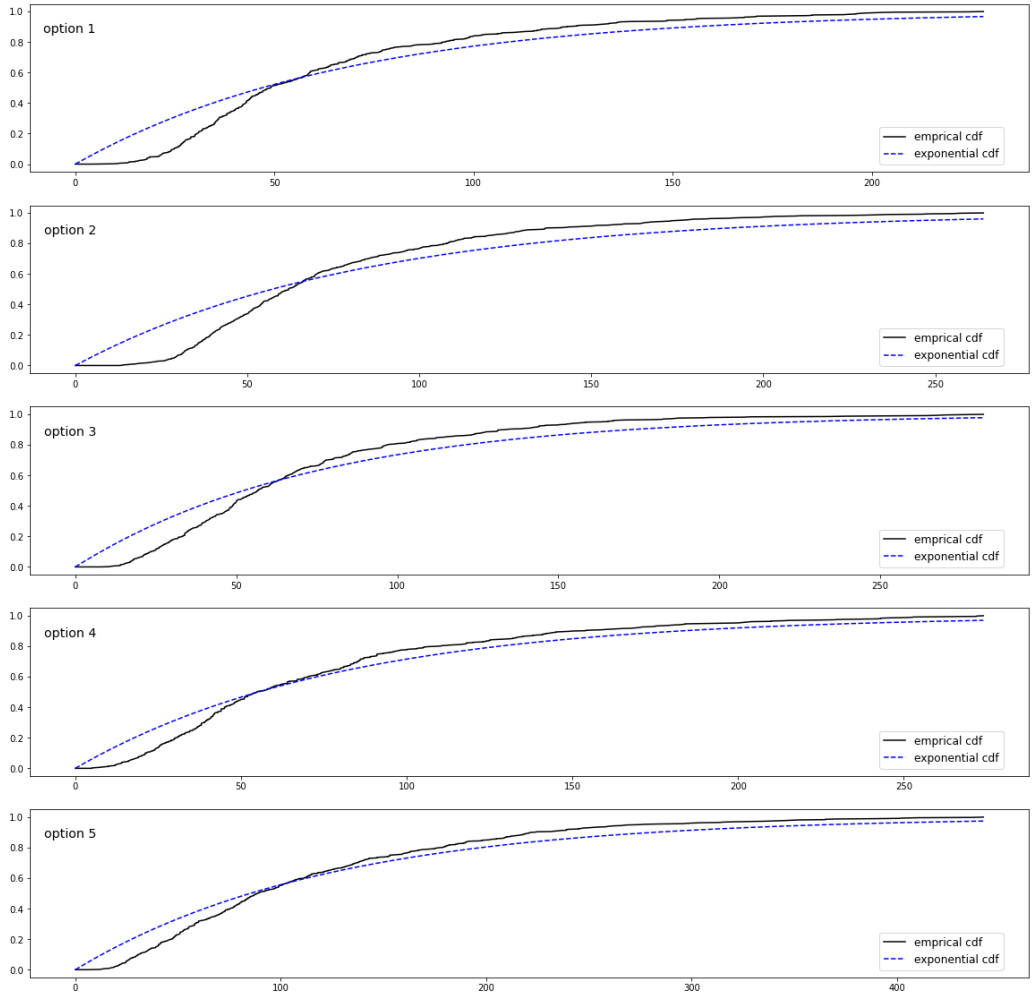

Figure 6.6: Real instance 1 - The empirical CDF of marginal costs distribution for the 5 options and the exponential distribution CDF with the same average.

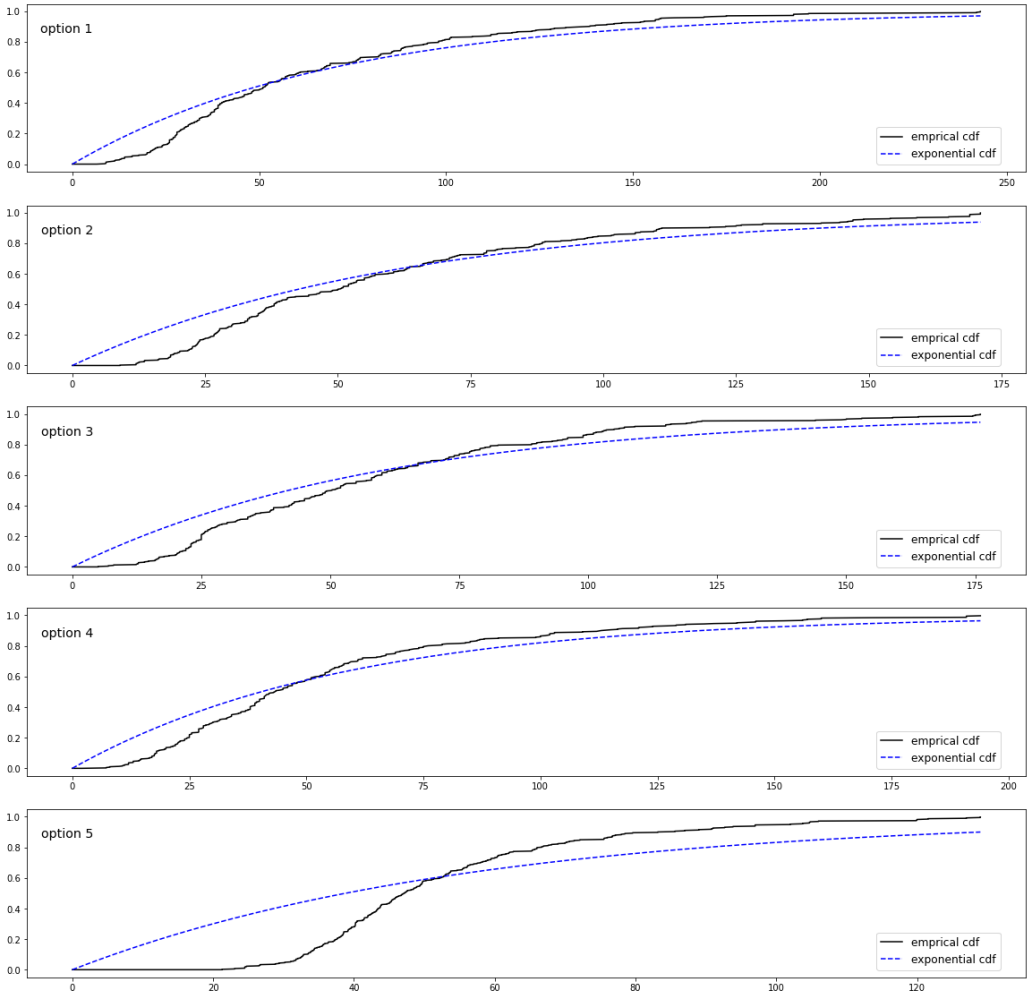

Figure 6.7: Real instance 2 - The empirical CDF of marginal costs distribution for the 5 options and the exponential distribution CDF with the same average.

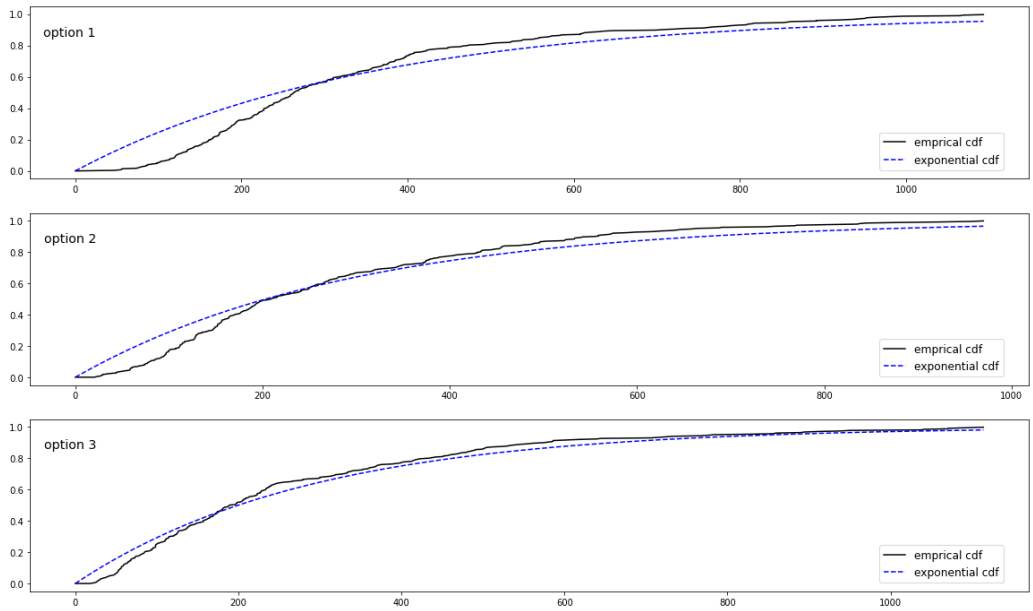

Figure 6.8: Real instance 3 - The empirical CDF of marginal costs distribution for the 5 options and the exponential distribution CDF with the same average.

7
Conclusions

In this work, we introduced the problem AdInvest, that models the budget allocation problem in marketing channels from the advertiser's point of view. We defined what should be an instance of this problem and we proved guarantees of the algorithm in many types of instance that were defined to model possible marketing channel behaviors. Although other works have also studied problems related to online optimization in marketing channels, we are not aware of any model that approached the advertiser's investment problem as proposed here.

The proven guarantees assure that the balGreedy policy is an efficient algorithm considering many types of deterministic and random instances, obtaining relative near OPT results in most of them. Among the types of instance considered in this is work, we highlighted those that model the saturation effect in marketing audiences, that is the natural loss of efficiency in a channel as the advertiser explores the audiences.

In addition, we computed computational experiments for synthetic and real data, in which the balGreedy obtained good results compared to the online alternative algorithms and even the offline policy offBestArm.

Bibliography

[1] Micah Adler, Phillip B Gibbons, and Yossi Matias. Scheduling space-sharing for internet advertising. *Journal of Scheduling*, 5(2):103–119, 2002.

[2] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *The Journal of Machine Learning Research*, 15(1):1111–1133, 2014.

[3] Deepak Agarwal, Rahul Agrawal, Rajiv Khanna, and Nagaraj Kota. Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–222, 2010.

[4] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

[5] Domagoj Bradac, Anupam Gupta, Sahil Singla, and Goran Zuzic. Robust algorithms for the secretary problem. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 32:1–32:26. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[6] Sébastien Bubeck and Nicolò Cesa-Bianchi. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122, 2012.

[7] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. Simple and scalable response prediction for display advertising. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(4):1–34, 2014.

[8] Ye Chen, Dmitry Pavlov, and John F Canny. Large scale behavioral targeting. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 209–218, 2009.

[9] Ye Chen and Tak W Yan. Position-normalized click prediction in search advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 795–803, 2012.

[10] David Maxwell Chickering and David Heckerman. Targeted advertising on the web with inventory management. *Interfaces*, 33(5):71–77, 2003.

[11] Dragos Florin Ciocan and Vivek Farias. Model predictive control for dynamic resource allocation. *Mathematics of Operations Research*, 37(3):501–525, 2012.

[12] Ying Cui, Ruofei Zhang, Wei Li, and Jianchang Mao. Bid landscape forecasting in online ad exchange marketplace. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 265–273, 2011.

[13] Nikhil R. Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A. Wilkens. Near optimal online algorithms and fast approximation algorithms for resource allocation problems. In *EC*, 2011.

[14] Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. Online allocation with traffic spikes: Mixing adversarial and stochastic models. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 169–186, New York, NY, USA, 2015. ACM.

[15] Jon Feldman, Monika Henzinger, Nitish Korula, Vahab S. Mirrokni, and Clifford Stein. Online stochastic packing applied to display ad allocation. In *ESA*, 2010.

[16] Arpita Ghosh, Benjamin Rubinstein, Sergei Vassilvitskii, and Martin Zinkevich. Adaptive bidding for display advertising. In *Proceedings of the 18th international conference on World wide web*, pages 251–260, 2009.

[17] Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. In *Conference on Learning Theory, COLT 2019, 25-28 June 2019, Phoenix, AZ, USA*, pages 1562–1578, 2019.

[18] Patrick Hummel and R Preston McAfee. Loss functions for predicted click-through rates in auctions for online advertising. *Journal of Applied Econometrics*, 32(7):1314–1328, 2017.

[19] Thomas Kesselheim, Robert D. Kleinberg, and Rad Niazadeh. Secretary problems with non-uniform arrival order. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 879–888, 2015.

[20] Bernhard H. Korte, László Lovász, and Rainer Schrader. *Greedoids*. Berlin, 1991.

[21] Nitish Korula, Vahab S Mirrokni, and Morteza Zadimoghaddam. Bicriteria online matching: Maximizing weight and cardinality. In *International conference on web and internet economics*, pages 305–318. Springer, 2013.

[22] Subodha Kumar, Varghese S Jacob, and Chelliah Sriskandarajah. Scheduling advertisements on a web page to maximize revenue. *European journal of operational research*, 173(3):1067–1089, 2006.

[23] Thodoris Lykouris, Vahab Mirrokni, and Renato Paes Leme. Stochastic bandits robust to adversarial corruptions. In *STOC 2018*, 2018.

[24] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, 2013.

[25] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22–es, 2007.

[26] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995.

[27] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. Bid optimizing and inventory scoring in targeted online advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 804–812, 2012.

[28] Yevgeny Seldin and Aleksandrs Slivkins. One practical algorithm for both stochastic and adversarial bandits. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 1287–1295, 2014.

[29] John Turner. The planning of guaranteed targeted display advertising. *Operations research*, 60(1):18–33, 2012.

[30] Erik Vee, Sergei Vassilvitskii, and Jayavel Shanmugasundaram. Optimal online assignment with forecasts. In *Proceedings of the 11th ACM conference on Electronic commerce*, pages 109–118, 2010.

[31] Wush Chi-Hsuan Wu, Mi-Yen Yeh, and Ming-Syan Chen. Predicting winning price in real time bidding with censored data. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1305–1314, 2015.

[32] Jian Xu, Xuhui Shao, Jianjie Ma, Kuang-chih Lee, Hang Qi, and Quan Lu. Lift-based bidding in ad selection. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[33] Weinan Zhang, Shuai Yuan, and Jun Wang. Optimal real-time bidding for display advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1077–1086, 2014.

[34] Weinan Zhang, Tianxiong Zhou, Jun Wang, and Jian Xu. Bid-aware gradient descent for unbiased learning with censored data in display advertising. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 665–674, 2016.

[35] Julian Zimmert and Yevgeny Seldin. An optimal algorithm for stochastic and adversarial bandits. In *The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS 2019, 16-18 April 2019, Naha, Okinawa, Japan*, pages 467–475, 2019.