POLITECNICO DI MILANO
DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA
DOCTORAL PROGRAMME IN INFORMATION TECHNOLOGY

MACHINE LEARNING ALGORITHMS FOR THE OPTIMIZATION OF INTERNET ADVERTISING CAMPAIGNS

Doctoral Dissertation of:
Alessandro Nuara

Supervisor:
Prof. Nicola Gatti

Tutor:
Prof. Raffaela Mirandola

The Chair of the Doctoral Program:
Prof. Barbara Pernici

2020 – XXXIII Cycle

Abstract

Online advertising revenue grew to 124.6 billion dollars in 2019, showing a 15.9% increase over the previous year. The opportunities provided by this market attracted wide attention of the scientific community as well as the industry that requires automatic tools to manage the main processes involved in this market. For this purpose, *Artificial Intelligence* can play a crucial role in providing techniques to support both publishers and advertisers in their tasks. In this thesis, we study the Internet advertising campaign optimization problem from the advertiser' point-of-view. An advertising campaign is composed of a set of *sub-campaigns* that differ for the ad (*e.g.*, including text or images), target (*e.g.*, keywords, age, interests), or channel (*e.g.*, search, social, display). In pay-per-click advertising, to get an ad impressed, the advertisers take part in an auction carried out by the publisher, in which they set a bid and a daily budget for each sub-campaign. The bid represents the maximum amount of money the advertisers are willing to pay for a click, whereas the daily budget is the maximum spend in a day for a sub-campaign. The advertisers' goal is to create a set of sub-campaigns and for each of them set the bid/daily budget values that maximize the revenue under budget or return-on-investment constraints. This optimization problem is particularly challenging, as it includes many intricate subproblems. In this work, we study four different settings and we propose algorithms specifically crafted for each of them.

First, we study the joint bid/budget optimization problem and we propose an online learning algorithm that combines *Gaussian Processes* estimation with *Combinatorial-bandit* techniques to find the optimal bid/budget values. We define four variants of our method and we derive high probability regret bounds of the order of $O(\sqrt{T})$, where T is the learning time horizon.

Second, we introduce the novel framework of *safe* bid optimization where the goal is to maximize the revenue while satisfying with high-probability some budget and return-on-investment constraints. We propose two algorithms, namely GCB and GCB_{safe} that differ on how they balance the revenue maximization and constraints violation. The first provides a sublinear bound on the regret but rarely satisfies the ROI constraints. The second keeps the probability of violating the constraints under a confidence δ at the cost of a linear pseudo-regret bound.

Third, we study the problem of optimizing the target, the bids and the budgets of an advertising campaign. We formulate it as a Learning from Logged Bandit Feedback (LLBF) problem, and we propose an offline algorithm that computes a tree expansion of the target space to learn the partition that maximizes the revenue.

Last, we study the problem of the sub-campaigns interdependence in the advertising campaign optimization. We provide an algorithm that employes Granger Causality to learn the campaign interdependence model from past data, Gaussian Processes to predict the sub-campaigns performance, and a dynamic programming procedure to compute the bid/daily budget allocation. Theoretical guarantees on the loss of the algorithm w.r.t. the clairvoyant solution are also proven.

For all the problems mentioned above, we evaluate our approaches in both synthetic and real-world scenarios showing their superiority if compared with baselines and with the previous human campaign management.

Contents

CHAPTER 1

Introduction

In the last decades, the rise of the Internet has drastically changed our way of life. Many of our daily activities like shopping, networking or entertainment switched from real to online life, bringing us to spend always more time on the Internet. According to recent analysis [1], the number of people using the Internet is about 4.54 billion and the average Internet user spends, each day, 6 hours and 43 minutes online. This extensive usage involving different channels and devices makes the Internet a goldmine for marketers that now have the opportunity to show advertisements quickly and to a broad but specific audience. Thanks to these capabilities, in the first quarter of 2020, despite the crisis due to the COVID-19 pandemic, the advertising revenues grew to 31.4 billion dollars with an increase of 12% from the previous year [27]. Accordingly, in the last two decades, online advertising has been given wide attention by the scientific community and by the industry that requires automatic tools to support publishers and advertisers in their tasks. In this thesis, we study the problem of optimizing an Internet advertising campaign from the advertiser' point of view.[1] This problem requires addressing multiple sub-problems and involves both creative and technical tasks. As the first step, advertisers have to select the

[1]This research has been funded by the Italian media agency *Mediamatic*.

right target of their campaign by identifying the typology of users that are more likely to be interested in their brand and products. For each target portion, they have to create some sub-campaigns belonging to different advertising channels (*e.g.,* search, social, video). All these sub-campaigns have distinct objectives that go through generating awareness about the brand to inducing the user to complete a purchase. Still, all have the common goal of maximizing the final advertiser revenue. Furthermore, for each of them, advertisers have to create an *ad* choosing the most effective message and format to promote their product for that specific target. Finally, they have to adjust some variables impacting the economic management of the campaign. For instance, in pay-per-click advertising, to get an ad impressed, the advertisers take part in an auction carried out by the publisher, in which they set, every day, a bid and a daily budget for each sub-campaign [46]. The bid represents the maximum amount of money the advertisers are willing to pay for a single click, whereas the daily budget is the maximum spent in a day for a sub-campaign. If we consider that an advertiser has to manage several campaigns, each of them composed of dozens of sub-campaigns, the number of variables that an advertiser has to adjust on advertising platforms every day becomes considerably large. As a consequence, effective and efficient execution of these tasks is not feasible for humans and the adoption of automatic tools is needed. For this purpose, Artificial Intelligence (AI) can play a crucial role in providing techniques able to exploit a huge amount of data to model users' behavior and support marketers in the optimization process. The goal of this thesis is to provide a set of AI algorithms able to solve the main problems an advertiser has to face in the advertising campaigns optimization. We focus on four different problems that have the same general goal of maximizing the advertiser revenue but exhibit different features.

In the first problem, we analyze the setting in which the advertiser has to manage a fixed set of sub-campaigns and has to set for each of them a bid/budget pair to maximize the revenue while satisfying an overall daily budget constraint. We propose an algorithm, named AdComb, that addresses this problem in an online fashion. Our approach combines Gaussian Processes estimation and a dynamic programming algorithm to predict sub-campaigns performance and subsequently solve a combinatorial optimization problem. We theoretically analyze the properties of our approach and provide a sub-linear regret bound for four flavors of our method that differ for the prediction model and exploration strategy they adopt. Then, we empirically evaluate our approach in synthetic settings generated from *Yahoo!* Webscope $A3$ dataset and in a real-world campaign showing its superiority compared to human management.

After that, we study the specific setting in which the advertiser has to optimize the bid to maximize the revenue and satisfy some spend and return-on-investment constraints for all the advertising period. Furthermore, in this setting, the advertiser is not allowed to limit the daily spend of a sub-campaign by setting a daily budget on platforms. In particular, we show that the problem of finding revenue-maximizing bids satisfying budget and ROI constraints cannot be approximated within any factor unless $\mathsf{P} = \mathsf{NP}$. However, we demonstrate that it is possible to design a pseudo-polynomial-time algorithm to find an optimal solution for the problem. Furthermore, we show that no online learning algorithm can violate the ROI constraints less than a linear number of times while guaranteeing a sublinear pseudo-regret. We propose the GCB algorithm that provides a sub-linear bound on the regret but rarely satisfies the ROI constraints. Then, we propose the $\mathsf{GCB}_{\mathtt{safe}}$ algorithm to keep the probability of violating the ROI constraints under a predefined confidence δ at the cost of a linear pseudo-regret bound. Finally, we experimentally evaluate the performances of GCB and $\mathsf{GCB}_{\mathtt{safe}}$ in terms of pseudo-regret/constraint-violation tradeoff, and we analyze the algorithms' sensitivity.

We also consider the problem in which the advertiser has to find the best user targets (a.k.a. *contexts*) that a media agency can use in a given Internet advertising campaign. We address it as an offline problem where the goal is to find the optimal set of sub-campaigns and the associated bid/budget pairs that maximize the revenue. More specifically, we formulate the problem of target optimization as a Learning from Logged Bandit Feedback (LLBF) problem and we propose the TargOpt algorithm, which uses a tree expansion of the target space to learn the partition that efficiently maximizes the campaign revenue. Furthermore, since the problem of finding the optimal target is intrinsically exponential in the number of the features, we propose a tree-search method, called A-TargOpt, and two heuristics to drive the tree expansion, aiming at providing an anytime solution. Finally, we present empirical evidence, on both synthetically generated and real-world data, that our algorithms provide a practical solution to find effective targets for Internet advertising.

Last, we consider the challenging setting in which the sub-campaigns are interdependent each other, i.e., the impressions and clicks generated by a sub-campaign can affect the performance of other sub-campaigns directed to the same users. We present an algorithm, called IDL, that, employing Granger Causality and Gaussian Processes, learns the sub-campaigns interdependency model from past data, and returns an optimal stationary bid/daily budget allocation. Finally, We provide theoretical guarantees on the loss of the algorithm w.r.t. the clairvoyant solution and we show some empirical evidence of the superiority of the proposed algorithm in both realistic and real-world settings when compared with approaches that do not take into account these interdependency relationships.

Thesis Structure

In this section, we provide the structure of the thesis and we briefly describe the content of each chapter.

Chapter 2. In this chapter, we provide an overview of the digital advertising market. We firstly describe the mechanism and the main players involved in this market. Then, we provide the main definitions and preliminary notions of this thesis. We introduce the marketing funnel model that describes the process to lead a user to complete a purchase. Then, we describe the goals and peculiarities of the main advertising channels that can be exploited to sponsor a product. Finally, we provide the main KPIs adopted to measure the campaigns' performance.

Chapter 3. In this chapter, we provide an overview of the main scientific works related to the internet advertising optimization problem, focusing on the bid optimization, budget optimization, and targeting optimization problems.

Chapter 4. In this chapter, we firstly describe the main tasks that an advertiser has to perform to manage an advertising campaign. Then, we provide a general formulation of the advertising campaign optimization problem that includes the sub-problems associated with the tasks mentioned above.

Chapter 5. In this chapter, we address the joint bid/budget optimization problem. We firstly provide an online formulation of this problem and we present a combinatorial bandit approach to address it. Then, we provide a theoretical analysis and an exhaustive experimental analysis of our approach in both synthetic and real-world settings. A preliminary version of the results is provided in [41], while an extended version including both theoretical and empirical results is provided in [42].

Chapter 6. In this chapter, we address the online safe bid optimization with return-on-investment constraints problem. We provide a risk-averse approach that balances optimization of the revenue and risk in the ROI and spend constraints violation. Then, we provide a theoretical analysis of both the offline optimization procedure and the regret of the proposed approach. The results of this work are under review and presented in [12].

Chapter 7. In this chapter, we address the target optimization problem. We provide an algorithm that uses a tree expansion of the target space to learn the partition that efficiently maximizes the campaign revenue. The results of this work are published in [21].

Chapter 8. In this chapter, we study the problem of the interdependency among sub-campaigns belonging to different advertising channels. We provide an algorithm that first detects the interdependency relationships among sub-campaigns and then exploits them to find the optimal bids and budgets that maximize the revenue. The results of this work are published in [40].

Chapter 9. Finally, in the last chapter, we provide the overall observations and we point out possible future works of this research.

CHAPTER 2

Digital Advertising

In this chapter, we introduce the main definitions and basic concepts of the advertising scenario involved in the problems we address in this thesis. Firstly, we provide an overview of the Digital Advertising market presenting the main players and mechanisms. Secondly, we describe the main tasks involved in the setup and management of an advertising campaign. Then, we introduce the marketing funnel model that describes how the advertising campaigns can interact with each other to reach a common goal. Finally, we provide an overview of the main advertising channels.

Overview

Digital Advertising is nowadays one of the main methods to sponsor a product or a service. In the last decades, the revenue of the digital advertising market increased exponentially while traditional advertising channels like TV or newspapers have not shown a significant increase.

The success of this tool can be motivated by several reasons. Firstly, fine-grained data collected for each user by third parties allow marketers to target their advertisements to a broad but specific audience. Secondly, for each of these users, they can create ads with different formats (*e.g.*, video, banner, textual, etc.) that can be tailored to user features. Another important feature introduced by Digital Advertising is the possibility to exactly measure the performance of a campaign. This is a crucial issue for marketers who have to understand which of the advertising investments are profitable and which are not. Finally, advertising platforms allow low-level management of the campaigns and to perform adjustments that are effective in real-time. Since the growth of the opportunities offered by this market, year by year, the digital advertising ecosystem is growing significantly involving new players that make its structure more complex. However, from these, we can identify the following three representative categories:

- **Users** are people surfing on the net that can potentially buy a product or more generally perform an action (e.g., compile a form, or a newsletter subscription).
- **Advertisers** aim at sponsoring or selling a product to users. Their goal is to show messages only to potentially interested users that can perform an action (e.g., buy a product) after clicking on an ad.
- **Publishers** (e.g., Google, Facebook, Amazon) are the intermediaries of this market. While a user surfs on a web page, they allow advertisers to show their ads on some slots. The choice of which ad to show for each slot is performed by running auctions among advertisers. For each of these auctions, an advertiser has to specify a bid value that corresponds to the maximum price she is willing to pay for a user click.

Campaign Setup and Management

The setup of an advertising campaign consists of creating a set of *sub-campaigns*, each of which is associated with a specific target and with an ad [1]. More precisely, for each sub-campaign, advertisers have to specify some features indicating the geographic area, the age, the interests, and users' information. If the campaign is promoted through a search engine platform, they have to indicate a group of keywords that allow targeting users by their intent. To get an ad impressed, an advertiser has to specify a bid value indicating the maximum price she is willing to pay for a user click and a daily budget indicating how much she is willing to spend on a day. Once a user visits a web-page or performs a search query, the publisher runs an auction among advertisers and retrieves all the ads that match that

[1]Though platforms allow associating multiple ads to the same sub-campaign, for the sake of simplicity, in what follows, we assume that each sub-campaign is associated with a single ad.

query. These ads are ranked according to a score computed as the product of the bid and the ad click-through rate, while the payment is computed depending on the auction mechanism.

Once an advertiser creates a sub-campaign, she has to set the payment method to adopt. Advertising platforms usually provide different models, such as pay-per-mille impressions (PPM), pay-per-click (PPC), and pay-per-action (PPA). In PPM campaigns, advertisers are charged for impressions. In PPC campaigns, advertisers are charged after a click, and in PPA campaigns advertisers are charged after a user conversion. The PPC is the most used method [46] and in the scenarios studied in this thesis we assume the adoption of this model.

Finally, during the whole advertising period, the advertiser has to adjust some economic variables that significantly affect the performance of the sub-campaigns. Though new platforms allow regulating many economic variables, here we focus on two key variables:

- **Bid**. The bid is the maximum price the advertiser is willing to pay for a user's click. A proper setting of this value is crucial to balance return-on-investment and volumes that can be achieved over a sub-campaign. Increasing the bid lead to win a high number of auctions (high volumes) but also to increase the cost per click. Low bid values, instead, allow achieving a higher return-on-investment but lower volumes.

- **Daily budget**. The daily budget is the maximum amount of money that the advertiser is willing to spend on a day for a sub-campaign. A proper setting of this value is crucial to optimally allocate the overall budget over the sub-campaigns and satisfy campaign constraints.

Setting these two variables optimally requires to tune them jointly. If we consider that an advertiser has usually to manage hundreds of sub-campaigns, the space of possible bid/budget combinations is considerably large and make this task hard to be performed by hand.

Marketing Funnel Model

The marketing funnel models the process that brings customers from the awareness about the product to the purchase decision (see Figure 2.1). The idea behind this model is that buyers have to be led to the purchase by crossing different stages. Ideally, a marketer has to design an advertising campaign able to guide the customers for all the following stages:

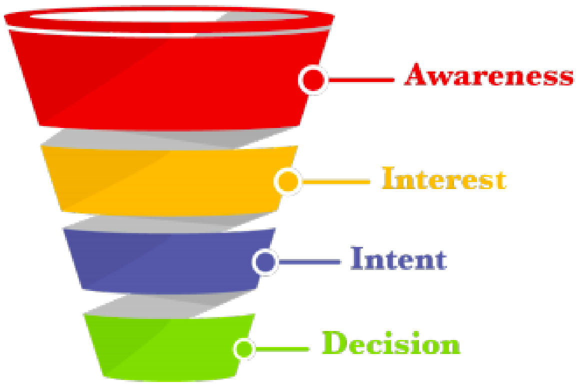


Figure 2.1: *Marketing funnel model.*

- **Awareness.** At this stage, the user becomes aware of the brand or company. Potential customers are usually brought to this stage through social or display campaigns that make the users familiar with the brand.
- **Interest.** At this stage, the user starts to be interested in what the company offers and she is aware of the product and services provided by the company. Here, the content and the target of the advertising starts to be more specific.
- **Intention.** At this stage, the user has already shown his intention of buying. For instance, after a view of a demo or when the product is on the shopping cart of an e-commerce website.
- **Decision.** This is the last stage of the marketing process. At this point, the user has to be led to perform the purchase decision. Usually, the sub-campaigns of this stage are the keyword-based or the retargeting ones.

Advertising Channels

An advertising campaign usually operates on different marketing channels each with a different goal. For instance, the same ad can be shown on a search engine (e.g., on Google), on a social network (e.g., on Facebook) or on a video (e.g., on YouTube). The so-called *multi-channel advertising* allows companies to be where customers are and to collect more precise and complete information about their interests and their behavior. Moreover, in such a way, companies allow users to generate conversions on the channel they are comfortable with. The challenge of *multi-channel advertising* is to create the right mix of sub-campaigns that operate at each level of the marketing funnel and that influence each other improving the overall performance. For instance, sponsoring a product on a social network to a broad audience will increase the chance that one of the targeted users will search that product on a search engine and then complete the purchase through the search channel. As a consequence, sponsoring a product across multiple channels make the management of the campaign more complex. For instance, the monitoring and the analysis of the performance become more challenging since it is not always possible to measure the real contribution of each sub-campaign on the overall performance. However, nowadays an internet advertising campaign is usually designed by taking into account all these issues and in such a way to exploit the opportunities provided by each channel. We can distinguish three main advertising channels that usually operate at different levels of the marketing funnel:

- **Search**. This advertising channel consists of showing advertisements with the search engine results returned by a user query. The advertisements have to be targeted to terms (a.k.a. keywords) such that they are related to the user search and therefore more likely to be tailored to her intent. This advertising channel allows advertisers to direct their ads to users who intent to complete a purchase through their search query. For this reason, the search sub-campaigns are ideally placed to the last stage of the marketing funnel since they have a high chance to lead the user to generate a conversion.

- **Social**. This advertising channel consists of showing ads on social network pages or apps. The advertisement messages include posts, videos, banners, and social networks pages. This method allows advertiser to target their ads to a very specific audience since data collected by social networks allow inferring detailed information about user interests and behavior. Social sub-campaigns are usually placed at middle or high levels of the marketing funnel. The goal of these campaigns, indeed, is usually to target users that can be potentially interested in a product based on their behavior or interests.

- **Display**. This advertising channel consists in showing graphic advertisiments on web-pages through banners, images, videos, etc. The goal of display advertising is usually to diffuse brand messages to increase the awareness of users about a product or a company. For this reason, these campaigns are usually placed on the top in the marketing funnel though there are some exceptions. For instance, re-targeting sub-campaigns, placed at the bottom levels of the marketing funnel, usually consists of display ads that remind users to complete a purchase.

Key Performance Indicators of an Advertising Campaign

Here, we introduce the main definitions and key performance indicators commonly adopted to measure the campaigns' performance.

- **Impressions.** An impression occurs when an advertisement message is shown on a web page, i.e., each time the advertiser wins an auction. It is worth noting that the generation of an impression does not necessarily imply that the user has seen the message. For this reason, some platforms introduced the *view* event to guarantee that the user has seen the message. The impressions generation is usually a goal of display and social sub-campaigns and they have an impact on brand awareness rather than leading users to complete a purchase. The payment method of these campaigns is usually CPM (cost-per-mille impressions).

- **Clicks.** Clicks occur each time a user clicks on an ad after an impression. This KPI is usually considered more relevant than impressions since indicates more engagement of the user. The click generation is usually performed by social and search sub-campaigns placed at the medium and low levels of the marketing funnel. The payment method adopted for these campaigns is usually pay-per-click (PPC).

- **Conversions.** A Conversion occurs when a user performs an action after clicking the ad. For instance, a purchase or a newsletter subscription can be considered conversions if they occur after clicking an ad. It is worth mentioning that the delay between a click and a conversion can be large (from 1 to 14 days), making the measurement of the conversions affected by attribution issues. The generation of conversions is usually performed by search sub-campaigns that operate at the low level of the marketing funnel. The most common payment method for these sub-campaigns is pay-per-click (PPC) but also pay-per-action (PPA) is often adopted.

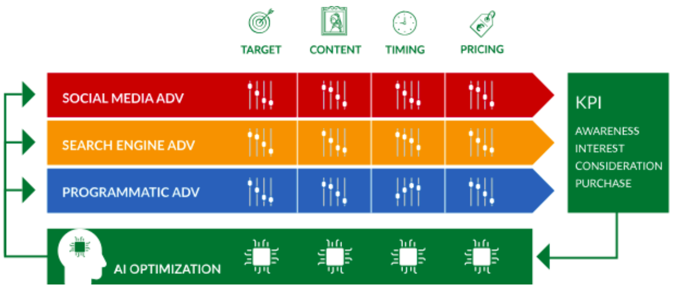


Figure 2.2: *Schema of the optimization of an advertising campaign involving different advertising channels.*

CHAPTER 3

State of the Art

The internet advertising campaigns optimization problem has been widely addressed in the scientific literature. In this chapter, we focus on works that cover the following three main topics that are more related to the problems we address in this work:

- Bid and budget optimization;
- Targeting optimization;
- Advertising channels interdependence.

Bid and Budget Optimization

The bid optimization and budget optimization are daily tasks that advertisers have to perform with at least daily frequency and that have a strong impact on the campaigns' performance. For this reason, these two problems are largely addressed in the scientific literature. Some works address both these two problems jointly ([66, 53]), while some other works address the bid optimization ([37, 18, 62, 56, 58, 65, 34, 61, 64, 60, 32]) and budget optimization separately ([28, 36, 5]).

Zhang et al. [66] address for the first time the joint optimization of the bid and the daily budget. They approach it as a constraint optimization problem where they adopt a sequential quadratic programming algorithm whose goal is maximizing the expected revenue. However, their approach show some limitations to its applicability since it requires to fit ad ranking models from data (*e.g.*, the position of the ad for every display and click) not always available to advertisers.

Thomaidou et al. [53] separate the optimization of the bid from that one of the daily budget and use a genetic algorithm to optimize the budget and subsequently applying some bidding strategies.

Geyik et al. [23] propose a budget optimization algorithm that allocates the budget starting from the top-level to the low-level campaigns according to their performance and their capability on spending the allocated budget. Finally, they examine different attribution models and they show that adopting a multi-touch attribution model leads to better performance.

Italia et al. [28] address the problem of optimizing the budget of advertising campaigns for events. They propose an algorithm that exploits models to predict the behavior of different populations of users and a backward induction algorithm to define the budget allocation.

Archak et al. [7] propose an algorithm for budget optimization that exploit an MDP that models the interaction among advertising individuals and a LP algorithm to solve the optimization problem.

The bid optimization problem has been addressed with different approaches and in different settings. Online learning approaches with regret guarantees are known only for the restricted cases with a single campaign and a budget constraint over all the length of the campaign without temporal deadlines. However, this last assumption rarely holds in real-world applications where, instead, results are expected by a given deadline. In particular, Ding et al. [18] and Xia et al. [62] work on a finite number of bid values and exploit a multi-armed bandit approach, whereas the approach proposed in Trovò et al. [56] deals with a continuous space of values for the bid and shows that assuring worst-case guarantees leads to the worsening of the average-case performance.

Several works study this problem in a real-time setting, namely Real-Time Bidding (RTB), where the advertiser exploits users' information in real-time to set a different bid for each auction. These techniques usually take into account budget [58, 65, 34, 61], cost-per-click [64, 60] or cost-per-action constraints [32].

Targeting Optimization

In recent years, many platforms, especially the ones belonging to the display and social channels, increased the amount and the quality of the information collected for each user. Since this new data availability and since the interest of the advertising big players on exploiting these data, also the scientific community put its effort into this topic and proposed several approaches that exploit such fine-grained information to increase the campaigns performance. However, it is worth to mention that while most of these works assume to track the activity of every single user, in our work, we assume that these punctual data are not always available to advertisers and therefore we resort to aggregated data to face this problem.

The targeting optimization problem has been addressed, for the first time, in [33] where authors propose a privacy-preserving approach that exploits the search query and the landing page URL to understand the short-term interests of a user in a certain topic and browsing information to understand his long-term interests. They then exploit both information to customize the advertisement content.

Other works concern the analysis and classification of users depending on their behavior on social networks [45], online shopping [44], and searching [6]. Provost et al. [45] propose a method to predict users' interests by observing the social networks pages they visit. Their approach leverages on the idea that users that like the same social network pages are more likely to buy the same products. Based on this assumption, they build a network whose edges are affinity values updated each time two users visit the same page.

Perlich et al. [44] adopt a multistage transfer learning approach to identify the users that are more likely to complete a purchase after a view of an advertisement.

Yan et al. [63] propose an approach for assigning ads to users and increase the CTR. Their method defines a segmentation of the users based on behavioral targeting under the assumption that users with the same web-browsing behavior also interact with advertisements similarly.

Other relevant related works concern the impact of timing ([10], [57]) (i.e., at which hour of the day an ad is displayed) on banner ads.

Bleier and Eisenbeiss [9] analyze how the performance varies depending on the timing, ad personalization, and the user position in the funnel model. They show that highly customized ads are more performing if shown to users that visited the online store recently. Conversely, the performance of these ads decreases as the time from the last visit increases.

Cheng and Cantú-Paz [15] provide a method to predict the user click probability by defining a parametric model whose features are divided in two sets. The first set includes demographic information combined with personal information (e.g., age, gender, status), while the second set includes information related to user browsing behavior.

Advertising Channels Interdependence

One of the more challenging problems that an advertiser has to address during the setup of an advertising campaign is how to define a campaign structure that involves all level of the marketing funnels and that exploit the advertising channels' potentialities. Indeed, different channels deeply affect each other's performance as Internet users regularly surf from one to another. For this reason, this problem raised the interest of the marketing literature and many works studied how advertising channels influence each other and how such interdependence can be exploited to increase the performance.

For instance, Lewis and Nguyen [35] provide empirical evidence that display advertising increases the search activity of the advertised brand for some days after the display ads visualization. Interestingly, they also show through an analysis conducted with Yahoo! Search data that display ads also increase the search activity of the competitors.

Kireyev et al. [30] show a similar result between the display and search advertising by using the Granger Causality test. The authors also show that this interdependence usually induces delayed dynamics, e.g., an increase in the display advertising impressions can lead to an increase in the conversions of search ads with a delay of some days.

Howard and Sheth [26] analyze the effect of display advertising on a user belonging to high and medium stages of the marketing funnel. More particularly, they show that display advertisements shown to registered users are more effective than the ones shown to users that previously visited the site without creating an account. Finally, Dinner et al. [19] authors show that online advertising can even affect the performance of (traditional) non-online advertising and they highlight that ignoring these cross effects leads to a miscalculation on the effectiveness of online advertising.

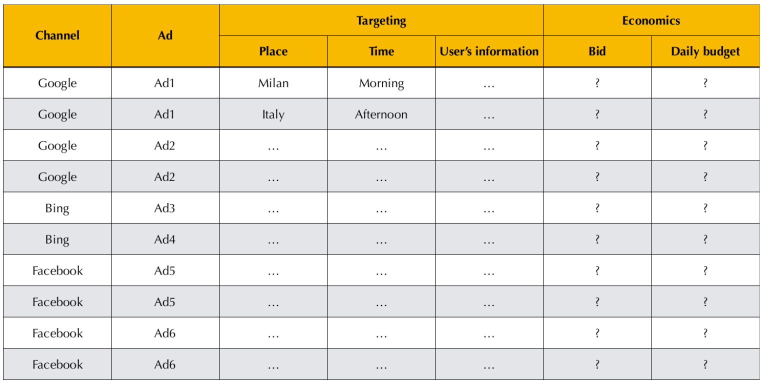
CHAPTER 4

Advertising Campaign Optimization Problem

In this chapter, we define the goal of our work providing a general formulation of the optimization problem we address in this research. Besides a general description of the optimization problem, we introduce the particular settings and sub-problems that will be analyzed in the following chapters of this thesis.

Motivation

The optimization of an internet advertising campaign involves the execution of multiple tasks. Firstly, an advertiser has to identify the more promising audience of the product or service to be sponsored and partition it in different portions. For each of these portions, she has to create a set of sub-campaigns, each with different ads, operating on different advertising channels (e.g., Google, Facebook, banners etc.) and at different levels of the marketing funnel. Finally, for each sub-campaign, at each day, she has to set a daily budget and a bid to meet some business goals. Performing these tasks optimally requires a continuous monitoring of the performance and a frequent tuning of the variables. If we consider that an advertiser



Channel	Ad	Targeting			Economics	
		Place	Time	User's information	Bid	Daily budget
Google	Ad1	Milan	Morning	...	?	?
Google	Ad1	Italy	Afternoon	...	?	?
Google	Ad2	...	...	...	?	?
Google	Ad2	...	...	...	?	?
Bing	Ad3	...	...	...	?	?
Bing	Ad4	...	...	...	?	?
Facebook	Ad5	...	...	...	?	?
Facebook	Ad5	...	...	...	?	?
Facebook	Ad6	...	...	...	?	?
Facebook	Ad6	...	...	...	?	?

Figure 4.1: *Example of campaign structure and set of variables that have to be optimized by the advertiser. Each row corresponds to a sub-campaign while each column is associated with a variable to optimize.*

has to optimize several advertising campaigns at the same time, the number of sub-campaigns that has to be managed every day becomes considerably large. As a consequence, performing these tasks optimally and efficiently is not feasible for human operators and automatic tools usually can solve these problems only partially. Therefore, it is crucial to develop new automatic systems able to solve thess tasks. The goal of this thesis is to provide a set of AI algorithms that addresses the main advertisers' problems and can be adopted in automatic systems for internet advertising campaigns optimization.

Model

An Internet advertising campaign $\mathcal{C} := \{C_1, \dots, C_N\}$ is described by a set of N sub-campaigns C_j, each of which is identified by a tuple of K features $C_j := (z_{j1}, \dots, z_{jK})$, e.g., specifying the gender, age or the interests of the users we target by the sub-campaign C_j. Each feature $z_{ij} \subseteq Z_i$ is a nonempty set of values characterizing the sub-campaign, where Z_i is the set of the feasible values for the i-th feature. For instance, if the i-th feature corresponds to the gender, with values M for male and F for female, we have $Z_i = \{M, F\}$ as the set of feasible values, and, thus, the corresponding feature can be $z_{ij} = \{M\}$ if the sub-campaign C_j targets only male users, $z_{ij} = \{F\}$ if it targets only the female ones, and $z_{ij} = \{M, F\}$ if it targets both. Let Ω be the space in which the advertiser searches for the optimal configuration of sub-campaigns:

$$\Omega = \{\mathcal{C} = \{C_1, \dots, C_N\} \text{ s.t. } \forall z_{ij} \in C_j \ z_{ij} \subseteq Z_i\}$$

The advertiser, at the beginning of the advertising period, has to define the optimal set of sub-campaigns $\mathcal{C}$ over all the possible sets in Ω.

Then, for every day $t \in \{1, \dots, T\}$, for each of the sub-campaigns $C_j \in \mathcal{C}$, the advertiser chooses a bid/daily budget pair $\boldsymbol{a}_{j,t} = (x_{j,t}, y_{j,t})$. The bid $x_{j,t}$ takes values in a finite space $X \subset \mathbb{R}^+$ and is constrained to be in the interval $[\underline{x}_{j,t}, \overline{x}_{j,t}]$, where $\underline{x}_{j,t}$ and $\overline{x}_{j,t} \in \mathbb{R}^+$ are the minimum and maximum bid an advertiser can choose, respectively. Similarly, the daily budget $y_{j,t}$ takes values in a finite and, for simplicity, evenly-spaced set $Y \subset \mathbb{R}^+$ and is constrained to be in $[\underline{y}_{j,t}, \overline{y}_{j,t}]$, where $\underline{y}_{j,t}$ and $\overline{y}_{j,t} \in \mathbb{R}^+$ are the minimum and maximum daily budget an advertiser can set, respectively. [1] By choosing a specific bid/daily budget pair $(x_{j,t}, y_{j,t})$ for a day t and for each sub-campaign C_j, an advertiser gets an expected cost for each sub-campaign $c_{j,t}$ and an overall expected revenue of $\mathcal{J}(\mathcal{C}, x_{1,t}, ..., x_{|C|,t}, y_{1,t}, ...y_{|C|,t},)$.

Optimization Problem

The goal of an advertiser is to define the optimal set of sub-campaigns C and at every day $t \in \{1, \dots, T\}$ choice the optimal set of values of the bid and daily budget for every sub-campaign to maximize the cumulative expected revenue. These values can be found by solving the following optimization problem.

$$\max_{(x_{j,t}, y_{j,t}) \in X \times Y, \mathcal{C}} \mathcal{J}(\mathcal{C}, x_{1,t}, \dots, x_{|\mathcal{C}|,t}, y_{1,t}, \dots, y_{|\mathcal{C}|,t},) \tag{4.1a}$$

$$\text{s.t.} \sum_{j=1}^{|\mathcal{C}|} y_{j,t} \leq \overline{B} \tag{4.1b}$$

$$\frac{\mathcal{J}(\mathcal{C}, x_{1,t}, \dots, x_{|\mathcal{C}|,t}, y_{1,t}, \dots, y_{|\mathcal{C}|,t},)}{\sum_{j=1}^{|\mathcal{C}|} c_{j,t}(\mathcal{C}, x_{1,t}, \dots, x_{|\mathcal{C}|,t}, y_{1,t}, \dots, y_{|\mathcal{C}|,t},)} \leq \lambda \tag{4.1c}$$

$$\underline{x}_{j,t} \leq x_{j,t} \leq \overline{x}_{j,t} \qquad \forall j \tag{4.1d}$$

$$\underline{y}_{j,t} \leq y_{j,t} \leq \overline{y}_{j,t} \qquad \forall j \tag{4.1e}$$

[1] The platforms allow different discretization for the bid and the daily budget. For instance, on the Facebook platform, the daily budget discretization has a step of 1.00 Euro, while, on the Google Adwords platform, a step of 0.01 Euro is allowed.

The objective function stated in Equation (4.1a) is a general formula indicating the revenue, that assumes different meanings depending on the specific context. Usually, this value can be modeled as the sum of the expected number of clicks (or impressions) generated over all the sub-campaigns weighted by the economic value of each click (or conversion). The constraint in Equation (4.1b) is a budget constraint, forcing one not to spend more than the cumulative daily budget limit, [2] while the constraint in Equation 4.1c is a return-on-investment constraint, forcing to keep the ratio between the revenue and the cumulative spend to be larger than a threshold value. Equations (4.1d) and (4.1e) force the variables to assume values in the given ranges for bid and daily budget. In Table 4.1, we report the notation of the variables and parameters of the general problem formulation described above. For the sake of readability, in the next chapters, we will recall this notation highlighting any variations that characterize each specific setting.

Specific Settings and Sub-Problems

Starting from this general formulation we define the following specific problems and, in Chapters 5-6-7-8, we address each of them.

- **Online Joint Bid/Budget Optimization** (Chapter 5). In this problem, we consider the *set of sub-campaigns $\mathcal{C}$ to be fixed a priori* at the beginning of the advertising period and we assume that *all the sub-campaigns are independent each other*. In this framework, we address the optimization problem in an *online fashion* where the goal is to find, at each day t, the bid value $x_{j,t}$ and the budget value $y_{j,t}$ for each sub-campaign C_j.

- **Safe Online Bid Optimization with Return-on-Investment Constraints** (Chapter 6). In this problem, we focus on the scenario in which the daily budget is not a variable that can be set on platforms by the advertiser to limit the daily spend ($y_j = +\infty$). Here, the goal is to find, at each day t, the optimal bid $x_{j,t}$ for each sub-campaign that maximizes the revenue while satisfying, with high probability, some return-on-investment and spend constraints.

- **Offline Targeting Optimization** (Chapter 7). In this problem, we address the targeting , bid and budget optimization in an off-line fashion. The goal is to find a stationary policy that returns the optimal set of

[2] In some platforms where it is not possible to set a daily budget, this constraint can be replaced by a spend constraint of the form $\sum_{j=1}^{|\mathcal{C}|} c_{j,t} \leq \overline{y}_t$.

Table 4.1: *Notation*

$\mathcal{C}$	$\triangleq$	set of advertising campaigns
N	$\triangleq$	number of campaigns
C_j	$\triangleq$	j-th campaign
t	$\triangleq$	current day
T	$\triangleq$	time horizon
$\overline{B}$	$\triangleq$	cumulative daily budget
$x_{j,t}$	$\triangleq$	bid value for the j-th campaign at time t
$y_{j,t}$	$\triangleq$	daily budget value for the j-th campaign at time t
$z_{i,j}$	$\triangleq$	i-th feature value of the j-th sub-campaign
$\boldsymbol{a}_{j,t} = (x_{j,t}, y_{j,t})$	$\triangleq$	bid/daily budget pair for the j-th campaign at time t
$\mathcal{D}$	$\triangleq$	set of the possible bid/daily budget pairs for each campaign
X	$\triangleq$	space of the possible bid values
Y	$\triangleq$	space of the possible daily budget values
M	$\triangleq$	cardinality of $\mathcal{D}$
Z_i	$\triangleq$	set of feasible values for feature z_i
v_j	$\triangleq$	value per click of the j-th campaign
$n_j(.)$	$\triangleq$	expected number of clicks
$c_j(.)$	$\triangleq$	expected spend

sub-campaigns $\mathcal{C}$, the bid value x_j and the optimal budget value y_j to be set for all the advertising period.

- **Offline joint bid/budget optimization with sub-campaigns interdependcies** (Chapter 8). In this problem, we consider the scenario in which the performance of the sub-campaigns are interdependent from each other, i.e., the bid and budget set on a sub-campaign can affect the performance of other sub-campaigns that operates on the same target. We approach this problem in an off-line fashion and we consider $\mathcal{C}$ to be fixed a priori. The goal of this work is to include interdependence relationships among sub-campaigns in the optimization problem and exploits them to return a stationary policy that defines the set of bid $x_{j,t}$ and budget values $y_{j,t}$ that maximize the revenue.

CHAPTER 5

Online Joint Bid/Budget Optimization

In this chapter, we address the joint bid/budget optimization problem where the learner has to find, for each sub-campaign, the optimal bid/budget pair that maximizes the revenue under an overall budget constraint. We formulate this problem as a combinatorial semi bandit problem [14], in which, at every round and for each campaign, an advertiser chooses a pair of bid/daily budget values and observes some information on the performance of the campaign. We discretize the bid/daily budget space, and we formulate the optimization problem as a special case of the Multiple-Choice Knapsack problem [51], that we solve by dynamic programming in a fashion similar to the approximation scheme for the knapsack problem. We resort to Gaussian Process (GP) regression models [48] to estimate the uncertain parameters of the optimization problem (*e.g.*, number of clicks and value per click), as the adoption of GPs requires mild assumptions on the regularity of the functions we need to learn and allows one to capture the correlation among the data thus mitigating the data scarcity issue.

We design four bandit techniques to balance exploration and exploitation in the learning process, that return either samples or upper confidence bounds of the stochastic variables estimated by the GPs to use in the optimization problem. We show that our algorithms enjoy a regret that is upper bounded with high probability as $O(\sqrt{T})$, where T is the learning time horizon.

Finally, we experimentally evaluate the convergence of our algorithms to the optimal (clairvoyant) solution and its empirical regret as the size of the problem varies using a realistic simulator based on the *Yahoo!* Webscope $A3$ dataset. Furthermore, we present the results of the adoption of our algorithms in a real-world application over a period longer than one year, with an average cumulative daily budget of about 1,000 Euros.

This chapter is structured as follows. In Section 5.1, we formulate our problem as a combinatorial semi bandit. In Section 5.2, we describe our algorithms, whereas, in Section 5.3, we provide their theoretical regret analysis. In Section 5.4, we present the experimental evaluation of the algorithms. We report in the Appendix A.1 the proofs of our theoretical results.

Problem Formulation

Optimization Problem Formulation

An advertiser is provided with a collection of $N \in \mathbb{N}$ advertising campaigns $\mathcal{C} = \{C_1, \dots, C_N\}$, where C_j is the j-th campaign, a finite time horizon of $T \in \mathbb{N}$ days, and cumulative daily budget $\overline{B}$ an advertiser is willing to spend at day $t \in \{1, \dots, T\}$ over all the sub-campaigns. [1] For every day $t \in \{1, \dots, T\}$ and every sub-campaign $C_j, j \in \{1, \dots, N\}$, an advertiser chooses a bid/daily budget pair $\boldsymbol{a}_{j,t} = (x_{j,t}, y_{j,t})$. The bid $x_{j,t}$ takes values in a finite space $X \subset \mathbb{R}^+$ and is constrained to be in the interval $[\underline{x}_{j,t}, \overline{x}_{j,t}]$, where $\underline{x}_{j,t}$ and $\overline{x}_{j,t} \in \mathbb{R}^+$ are the minimum and maximum bid an advertiser can choose, respectively. Similarly, the daily budget $y_{j,t}$ takes values in a finite and, for simplicity, evenly-spaced set $Y \subset \mathbb{R}^+$ and is constrained to be in $[\underline{y}_{j,t}, \overline{y}_{j,t}]$, where $\underline{y}_{j,t}$ and $\overline{y}_{j,t} \in \mathbb{R}^+$ are the minimum and maximum daily budget an advertiser can set, respectively. [2] By choosing a specific bid/daily budget pair $(x_{j,t}, y_{j,t})$ for a day t on campaign C_j, an advertiser gets an expected revenue of $v_j \, n_j(x_{j,t}, y_{j,t})$, where v_j is the value per click for campaign C_j and $n_j(x_{j,t}, y_{j,t})$ is the corresponding expected number of clicks. The goal of an advertiser at every day $t \in \{1, \dots, T\}$ is the choice of the values of the bid and daily budget for every campaign to maximize the cumulative expected revenue. These values can be found by solving the

[1] For the sake of presentation, from now on, we set one day as the unitary temporal step of our algorithms. The application of our techniques to different time units is straightforward by opportunely scaling the variables and parameters.

[2] The platforms allow different discretization for the bid and the daily budget. For instance, on the Facebook platform, the daily budget discretization has a step of 1.00 Euro, while, on the Google Adwords platform, a step of 0.01 Euro is allowed.

following optimization problem.

$$\max_{(x_{j,t},y_{j,t})\in X\times Y} \sum_{j=1}^{N} v_j \, n_j(x_{j,t}, y_{j,t}) \tag{5.1a}$$

$$\text{s.t.} \sum_{j=1}^{N} y_{j,t} \leq \overline{B} \tag{5.1b}$$

$$\underline{x}_{j,t} \leq x_{j,t} \leq \overline{x}_{j,t} \qquad \forall j \tag{5.1c}$$

$$\underline{y}_{j,t} \leq y_{j,t} \leq \overline{y}_{j,t} \qquad \forall j \tag{5.1d}$$

The objective function stated in Equation (6.1a) is the weighted sum of the expected number of clicks n_j generated by all the campaigns, where the weights v_j are the campaigns' value per click. The constraint in Equation (5.1b) is a budget constraint, forcing one not to spend more than the cumulative daily budget limit, while the constraints in Equations (5.1c) and (5.1d) force the variables to assume values in the given ranges for bid and daily budget. The above problem is a special case of the Multiple-Choice Knapsack (MCK) [29], which is a variant of the knapsack problem where the items are divided into classes and at most one item per class can be chosen. In the problem above, a class corresponds to a campaign, whereas an item corresponds to a pair of bid/daily budget values.

Each function $n_j(x, y)$ is fully described by $|X| \, |Y|$ parameters. However, exploiting the structure of the problem, a much more concise representation using only $2\,|X|$ parameters can be provided. [3] We factorize the dependency of the number of clicks on x and y as follows:

$$n_j(x, y) := \min\left\{n_j^{\mathsf{sat}}(x), y \, e_j^{\mathsf{sat}}(x)\right\}, \tag{5.2}$$

where the functions $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ describe:

- the maximum number of clicks $n_j^{\mathsf{sat}} : X \to \mathbb{R}^+$ that can be obtained with a given bid x without any daily budget constraint (or, equivalently, letting $y \to +\infty$);
- the number of clicks per unit of daily budget $e_j^{\mathsf{sat}} : X \to \mathbb{R}^+$ with a given bid x, under the assumption that the number of clicks depends linearly on the daily budget when $n_j \leq n_j^{\mathsf{sat}}$.

The rationale is that the maximum number of clicks $n_j^{\mathsf{sat}}(x)$ obtainable with bid x is finite and depends on the number of auctions an advertiser can win

[3] The reduction of the number of the parameters from $|X| \, |Y|$ to $2\,|X|$ does not affect the complexity of the optimization problem, but it plays a crucial role when one needs to learn these parameters.

when using bid x. More specifically, $n_j^{\text{sat}}(x)$ is monotonically increasing in x, since the number of auctions won by an advertiser and the average quality of the slots in which the ad is displayed monotonically increase in x. Notably, the cost per click monotonically increases in x and, therefore, $e_j^{\text{sat}}(x)$ monotonically reduces in x. Finally, fixed the value of x, the number of clicks increases linearly in the daily budget y, where the slope is the number of clicks per unit of daily budget $e_j^{\text{sat}}(x)$, until the maximum number of clicks $n_j^{\text{sat}}(x)$ obtainable with bid x is achieved. For the sake of clarity, we report in Figure 5.1 an example of function n_j for four values of bid; the black dashed curve depicts $\max_{x \in X}\{n_j(x, y)\}$ as the daily budget y varies.

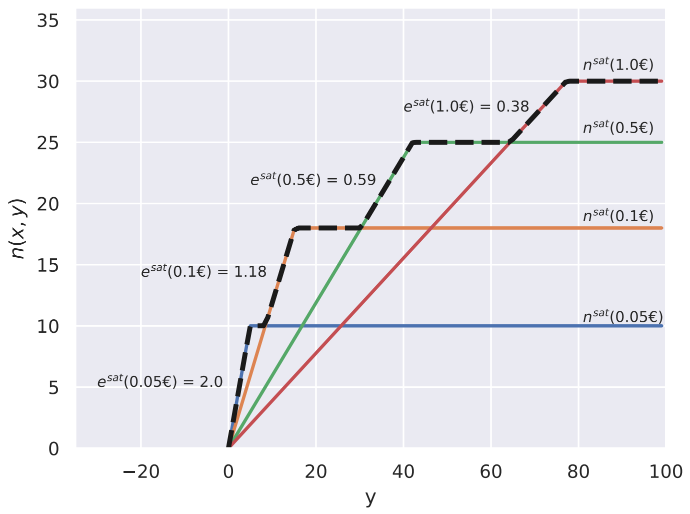


Figure 5.1: *An example of n_j for four values of bid.*

Learning Problem Formulation

In concrete scenarios, the functions $n_j(\cdot, \cdot)$ and the parameters v_j in the optimization problem stated in Equations (6.1a)–(5.1d) are not *a priori* known, but they need to be estimated online. Thus, an algorithm needs to gather as much information as possible about these functions during the operational life of the system, and, at the same time, not to lose too much revenue in exploring suboptimal bid/daily budget allocations (a.k.a. exploration/exploitation dilemma). Thus, our learning problem can be naturally formulated in a sequential decision fashion [13] as a Combinatorial Semi

Bandit problem (CSB) [14]. [4] In the CSB framework, at every round, the learner chooses from a finite set of options, called *arms*, a subset of these, called *superarm*, subject to some combinatorial constraints (*e.g.*, knapsack constraints). Subsequently, the learner observes the payoffs of every single arm belonging to the chosen superarm and gets the corresponding reward. In the optimization problem stated in Equations (6.1a)–(5.1d), each arm corresponds to a bid/daily budget pair, each superarm corresponds to a collection of pairs, one per campaign, and the constraints consist in satisfying the overall daily budget constraint and the range constraints for the bid and daily budget. The payoff of every arm is the revenue we obtain by setting a bid/daily budget pair.

We denote with $\mathcal{D} = X \times Y$, where $|\mathcal{D}| = M$, the finite space of bid/daily budget pairs. The learning process proceeds as follows. Every day t, an advertiser, called *learner* from hereafter, chooses a superarm $\mathcal{S} \in \mathcal{D}^N$, where $S_t := (\boldsymbol{a}_{1,t}, \dots, \boldsymbol{a}_{N,t})$ and the arm $\boldsymbol{a}_{j,t} \in \mathcal{D}$ is the bid/daily budget pair we set for campaign C_j at day t. Such a superarm S_t must be feasible according to the constraints in Equations (5.1b)-(5.1d). The choice of superarm S_t leads to a revenue expressed in terms of clicks and value per click. We denote the random variable corresponding to the number of clicks of campaign C_j by $N_j(x_{j,t}, y_{j,t})$ and the random variable corresponding to the value per click of campaign C_j by V_j. Thus, the revenue is a random variable $\sum_j V_j N_j(x_{j,t}, y_{j,t})$. We denote with $r_{\boldsymbol{\mu}}(S_t)$ the expected value of the revenue when we pull superarm S_t, and the vector $\boldsymbol{\mu}$ of the expected revenues of each arm of every campaign is:

$$\boldsymbol{\mu} := (v_1 n_1(x_1, y_1), \dots, v_1 n_1(x_M, y_M), v_{N-1} n_{N-1}(x_1, y_1), \dots$$
$$v_{N-1} n_{N-1}(x_M, y_M), v_N n_N(x_1, y_1), \dots, v_N n_N(x_M, y_M)).$$

From now on, we refer to the problem defined above as the Advertisement Bid/ daily Budget Allocation (ABBA) problem.

A policy $\mathfrak{U}$ solving our problem is an algorithm returning, each day $t \in \{1, \dots, T\}$, a superarm S_t. Given a policy $\mathfrak{U}$, we define the expected *pseudo-regret* over a time horizon of T as:

$$\mathcal{R}_T(\mathfrak{U}) := T\, r^*_{\boldsymbol{\mu}} - \mathbb{E}\left[\sum_{t=1}^{T} r_{\boldsymbol{\mu}}(S_t)\right],$$

where $r^*_{\boldsymbol{\mu}} := r_{\boldsymbol{\mu}}(S^*)$ is the expected value of the revenue provided by the clairvoyant algorithm choosing the optimal superarm $S^* = (\boldsymbol{a}^*_1, \dots \boldsymbol{a}^*_N\}_{j=1}^N$

[4] Another approach solving this problem is to use a multistage method, *e.g.*, backward induction, but, even for problems with few campaigns and for only 2 stages, such technique would require a huge computational effort that makes these methods an unfeasible solution in practice.

Table 5.1: *Notation*

$\mathcal{C}$	$\triangleq$	set of advertising campaigns
N	$\triangleq$	number of campaigns
C_j	$\triangleq$	j-th campaign
t	$\triangleq$	current day
T	$\triangleq$	time horizon
$\overline{B}$	$\triangleq$	cumulative daily budget
$x_{j,t}$	$\triangleq$	set of bid values for the j-th campaign at time t
$y_{j,t}$	$\triangleq$	set of the daily budget values for the j-th campaign at time t
$\boldsymbol{a}_{j,t} = (x_{j,t}, y_{j,t})$	$\triangleq$	set of the bid/daily budget pairs for the j-th campaign at time t
$\mathcal{D}$	$\triangleq$	set of the possible bid/daily budget pairs for each campaign
X	$\triangleq$	space of the possible bid values
Y	$\triangleq$	space of the possible daily budget values
M	$\triangleq$	cardinality of $\mathcal{D}$
S_t	$\triangleq$	tuple of bid/daily budget pairs (superarms) for the campaigns at time t
v_j	$\triangleq$	value per click of the j-th campaign
$n_j(x_{j,t}, y_{j,t})$	$\triangleq$	expected number of clicks given by a bid/daily budget pair $(x_{j,t}, y_{j,t})$
$\boldsymbol{\mu}$	$\triangleq$	vector of the expected revenues of each arm of every campaign
$\boldsymbol{a}_j^*$	$\triangleq$	optimal bid/budget pair for the j-th campaign
S^*	$\triangleq$	optimal superarm for the set of campaigns $\mathcal{C}$
$r_{\boldsymbol{\mu}}(S_t)$	$\triangleq$	expected value of the revenue when we pull superarm S_t at round t
$r^*_{\boldsymbol{\mu}} = r_{\boldsymbol{\mu}}(S^*)$	$\triangleq$	expected value of the revenue when we pull the optimal superarm S_t at round t

that is the solution to the problem in Equations (6.1a)-(5.1d), 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 $\mathcal{R}_T(\mathfrak{U})$. A recap of the notation defined in this section and used from now on is provided in Table 5.1.

Previous Results on Related Learning Problems

The Combinatorial Bandit framework is introduced in the seminal work [14], in which the authors also propose an algorithm based on statistical upper confidence bounds, namely CUCB. Under the assumption that the support of the payoff functions is bounded on $[0, 1]$, the CUCB algorithm provides a regret $O(\log(T))$. The CUCB does not exploit the potential correlation existing among the expected reward of the arms, which makes its application to our specific scenario unfeasible due to the long time needed for learning the parameters. Another work in the Combinatorial Bandit literature related to our paper is the one presented in [50], in which the authors design an algorithm for a combinatorial semi-bandit problem with knapsack constraints. Differently from our setting, in this scenario, every single arm is assigned a specific budget that recedes every time the arm is pulled. The process stops as soon as one of the arms runs out of its budget. Differently, in our setting, we have a cumulative budget for every day, allowing the pull of the arms for an arbitrary number of times. In the Combinatorial Bandit literature, few works are known to exploit arm correlation to speed up the learning process. The most significant result, provided by Degenne and Perchet [16], describes an algorithm for the specific case of combinatorial constraints in which we are allowed to pull a fixed number of arms at each round.

Other works related to ours can be found in the (non-combinatorial) MAB literature. More precisely, Srinivas *et al.* [52] propose the GP-UCB algorithm that employs GPs in the basic stochastic MAB setting where, at every round, only a single arm can be pulled. The pseudo-regret of the GP-UCB algorithm is proved to be upper bounded with high probability as $\tilde{O}(\sqrt{T})$. These algorithms cannot be directly applied to our scenario where, instead, we can pull a superarm subject to a set of constraints. Notably, we extend the work in [52] to the more challenging combinatorial setting and show that it has the same upper bound on the pseudo-regret $\tilde{O}(\sqrt{T})$.

Proposed Method: the AdComb Algorithm

The Main Algorithm

Algorithm 1, named AdComB, provides the high-level pseudocode of our method, and its scheme is reported in Figure 5.2. The input to the algorithm is composed of: the discrete set of bid values X, the discrete set of daily budget values Y, a model $\mathcal{M}_j^{(0)}$ that, for each campaign C_j, captures the prior knowledge of the learner about the function $n_j(\cdot, \cdot)$ and the parameter v_j, a overall daily budget $\overline{B}$, and a time horizon T. We distinguish three phases that are repeated every day $t \in \{1. \dots, T\}$.

In the first phase (Lines 4–8), denoted with *Estimation* in Figure 5.2, the algorithm learns, from the observations of days $\{1, \dots, t-1\}$, the model $\mathcal{M}_j$ of every campaign C_j. In particular, the model $\mathcal{M}_j$ provides a probability distribution over the average number of clicks $n_j(x, y)$ as the bid x and the daily budget y vary and over the average value per click v_j. The

Algorithm 1 AdComB

```
1: Input: set $X$ of bid values, set $Y$ of daily budget values, prior model $\{\mathcal{M}_j^{(0)}\}_{j=1}^N$,
   overall daily budget $\overline{B}$, time horizon $T$
2: for $t \in \{1, \dots, T\}$ do
3:     for $j \in \{1, \dots N\}$ do
4:         if $t = 1$ then
5:             $\mathcal{M}_j \leftarrow \mathcal{M}_j^{(0)}$
6:         else
7:             Get $(\tilde{n}_{j,t-1}, \tilde{c}_{j,t-1}, \tilde{g}_{j,t-1}, \tilde{v}_{j,t-1})$
8:             $\mathcal{M}_j \leftarrow$ Update $(\mathcal{M}_j, (\hat{x}_{j,t-1}, \hat{y}_{j,t-1}, \tilde{n}_{j,t-1}, \tilde{c}_{j,t-1}, \tilde{g}_{j,t-1}, \tilde{v}_{j,t-1}))$
9:         $(\hat{n}_j(\cdot,\cdot), \hat{v}_j) \leftarrow$ Sampling $(\mathcal{M}_j, X, Y)$
10:    $\{(\hat{x}_{j,t}, \hat{y}_{j,t})\}_{j=1}^N \leftarrow$ Optimize $(\{\hat{n}_j(\cdot,\cdot), \hat{v}_j, X, Y\}_{j=1}^N, \overline{y}_t)$
11:    Pull $(\hat{x}_{1,t}, \hat{y}_{1,t}, \dots, \hat{x}_{N,t}, \hat{y}_{N,t})$
```

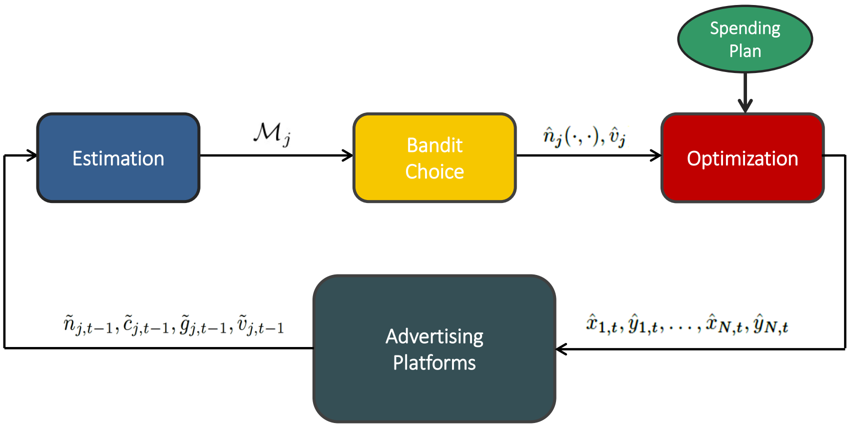

Spending Plan
Estimation
$\mathcal{M}_j$
Bandit Choice
$\hat{n}_j(\cdot,\cdot), \hat{v}_j$
Optimization
$\tilde{n}_{j,t-1}, \tilde{c}_{j,t-1}, \tilde{g}_{j,t-1}, \tilde{v}_{j,t-1}$
Advertising Platforms
$\hat{x}_{1,t}, \hat{y}_{1,t}, \dots, \hat{x}_{N,t}, \hat{y}_{N,t}$

Figure 5.2: *The information flow in the* AdComB *algorithm along with the three phases.*

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, during the subsequent days, for every campaign C_j, the algorithm gets an observation corresponding to day $t-1$ (Line 7) composed of:

- $(\tilde{n}_{j,t-1})$ the actual number of clicks;
- $(\tilde{c}_{j,t-1})$ the actual total daily cost of the campaign;
- $(\tilde{g}_{j,t-1})$ the time when the daily budget $\overline{y}_{t-1}$ exhausted at $t-1$, if so;
- $(\tilde{v}_{j,t-1})$ the actual value per click;

and, subsequently, updates the model of each campaign $\mathcal{M}_j$ using those observations (Line 8).

In the second phase (Line 9), denoted with *Bandit Choice* in Figure 5.2, the algorithm uses the model $\mathcal{M}_j$ just updated to infer an estimate of the function $n_j(\cdot,\cdot)$, for every value of bid and daily budget in X and Y, respectively, and of the parameter v_j. We denote these estimates with $\hat{n}_j(\cdot,\cdot)$ and $\hat{v}_j$, respectively.

In the third phase (Lines 10–11), denoted with *Optimization* in Figure 5.2, the algorithm employs the estimates $\hat{n}_j(\cdot,\cdot)$ and $\hat{v}_j$ in place of $n_j(\cdot,\cdot)$ and v_j in the problem stated in Equations (6.1a)–(5.1d). Finally, it solves the optimization problem returning the bid/daily budget allocation for the next day t (Line 11).

In what follows, we provide a detailed description of the model $\mathcal{M}_j$ and of the subroutines $\mathsf{Update}(\cdot)$, $\mathsf{Sampling}(\cdot)$, and $\mathsf{Optimize}(\cdot)$ used in Algorithm 1.

Model and Update Subroutine

As mentioned before, the Update subroutine generates an estimate of the number of clicks $n_j(\cdot,\cdot)$ and value per click v_j using the previous observations. To avoid data scarcity issues and speed up the learning process, we make mild assumptions on the function $n_j(\cdot,\cdot)$, and we model it by resorting to GPs [48]. These models, developed in the statistical learning field, capture the correlation of the nearby points in the input space exploiting kernel functions. 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 discretized space of bid/daily budget pairs—, thus giving information both on the expected values of the quantities to estimate and their uncertainty.

For the sake of presentation, we describe how we model the maximum number of clicks $n_j^{\mathsf{sat}}(\cdot)$ with a GP regression model. The model directly applies to the number of clicks per unit of daily budget $e_j^{\mathsf{sat}}(\cdot)$. Furthermore, in some situations, the factorization introduced in Equation (5.2) may not be exploited by a learning algorithm, as we discuss below. In these cases, one can adopt a 2-dimensional GP to model $n_j(\cdot,\cdot)$. The treatment of this case, called *unfactorized* hereafter, is analogous to that of $n_j^{\mathsf{sat}}(\cdot)$, but, every time the factorized model can be employed, its use is preferable due to the curse of dimensionality [8]. In the following, we use AdComb-F to refer to the algorithm when the factorized model is used, while we use AdComb-U for the case in which we do not use the factorized model.

We model $n_j^{\mathsf{sat}}(\cdot)$ for campaign C_j with a GP over the bid space X, *i.e.*, using a collection of random variables having a joint Gaussian distribution. Following the definition provided in [48], a GP is completely specified by the mean $m : X \to \mathbb{R}$ and covariance $k : X \times X \to [0, 1]$ functions. Hence, we denote the GP that models the maximum number of clicks in C_j as follows:

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

More specifically, the correlation structure we use is given by a squared exponential kernel:

$$k(x, x') = \exp\left\{-\frac{(x - x')^2}{2\, l^2}\right\} \quad \forall x, x' \in X,$$

where $l \in \mathbb{R}^+$ is a length-scale parameter determining the smoothness of the function. [5] Other common choices for the kernel can be found in [52].

According to GP model, at every day t, the predictive distribution corresponding to the maximum number of clicks $n_{j,t}^{\mathsf{sat}}(x)$ on campaign C_j for the bid x is estimated by $\mathcal{N}(\hat{\mu}_{j,t-1}(x), \hat{\sigma}^2_{j,t-1}(x))$ with:

$$\begin{aligned}\hat{\mu}_{j,t-1}(x) &= m(x) + \mathbf{k}(x, \hat{\boldsymbol{x}}_{j,t-1})^\top \Phi^{-1}\left(\tilde{\boldsymbol{n}}_{j,t-1}^{\mathsf{sat}} - \mathbf{m}_{j,t-1}\right),\\ \hat{\sigma}^2_{j,t-1}(x) &= \ k(x, x) - \mathbf{k}(x, \hat{\boldsymbol{x}}_{j,t-1})^\top\ \Phi^{-1}\, \mathbf{k}(x, \hat{\boldsymbol{x}}_{j,t-1}),\end{aligned}$$

where $\hat{\boldsymbol{x}}_{j,t-1} := (\hat{x}_{j,1}, \ldots, \hat{x}_{j,t-1})^\top$ is the observed bids vector, $\mathbf{k}(x, \hat{\boldsymbol{x}}_{j,t-1}) := (k(x, \hat{x}_{j,1}), \ldots, k(x, \hat{x}_{j,t-1}))^\top$ is the correlation value for the bid x w.r.t. each element of the vector $\hat{\boldsymbol{x}}_{j,t-1}$, $\mathbf{m}_{j,t-1} := (m(\hat{x}_{j,1}), \ldots, m(\hat{x}_{j,t-1}))^\top$ is the vector of the prior for the input in $\hat{\boldsymbol{x}}_{j,t-1}$, $\tilde{\boldsymbol{n}}_{j,t-1}^{\mathsf{sat}} := \left(\tilde{n}_{j,1}^{\mathsf{sat}}, \ldots, \tilde{n}_{j,t-1}^{\mathsf{sat}}\right)^\top$ is the vector of maximum number of clicks achieved the previous days, $[\Phi]_{h,k} := k(\hat{x}_{j,h}, \hat{x}_{j,k}) + \lambda$ is the Gram matrix built on the available data, and λ is the variance of the realizations we use in the estimation process. [6][7] Note that the distribution of the maximum number of clicks at the first day is $\mathcal{N}(m(x), k(x, x))$ for each $x \in X$ since no information from the data can be used yet.

[5] If available, *a priori* information on the process can be employed to design a function $m(x)$ over the input space X which specifies the mean value. For instance, when information on the maximum number θ of clicks achievable for any bid is available, one may use a linearly increasing function over the bid space as $m(x) = \frac{x\theta}{\max_{x' \in X} x'}$. If instead no *a priori* information is available, one can use a uninformative prior mean by setting $m(x) = 0, \forall x \in X$.

[6] From now on, we denote with $\mathcal{N}(\mu, \sigma^2)$ the Gaussian with mean μ and variance σ^2.

[7] The computation cost of the estimation can be dramatically reduced by using an alternative, but much more involved, approach whereby the inverse of the Gram matrix Φ^{-1} is stored and updated iteratively at each day; see [48] for details.

The parameter $\tilde{n}^{\mathsf{sat}}_{j,t}$ is set equal to the observation $\tilde{c}_{j,t}$ when the daily budget $\hat{y}_{j,t}$ used for campaign C_j did not exhaust. When instead $\hat{y}_{j,t}$ exhausted, we have not a direct observation of $\tilde{n}^{\mathsf{sat}}_{j,t}$, and, thus, we set $\tilde{n}^{\mathsf{sat}}_{j,t}$ as a function of the time $\tilde{g}_{j,t}$. For instance, if we assume a uniform distribution of the clicks over the day, the value of $\tilde{n}^{\mathsf{sat}}_{j,t}$ has the following expression:

$$\tilde{n}^{\mathsf{sat}}_{j,t} := \frac{24}{\tilde{g}_{j,t}} \tilde{n}_{j,t},$$

where $\tilde{g}_{j,t} \in (0, 24]$ is expressed in hours. In general, this relationship can be estimated from historical data coming from past advertising campaigns of products belonging to the same category (*e.g.*, toys, insurances, beauty products). Conversely, if no information on how the clicks distribute over the day is available, one has to rely on the unfactorized model for $n_j(\cdot, \cdot)$. Similar considerations hold for the estimation of the value per click v_j. We estimate v_j, at day t, from the observations $\tilde{\boldsymbol{v}}_{j,t-1} := (\tilde{v}_{j,1}, \ldots, \tilde{v}_{j,t-1})^\top$ of the previous days up to $t-1$. We use a single Gaussian probability distribution to model the value per click v_j, thus, at every day t, given the observations $\tilde{\boldsymbol{v}}_{j,t-1}$, we estimate its mean $\hat{\nu}_{j,t}$ and variance $\hat{\psi}^2_{j,t}$ relying on the Bayesian update of a prior $\mathcal{N}(0, \psi^2_j)$ [22], as follows:

$$\hat{\nu}_{j,t-1} := \frac{\psi_j^2 \sum_{h=1}^{t-1} \tilde{v}_{j,h}}{\xi + (t-1)\psi_j^2},$$
$$\hat{\psi}^2_{j,t-1} := \frac{\psi_j^2 \xi}{\xi + (t-1)\psi_j^2},$$

where ξ is the measurement noise variance. To summarize, the data needed for updating the model $\mathcal{M}_j$ corresponding to campaign C_j at day t consists of the following elements:

- the values per click $\tilde{\boldsymbol{v}}_{j,t-1}$,
- the chosen bids $\hat{\boldsymbol{x}}_{j,t-1}$,
- the maximum number of achievable clicks $\tilde{\boldsymbol{n}}^{\mathsf{sat}}_{j,t-1}$,
- the number of clicks per unit of daily budget $\tilde{\boldsymbol{e}}^{\mathsf{sat}}_{j,t-1} := \left(\frac{\tilde{n}_{j,1}}{\tilde{c}_{j,1}}, \ldots, \frac{\tilde{n}_{j,t-1}}{\tilde{c}_{j,t-1}}\right)^\top$

Sampling Subroutine

The Sampling subroutine aims at returning an estimate of the expected number of clicks and the value per click to use in the optimization problem stated in Equations (6.1a)–(5.1d). The näive choice of using the expected value computed from $\mathcal{M}_j$ may not provide any guarantee to minimize the regret $\mathcal{R}_T(\mathfrak{U})$, as it is well known in the bandit literature. To guarantee that our algorithm minimizes the cumulative expected regret, we compute an estimation exploiting the information on the uncertainty provided by model $\mathcal{M}_j$. More precisely, the model $\mathcal{M}_j$ associated with campaign C_j provides a probability distribution over the values of the function $n_j(\cdot, \cdot)$ and the values v_j can assume. This is equivalent to say that $\mathcal{M}_j$ provides a probability distribution over the possible instances of the optimization problem in Equations (6.1a)–(5.1d). 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 $x \in X, y \in Y$ and v_j.

We propose two different approaches for the sampling phase, namely AdComB-UCB and AdComB-TS, taking inspiration from the GPUCB algorithm [52], and the Thompson Sampling (TS) algorithm [54], respectively. [8]

The AdComB-UCB algorithm uses upper confidence bounds on the expected value of the posterior distributions to estimate $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$. More specifically, $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ are replaced in the optimization problem defined in Equation (6.1a)–(5.1d) by:

$$u_{j,t-1}^{(n)}(x) := \hat{\mu}_{j,t-1}(x) + \sqrt{b_{j,t-1}^{(n)}}\, \hat{\sigma}_{j,t-1}(x),$$
$$u_{j,t-1}^{(e)}(x) := \hat{\eta}_{j,t-1}(x) + \sqrt{b_{j,t-1}^{(e)}}\, \hat{s}_{j,t-1}(x),$$

respectively, where $\hat{\eta}_{j,t-1}(x)$ and $\hat{s}^2_{j,t-1}(x)$ are the mean and the variance provided by the GP modeling $e_j^{\mathsf{sat}}(\cdot)$, respectively, $b_{j,t-1}^{(n)} \in \mathbb{R}^+$ and $b_{j,t-1}^{(e)} \in \mathbb{R}^+$ are non-negative sequences of values, which will be discussed later on in Section 5.3. Properly setting the values of $b_{i,t-1}^{(n)}$ and $b_{i,t-1}^{(e)}$ leads us to design optimistic bounds, that are necessary for the convergence of the algorithm to the optimal solution. Similarly, for the value per click v_j, we use:

$$u_{j,t-1}^{(v)} := \hat{\nu}_{j,t-1} + \sqrt{b_{j,t-1}^{(v)}}\, \hat{\psi}_{j,t-1},$$

where $b_{j,t-1}^{(v)} \in \mathbb{R}^+$ is a non-negative sequence of values.

Conversely, the AdComB-TS algorithm draws samples from the distributions corresponding to $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ and, consequently, computes the value of $n_j(x, y)$ to be used in the following optimization phase. More

[8] For the sake of clarity, in what follows we describe the our sampling procedure for the AdComb-F version of AdComb; the case for the unfactored model AdComb-U is analogous.

formally, at a given day t and for every bid in $x \in X$, we replace $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ in the optimization problem defined in Equation (6.1a)–(5.1d) with:

$$\theta_{j,t-1}^{(n)}(x) \sim \mathcal{N}(\hat{\mu}_{j,t-1}(x), \hat{\sigma}_{j,t-1}^2(x)),$$
$$\theta_{j,t-1}^{(e)}(x) \sim \mathcal{N}(\hat{\eta}_{j,t-1}(x), \hat{s}_{j,t-1}^2(x)),$$

respectively. Similarly, for the value per click, we draw a sample $\theta_{j,t-1}^{(v)}(x)$ as follows:

$$\theta_{j,t-1}^{(v)} \sim \mathcal{N}(\hat{\nu}_{j,t-1}, \hat{\psi}_{j,t-1}^2).$$

Finally, given the values for $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ generated by one of the two aforementioned methods, we compute $n_j(x, y)$ as prescribed by Equation (5.2) for each bid $x \in X$ and for each daily budget $y \in Y$, and use them in the following optimization procedure.

Optimize Subroutine

Finally, for every campaign C_j, we need to choose the best bid/daily budget pair to set at day t. We resort to a modified version of the algorithm in [29] used for the solution of the knapsack problem. Let us define the set of the feasible bid and daily budgets for the round t and the campaign C_j as $X_{j,t} := X \cap [\underline{x}_{j,t}, \overline{x}_{j,t}]$ and $Y_{j,t} := Y \cap [\underline{y}_{j,t}, \overline{y}_{j,t}]$, respectively. At first, for every value of daily budget $y \in Y_{j,t}$, we define $x_j^*(y) \in X_{j,t}$ as the bid maximizing the number of clicks, formally:

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

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

$$w_j(y) := \begin{cases} v_j \, n_j(x_j^*(y), y) & \underline{y}_{j,t} \le y \le \overline{y}_{j,t} \\ 0 & y < \underline{y}_{j,t} \vee y > \overline{y}_{j,t} \end{cases}.$$

This allows one to remove the dependency of the optimization problem defined in Equations (6.1a)–(5.1d) from x, letting variables y the only variables to deal with.

Finally, the optimization problem is solved in a 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 campaign C_j were the only campaign 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 campaign C_j and, finally, taking the maximum among all these combinations. At the end of the iterative process, 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 daily budget, it is sufficient to store the partial best assignments of budget in the cells of the matrix.

The complexity of the aforementioned algorithm is $O(NH^2)$, *i.e.*, it is linear in the number of campaigns N and quadratic in the number of different values of the budget $H := |Y|$, where $|\cdot|$ is the cardinality of a set. When H is huge, the above algorithm may 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.

Regret Analysis

We provide a theoretical finite-time analysis of the regret $\mathcal{R}_T(\mathfrak{U})$ of the algorithms proposed in the previous section. The derivation of the guarantees of our algorithm exploits the results presented in [2].

Initially, we define the *Maximum Information Gain*, which we use to bound the regret of the AdComb algorithm. Let us start defining the Information Gain of a set of samples drawn from a GP according to [52] as follows: [Information Gain] Given a realization of a GP $f(\cdot)$ and a vector of noisy observations $\mathbf{y}(\mathbf{x}) = (y(x_1), \dots, y(x_t))^\top$ over the input points $\mathbf{x} = (x_1, \dots, x_t)^\top$ for the function $f(\cdot)$, the Information Gain of the set of samples $(\mathbf{x}, \mathbf{y}(\mathbf{x}))$ is defined as:

$$IG(\mathbf{y}(\mathbf{x}) \,|\, f) := \frac{1}{2} \log \left| I + \frac{\Phi}{\lambda} \right|,$$

where I is the identity matrix of order t, λ is the noise variance of the realizations and $[\Phi]_{ij} := k(x_i, x_j)$ is the Gram matrix of the vector computed on the inputs $\mathbf{x}$. Using the previous definition, we define the Maximum Information Gain, as follows. [Maximum Information Gain] Given a realization of a GP $f(\cdot)$, the Maximum Information Gain of a generic set of t noisy observations $\mathbf{y}(\mathbf{x})$ from the function $f(\cdot)$ is defined as:

$$\gamma_t(f) := \max_{\mathbf{x} \in X^t} IG(\mathbf{y}(\mathbf{x}) \mid f),$$

where X is the input space.

For the sake of presentation, we report our regret analysis separately for the case in which the model of $n_j(\cdot, \cdot)$ is unfactorized (Section 5.3.1) and the case in which it is factorized (Section 5.3.2).

Unfactorized Model

We show that the worst-case pseudo-regret of the AdComb algorithm when using the unfactorized model is upper bounded as follows.

Theorem 1. *Let us consider an ABBA problem over T rounds where the functions $n_j(x, y)$ is the realization of a GP. Using the AdComb-U-UCB algorithm with the following upper bounds for the number of clicks and of value per click:*

$$\hat{u}^{(n)}_{j,t-1}(x, y) := \hat{\mu}_{j,t-1}(x, y) + \sqrt{b_t}\, \hat{\sigma}_{j,t-1}(x, y),$$
$$\hat{u}^{(v)}_{j,t-1} := \hat{\nu}_{j,t-1} + \sqrt{b'_t}\, \hat{\psi}^2_{j,t-1},$$

respectively, with $b_t := 2 \log \left(\frac{\pi^2 N M t^2}{3\delta} \right)$ and $b'_t := 2 \log \left(\frac{\pi^2 N t^2}{3\delta} \right)$. For every $\delta \in (0, 1)$, the following holds with probability at least $1 - \delta$:

$$\mathcal{R}_T(\mathfrak{U}) \leq \left\{ 8 T N b_T \left[\frac{v^2_{\max}}{\log\left(1 + \frac{1}{\lambda}\right)} \sum_{j=1}^{N} \gamma_T(n_j) \right.\right.$$
$$\left.\left. + \xi (n_{\max} + 2\sqrt{b'_t}\sigma)^2 \sum_{j=1}^{N} \log \left(\frac{\xi}{\psi^2_j} + T \right) \right] \right\}^{\frac{1}{2}},$$

where, λ and ξ are variances of the measurement noise of the click functions $n_j(\cdot)$ and of the value per click v_j, respectively, $v_{\max} := \max_{j \in \{1,\dots,N\}} v_j$ is the maximum expected value per click, $n_{\max} := \max_{x \in X, y \in Y, j \in \{1,\dots,N\}} n_j(x, y)$ is the maximum expected number of click we might obtain on average over *all the campaigns C_j, and $\sigma^2 := k(\boldsymbol{a}, \boldsymbol{a}) \geq \hat{\sigma}^2_{j,t}(\boldsymbol{a})$ for each j, t and $\boldsymbol{a}$. Equivalently, with probability at least $1 - \delta$, it holds:*

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN \sum_{j=1}^{N} \gamma_T(n_j)}\right),$$

where the notation $\tilde{O}(\cdot)$ disregards the logarithmic factors.

Theorem 2. *Let us consider an ABBA problem over T rounds where the functions $n_j(x, y)$ is the realization of a GP. Using the AdComb-U-TS algorithm, for every $\delta \in (0, 1)$, the following holds with probability at least $1 - \delta$:*

$$\mathcal{R}_T(\mathfrak{U}) \leq \left\{ 8TN \left[\frac{v^2_{\max}}{\log\left(1 + \frac{1}{\lambda}\right)} b_T \sum_{j=1}^{N} \gamma_T(n_j) \right.\right. \left.\left. + \xi b'_T (n_{\max} + \sqrt{b_T}\sigma)^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\psi_j^2} + T\right) \right] \right\}^{1/2},$$

where $b_t := 8 \log\left(\frac{2NMt^2}{3\delta}\right)$, $b'_t := 8 \log\left(\frac{2Nt^2}{3\delta}\right)$, λ and ξ are variances of the measurement noise of the click functions $n_j(\cdot)$ and of the value per click v_j, respectively, $v_{\max} := \max_{j \in \{1,\dots,N\}} v_j$ is the maximum expected value per click, $n_{\max} := \max_{x \in X, y \in Y, j \in \{1,\dots,N\}} n_j(x, y)$ is the maximum expected number of click we might obtain on average over all the campaigns C_j, and $\sigma^2 := k(\boldsymbol{a}, \boldsymbol{a}) \geq \hat{\sigma}^2_{j,t}(\boldsymbol{a})$ for each j, t and $\boldsymbol{a}$.

Equivalently, with probability at least $1 - \delta$, it holds:

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN \sum_{j=1}^{N} \gamma_T(n_j)}\right).$$

Factorized Model

We show that the worst-case pseudo-regret of the AdComb algorithm when using the factorized model is upper bounded as follows.

Theorem 3. *Let us consider an ABBA problem over T rounds where the functions $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ are the realization of GPs. Using the AdComb-F-UCB algorithm with the following upper bounds for the number of clicks,* *the number of clicks per unit of budget, and the value per click, respectively:*

$$\begin{aligned}
u^{(n)}_{j,t-1}(x) &:= \hat{\mu}_{j,t-1}(x) + \sqrt{b_t}\hat{\sigma}_{j,t-1}(x),\\
u^{(e)}_{j,t-1}(x) &:= \hat{\eta}_{j,t-1}(x) + \sqrt{b_t}\hat{s}_{j,t-1}(x),\\
u^{(v)}_{j,t-1} &:= \nu_{j,\hat{t}-1} + \sqrt{b'_t}\hat{\psi}_{j,t-1},
\end{aligned}$$

with $b_t = 2\log\left(\frac{\pi^2 NMt^2}{2\delta}\right)$ *and* $b'_t := 2\log\left(\frac{\pi^2 Nt^2}{2\delta}\right)$. *For every* $\delta \in (0,1)$, *the following holds with probability at least* $1-\delta$*:*

$$\mathcal{R}_T(\mathfrak{U}) \leq \Bigg\{TN\Bigg[\bar{c}_1 b_T \sum_{j=1}^N \gamma_T(n_j) + \bar{c}_2 b_T \sum_{j=1}^N \gamma_T(e_j)$$
$$+\bar{c}_3 b'_T \left(2sy_{\max}\sqrt{b_T} + 2\sigma\sqrt{b_T} + n^{\mathsf{sat}}_{\max}\right)^2 \sum_{j=1}^N \log\left(\frac{\xi}{\psi_j^2} + T\right)\Bigg]\Bigg\}^{1/2},$$

where $\bar{c}_1 := \frac{12v^2_{\max}}{\log\left(1+\frac{1}{\lambda}\right)}$, $\bar{c}_2 := \frac{12v^2_{\max}y^2_{\max}}{\log\left(1+\frac{1}{\lambda'}\right)}$, *and* $\bar{c}_3 := 12\xi$, ξ, λ *and* λ' *are the variance of the value per click, measurement noise on the maximum number of clicks and number of clicks per unit of daily budget, respectively,* $v_{\max} := \max_{j\in\{1,\dots,N\}} v_j$ *is the maximum expected value per click,* $n_{\max} := \max_{x\in X, y\in Y, j\in\{1,\dots,N\}} n_j(x,y)$ *is the maximum expected number of click we might obtain on average over all the campaigns* C_j, $y_{\max} := \max_{y\in Y} y$ *is the maximum budget one can allocate on a campaign, and* $\sigma^2 := k(x,x) \geq \hat{\sigma}^2_{j,t}(x)$, $s^2 := k'(x,x) \geq \hat{s}^2_{j,t}(x)$ *for each* j, t *and* x.

Equivalently, with probability at least $1-\delta$, *it holds:*

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN\sum_{j=1}^N [\gamma_T(n_j) + \gamma_T(e_j)]}\right).$$

Theorem 4. *Let us consider an ABBA problem over* T *rounds where the functions* $n^{\mathsf{sat}}_j(x)$ *and* $e^{\mathsf{sat}}_j(x)$ *are the realization of GPs. Using the* AdComb-F-TS *algorithm, for every* $\delta \in (0,1)$, *the following holds with probability at least* $1-\delta$*:*

$$\mathcal{R}_T(\mathfrak{U}) \leq \Bigg\{TN\Bigg[\bar{c}_1 b_T \sum_{j=1}^N \gamma_T(n_j) + \bar{c}_2 b_T \sum_{j=1}^N \gamma_T(e_j)$$
$$+\bar{c}_3 b'_T \left(2sy_{\max}\sqrt{b_T} + 2\sigma\sqrt{b_T} + n^{\mathsf{sat}}_{\max}\right)^2 \sum_{j=1}^N \log\left(\frac{\xi}{\psi_j^2} + T\right)\Bigg]\Bigg\}^{1/2},$$

where $b_t = 2\log\left(\frac{\pi^2 N M t^2}{2\delta}\right)$, $b'_t := 2\log\left(\frac{\pi^2 N t^2}{2\delta}\right)$, $\bar{c}_1 := \frac{48 v_{\max}^2}{\log\left(1+\frac{1}{\lambda}\right)}$, $\bar{c}_2 := \frac{48 v_{\max}^2 y_{\max}^2}{\log\left(1+\frac{1}{\lambda'}\right)}$, *and* $\bar{c}_3 := 12\xi$, ξ, λ *and* λ' *are the variance of the value per click, measurement noise on the maximum number of clicks and number of clicks per unit of daily budget, respectively,* $v_{\max} := \max_{j\in\{1,\dots,N\}} v_j$ *is the maximum expected value per click,* $n_{\max} := \max_{x\in X, y\in Y, j\in\{1,\dots,N\}} n_j(x,y)$ *is the maximum expected number of click we might obtain on average over all the campaigns* C_j, $y_{\max} := \max_{y\in Y} y$ *is the maximum budget one can allocate on a campaign, and* $\sigma^2 := k(x,x) \geq \hat{\sigma}^2_{j,t}(x)$, $s^2 := k'(x,x) \geq \hat{s}^2_{j,t}(x)$ *for each* j, t *and* x.

Equivalently, with probability at least $1-\delta$, *it holds:*

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN\sum_{j=1}^{N}[\gamma_T(n_j)+\gamma_T(e_j)]}\right).$$

The upper bounds provided by Theorems 1–4 are expressed in terms of the maximum information gain $\gamma_T(\cdot)$ one might obtain selecting T samples from the GPs defined in Section 5.2.2. The problem of bounding $\gamma_T(f)$ for a generic GP f has been already addressed by [52], where the authors present the bounds for the squared exponential kernel $\gamma_T(f) = O((\log T)^{d+1})$, where d is the dimension of the input space of the GP ($d=2$ for AdComb-U, and $d=1$ for AdComb-F). Notice that, thanks to the previous result our AdComb algorithm suffers from a sublinear pseudo-regret since the terms $\gamma_T(n_j)$ and $\gamma_T(e_j)$ are bounded by $O((\log T)^{d+1})$, and the bound in Theorems 1–4 is then $O(N\sqrt{T(\log T)^{d+1}})$.

Experimental Evaluation

This section is structured as follows. In Section 5.4.1, we experimentally evaluate the convergence to the optimal solution and the empirical regrets of our algorithms in synthetic settings. In Section 5.4.2, we present the results of the adoption of our algorithms in a real-world setting.

Evaluation in Synthetic Settings

We evaluate our algorithms in synthetic settings generated as follows. In every setting, there is a single advertiser optimizing a set of N campaigns by our algorithms, and, for every campaign C_j, there are other $\delta_j - 1$ advertisers whose behavior is, instead, stochastic. At day t, every campaign C_j can be involved in a set $AU_{j,t}$ of auctions, whose number $|AU_{j,t}|$ is drawn

Table 5.2: *Parameters of the synthetic settings.*

	C_1	C_2	C_3	C_4
μ_j	1000	1500	1500	1250
σ_j	50	50	50	50
γ_j	5	5	5	5
δ_j	7	7	7	7
$\mu^{(b)}$	0.5	0.33	0.4	0.39
$\sigma^{(b)}$	0.1	0.07	0.1	0.51
$p^{(obs)}(1)$	0.9	0.9	0.9	0.9
$p^{(obs)}(2)$	0.7	0.8	0.7	0.8
$p^{(obs)}(3)$	0.6	0.7	0.6	0.6
$p^{(obs)}(4)$	0.4	0.6	0.4	0.5
$p^{(obs)}(5)$	0.2	0.5	0.3	0.3
$p_j^{(cl)}$	0.5	0.3	0.4	0.4
$p_j^{(co)}$	0.05	0.05	0.04	0.05

from a Gaussian probability distribution $\mathcal{N}(\mu_j^{(s)}, (\sigma_j^{(s)})^2)$ with mean $\mu_j^{(s)}$ and standard deviation $\sigma_j^{(s)}$ and subsequently rounded to the nearest integer. We denote, for campaign C_j, the click and conversion probabilities of the advertiser using our algorithms with $p_j^{(cl)}$ and $p_j^{(co)}$, respectively. These probabilities are the same for all the auctions in which C_j is involved. We assign a tuple of parameters $\mu^{(b)}$, $(\sigma^{(b)})^2$ to every other advertiser before the beginning of the experiment, and, at every auction, the bids b_h are drawn from a Gaussian distribution $\mathcal{N}(\mu^{(b)}, (\sigma^{(b)})^2)$, being $\mu^{(b)}$ and $\sigma^{(b)}$ the mean and standard deviation parameters of the bid distribution, respectively. Similarly, the click probabilities ρ_h are uniformly sampled in the interval $[0, 1]$ at every auction.

The auction mechanism we use is the Vickrey-Clarke-Groves [38] and the number of available slots is γ_j, with $\gamma_j \leq \delta_j$. Once the optimal allocation is found, we simulate a user who may or may not click the ad, and generate a click and/or a conversion according to probabilities $p_j^{(cl)}$ and $p_j^{(co)}$, respectively. After the click, the daily budget of the advertiser using our algorithms is reduced as prescribed by the Vickrey-Clarke-Groves mechanism.

Experiment $\#1$

This experiment aims at showing that the algorithms, which do not sufficiently explore the space of the arms, may not converge to the (clairvoyant) optimal solution. We use a setting with $N = 4$ different campaigns. The parameters describing the setting are provided in Table 5.2. Furthermore, in Figure 5.3, we report, for each campaign C_j, the best instantaneous revenue $v_j \max_x n_j(x, y)$ as the daily budget allocated to the single campaign varies (this is done maximizing the performance over the feasible bid values $x \in X$). The peculiarity of this setting is the similarity of the performance of campaigns C_1 and C_4. Indeed, this similarity makes the identification of the optimal solution hard. We set the following limits for every $t \leq T$ where $T = 200$ days and for every campaign C_j: cumulative budget $\overline{y}_t = 500$, minimum and maximum bid values $\underline{x}_{j,t} = 0$, and $\overline{x}_{j,t} = 2$, respectively, minimum and maximum daily budget values $\underline{y}_{j,t} = 0$, $\overline{y}_{j,t} = 500$, respectively. Furthermore, we use an evenly spaced discretization of $|X| = 10$ bids and $|Y| = 10$ budgets over the aforementioned intervals. We assume a uniform distribution of the clicks and conversions over the day (see Section 5.2).

We compare the experimental results of our algorithms (AdComb-U-UCB, AdComb-U-TS AdComb-F-UCB, and AdComb-F-TS) to identify the modeling and/or exploration strategies providing the best performance. [9] Furthermore, we introduce a baseline represented by the algorithm AdComb-F-MEAN, which is a "pure exploration" version of AdComb-F such that, at every round t, the posterior expected value of the number of clicks for each bid/daily budget pair is given in input to the optimization procedure. In the GPs used by all the algorithms, we adopt a squared exponential kernel, whose hyper-parameters are chosen as prescribed by the GP literature, see [48] for details, and we started from an uninformative zero-mean prior.

In this experiment, in addition to the cumulative pseudo-regret $R_t(\mathfrak{U})$, we also evaluate the expected value of the revenue $r_{\boldsymbol{\mu}}(S_t)$. Obviously, in the case of the cumulative pseudo-regret, the performance improves as the cumulative pseudo-regret reduces, and, conversely, in the case of the revenue, the performance improves as it increases. The experimental results are averaged over 100 independent executions of the algorithms.

In Figure 5.4a, we report the average instantaneous reward $r_{\boldsymbol{\mu}}(S_t)$ of our algorithms, while, in Figure 5.4b, we report their average cumulative pseudo-regret $R_t(\mathfrak{U})$. The reward provided by all the algorithms but AdComb-F-MEAN converges to the optimal reward provided by a clairvoyant algorithm and presents a slightly varying reward even at the end of the time horizon due the variance of the GP used to choose the daily budget allocation over time. This variance is larger at the beginning of the process, thus incentivising exploration, and it decreases as the number of observations in-

[9]Notice that a straightforward extension of the algorithm proposed in [14], *i.e.*, designing a version accounting for Gaussian distribution, would require 100 days to have a single sample for each different bid/daily budget pair. Indeed, it would purely explore the space of arms without any form of exploitation for $t \leq 100$. Therefore, we omit the comparison that would not provide any meaningful insight to the problem.

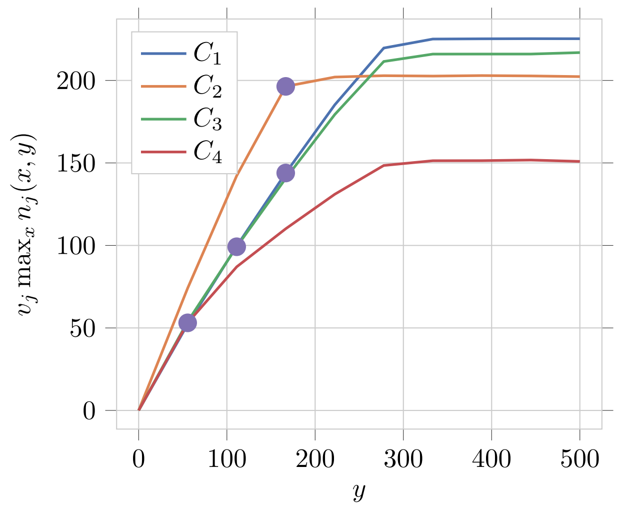
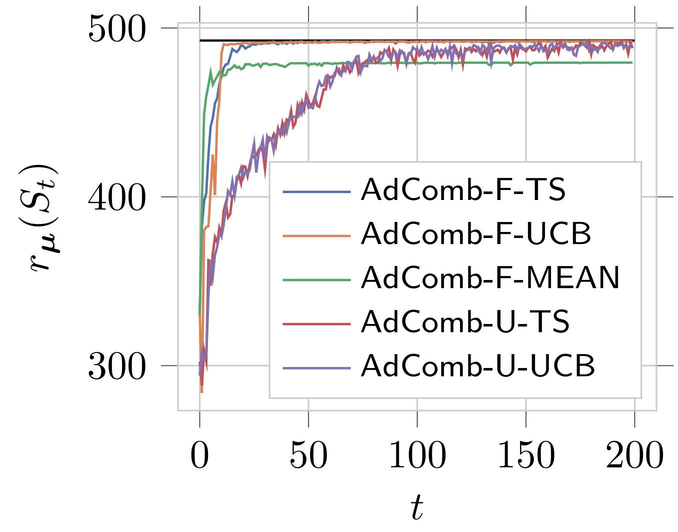


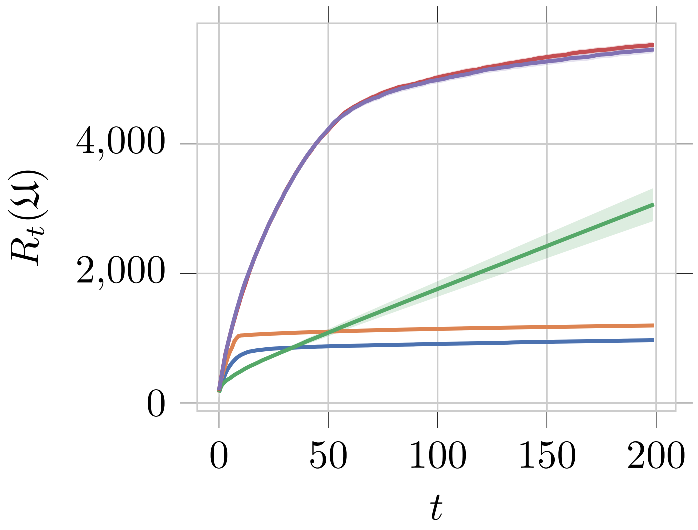
Figure 5.3: *Expected value of* $v_j \max_x n_j(x,y)$ *used in Experiment* $\#1$ *for each campaign and for each value of the daily budget. Violet dots corresponds to the expected value of the number of conversions associated to the optimal daily budget allocation.*

creases, allowing the algorithms to reach the (clairvoyant) optimal reward asymptotically. Although all our algorithms converge to the (clairvoyant) optimal solution, the AdComb-F-TS algorithm provides the smallest cumulative pseudo-regret for every $t \geq 30$. AdComb-F-UCB has performance slightly worse than that of AdComb-F-TS. The AdComb-F-MEAN algorithm provides the best performance for $t \leq 30$, but it is not capable to achieve the (clairvoyant) optimal solution. As a result, for larger values of t, the performance of AdComb-F-MEAN decreases achieving, at $t = 200$, a regret significantly larger than that one provided by AdComb-F-TS. This is because AdComb-F-MEAN does not explore the arms space properly and, as a consequence, in some of the 100 independent runs, it gets stuck in a suboptimal solution of the optimization problem. Conversely, AdComb-F-TS and AdComb-F-UCB, thanks to their exploration incentives, converge to the optimal solution asymptotically in all the runs. We have a similar behavior of the algorithms is a situation in which the performance of the algorithms are rather different and the observations are very noisy. Finally, we observe that AdComb-U-UCB and AdComb-U-TS suffer from a much larger regret than that one of their factorized counterparts—more than 100% at $t = 200$—and this is mainly accumulated over the first half of the time horizon.

In real-world settings, it may be usual dealing with scenarios in which multiple campaigns have similar performance, or they have different perfor-



(a)



(b)

Figure 5.4: *Results for Experiment #1: revenue (a), and cumulative pseudo-regret (b). The black horizontal line in (a) is the optimal reward of the clairvoyant algorithm r^*_{μ}. The shaded regions in (b) represent the 95% confidence intervals of the mean.*

mance and the observations are very noisy. In those situations, AdComb-F-MEAN might get stuck in a suboptimal solution, thus providing a small expected revenue w.r.t. a clairvoyant algorithm. Conversely, both the unfactorized and the factorized versions of our algorithms might still be a viable solution since they have proven to converge to the optimum asymptotically. For this reason, we do not recommend the adoption of the AdComb-F-MEAN algorithm in practice and we omit its evaluation in the following experimental activities.

Experiment $\#2$

This experiment aims at evaluating how the size of the discretization of the bid space (in terms of $|X|$) and daily budget space (in terms of $|Y|$) used in our algorithms affect their performance. Indeed, if, on the one hand, an increase in the number of the available bid/daily budget pairs corresponds to an increase of the expected revenue of the clairvoyant solution, on the other hand, a larger arms space results in larger exploration costs. We investigate how the impact on the exploration cost is mitigated by the correlation between arms that allows one to gain information over all the arms space once one arm is pulled.

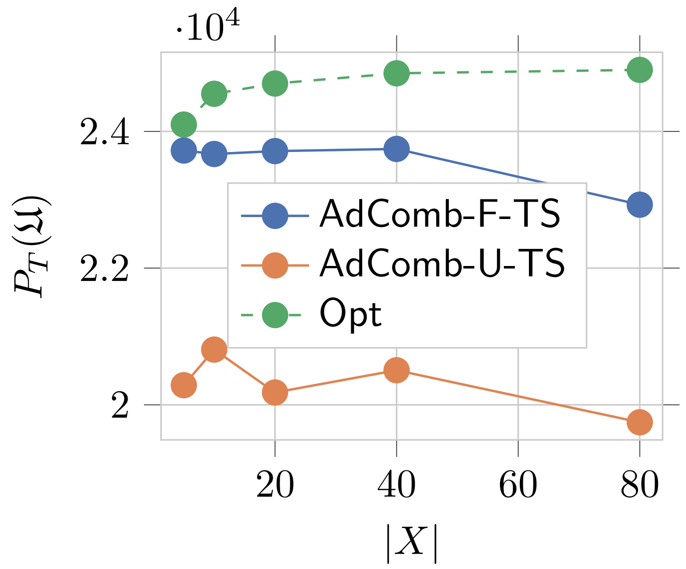
In this experiment, we adopt the same setting used in Experiment $\#1$ (see Table 5.2) with different granularities of discretization in the bid space X or daily budget space Y. In particular, we study two settings over a time horizon of $T = 50$ rounds. In the first setting, the number of values of the bid space is $|X| \in \{5, 10, 20, 40, 80\}$, while the number of daily budget values is $|Y| = 10$. In the second setting, the number of values of the daily budget space is $|Y| \in \{5, 10, 20, 40, 80\}$, while the number of bid values is $|X| = 10$. These discretizations are such that every space with a larger number of values includes the space with a smaller number of values. For instance, a set with 40 values strictly includes the one with 20 values.

We compare AdComb-U-TS and AdComb-F-TS in terms of both cumulative expected revenue (over time) $P_T(\mathfrak{U}) := \sum_{t=1}^{T} r_{\boldsymbol{\mu}}(S_t)$ and the following performance index:

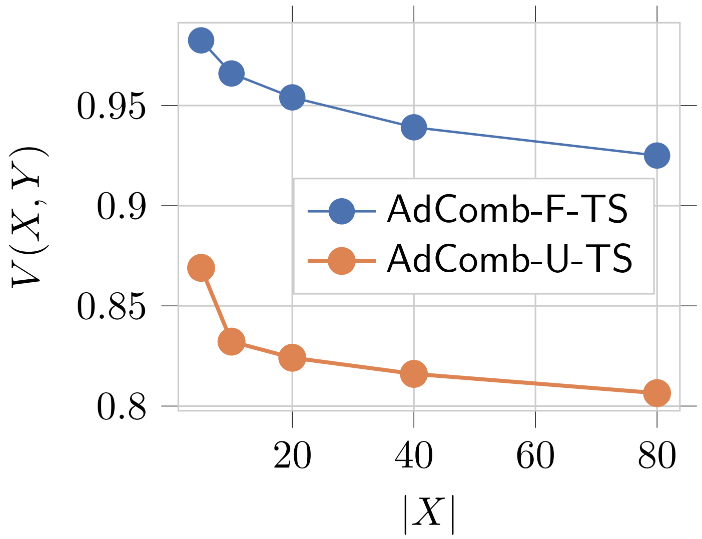
$$V(X, Y) = \frac{P_T(\mathfrak{U})}{T\, r^*_{\boldsymbol{\mu}}},$$

where both the algorithms selecting S_t and the clairvoyant algorithm selecting S^* (corresponding to $r^*_{\boldsymbol{\mu}}$) are run on the space $X \times Y$. [10] Basically, $V(X, Y) \in [0, 1]$ is a ratio returning, given a space of arms $X \times Y$, the efficiency of a learning algorithm with respect to the optimal solution achievable with that arm space. The normalization with respect to the optimal solution achievable with a given arm space mitigates the fact that, enlarging the arms space, the optimal revenue may increase.

[10] The results for AdComb-U-UCB and AdComb-F-UCB are omitted since they are in line with the ones we present and do not provide any further insight.



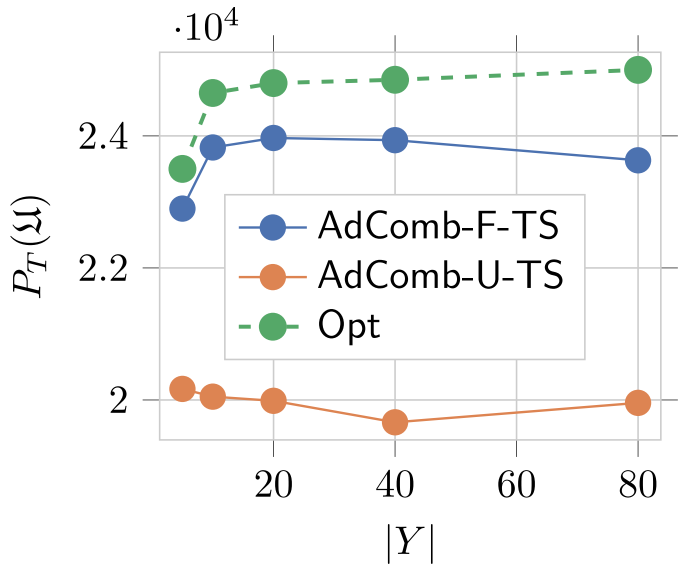
(a)



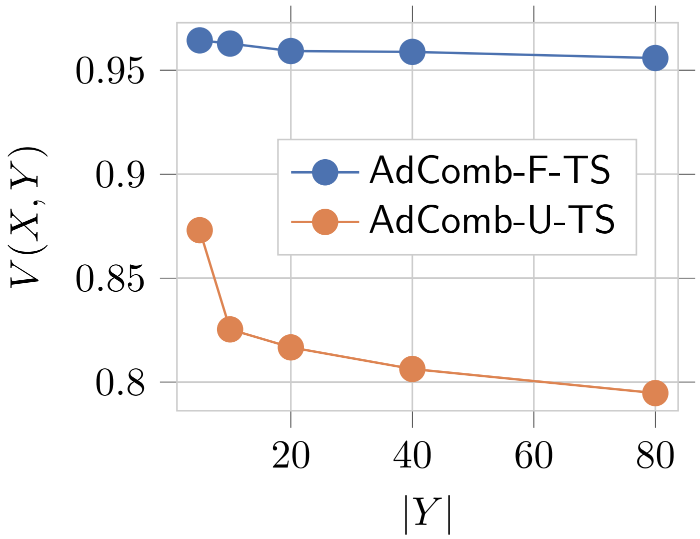
(b)

Figure 5.5: *Results for Experiment #2: performance of the AdComb-F-TS and AdComb-U-TS algorithms as $|X|$ varies: (a) cumulative expected revenue, (b) $V(X,Y)$ index.*

In Figure 5.5a, we show the value of $P_T(\mathfrak{U})$ for our algorithms and of the optimal solution (denoted with Opt) as the bid space granularity $|X|$



(a)



(b)

Figure 5.6: *Results for Experiment #2: performance of the AdComb-F-TS and AdComb-U-TS algorithms as $|Y|$ varies: (a) cumulative expected revenue, (b) $V(X,Y)$ index.*

varies. It is worth noting that the increase in the clairvoyant optimal reward for $|X| \geq 10$ is negligible. The cumulative revenue provided by the

AdComb-F-TS algorithm is decreasing for $|X| > 40$. However, the loss w.r.t. the revenue gained when $|X| = 80$ is about 3%. The AdComb-U-TS algorithm has a similar behavior. This result shows that the cost due to the exploration of a larger space of arms is larger than the increase of the optimal achievable reward, and it suggests, in practice, the adoption of a discretization of the bid space with about $|X| = 40$ evenly-spaced values. In Figure 5.5b, we show the values of $V(X, Y)$ achieved by the AdComb-U-TS and AdComb-F-TS algorithms. In this case, for both algorithms, the value of $V(X, Y)$ decreases as $|X|$ increases. This is because the algorithms pays a larger exploration cost. However, the empirical increase in inefficiency is only logarithmic in $|X|$. This result shows that the performance of the algorithms is robust to an increase of the number of possible bids.

In Figure 5.6a, we show the value $P_T(\mathfrak{U})$ of our algorithms and of the optimal solution as $|Y|$ varies. The results are similar to those obtained above when $|X|$ varies. The only peculiarity, in this case, concerns the performance of the AdComb-F-TS algorithm. Despite the theoretical analysis provides a logarithmic dependency of the regret from $|Y|$, the empirical results seem to suggest that the values of $V(X, Y)$ are approximately constant as $|Y|$ varies. This empirical result does not hold for the AdComb-U-TS algorithm, whose performance are significantly affected by the number of daily budget intervals. Intuitively, this is because, in the factorized model, the daily budget is not an input to the GPs we use, thus, it does not affect the exploration of the algorithm. Conversely, in AdComb-U-TS, the daily budget values constitute a part of the input to the GPs, and, therefore, an increase of the number of the daily budget values results in a larger regret since the algorithm needs to explore a wider space of arms.

In conclusion, the use of a more fine-grained bid/daily budget space to explore provides less revenue overall, but with a mild decrease in terms of performance, allowing, in practical cases, to use a large discretization space.

Experiment $\#3$

This experiment aims at evaluating the performance of our algorithms with random realistic settings generated by exploiting *Yahoo!* Webscope $A3$ dataset. More specifically, we consider $N = 4$ campaigns whose parameters μ_j, σ_j, γ_j, δ_j are those reported in Table 5.2. The values of the parameters $\mu^{(b)}$, $\sigma^{(b)}$, $p^{(obs)}(\delta)$, $p_j^{(cl)}$ are, instead, generated according to distributions estimated from the auctions of the *Yahoo!* Webscope $A3$ dataset.

We set a constant cumulative daily budget $\overline{B}_t = 100$ over a time horizon of $T = 100$ days, with limits $\underline{y}_{j,t} = 0$, $\overline{y}_{j,t} = 100$, $\underline{x}_{j,t} = 0$, and $\overline{x}_{j,t} = 1$ for every $t \leq T, C_j$. Furthermore, we use an evenly spaced discretization of $|X| = 10$ values of bid and $|Y| = 10$ values of daily budget. We generate 10 different scenarios, for each of them, we run 100 independent experiments over the same scenarios and averaged over them. Given a setting and an algorithm, we denote with β the percentage of runs in which the given algorithm has the best performance in terms of cumulative reward in the given setting.

In Table B.3, we report, for every algorithm and every setting, the average cumulative regret R_T, its standard deviation σ_{R_T}, and β. In almost all the settings, the best algorithm is AdComb-F-TS. Furthermore, AdComb-F-TS outperforms the other algorithms in more than the 70% of the runs in all settings. However, it is worth to note that in settings $1 - 5 - 6$, for $t \leq 25$, AdComb-F-UCB outperforms the other algorithms in more than $11\%, 73\%, 7\%$ of the cases, respectively. This provides evidence that only in some specific scenarios AdComb-F-UCB provides a viable solution to the ads optimization problem. Conversely, AdComb-U-TS and AdComb-U-UCB achieve lower performance than AdComb-F-TS and AdComb-F-UCB algorithms in all runs. For this reason, in the real-world setting, we adopt the AdComb-F-TS algorithm.

Evaluation in a Real-world Setting

We adopted the AdComB-F-TS algorithm to advertise in Italy a set of campaigns for a loan product of a large international company. The goal was the maximization of the number of the leads. Due to reasons of industrial secrecy, we cannot disclose the name of the product and the name of the company. The campaigns started July 1^{st} 2017 and were active up to December 31^{st} 2019. In our discussion, we report the results corresponding to the first 365 days of the experiment, grouped by weeks as the behavior of the users can be slightly different during the days of a single week and some campaigns were active only some specific days of the week. During these 365 days, the set of campaigns changed over time, some being added, others being discarded or changed, due to business needs such as, *e.g.*, the creation of new graphical logos, messages, or new user profiles to target. In particular, the total number of campaigns used is 29, while, initially, the campaigns were 8. The activation/deactivation of the campaigns in time and their actual costs per week are depicted in Figure 5.7. After 365 days, the previously active campaigns were completely discarded and a new set

		AdComb-F-TS			AdComb-F-UCB			AdComb-U-TS			AdComb-U-UCB		
		R_T	σ_{R_T}	β	R_T	σ_{R_T}	β	R_T	σ_{R_T}	β	R_T	σ_{R_T}	β
Setting 1	$t = 25$	73	13	89%	104Â	21	11%	287	24	0%	264	23	0%
	$t = 50$	111	22	97%	169	24	3%	377	28	0%	345	30	0%
	$t = 100$	170	37	99%	279	29	1%	485	35	0%	444	36	0%
Setting 2	$t = 25$	66	16	98%	125	15	2%	255	21	0%	263	21	0%
	$t = 50$	93	21	98%	155	18	2%	327	26	0%	340	24	0%
	$t = 100$	132	31	92%	194	22	8%	415	28	0%	424	30	0%
Setting 3	$t = 25$	75	15	100%	132	18	0%	319	31	0%	316	30	0%
	$t = 50$	103	21	100%	182	20	0%	421	37	0%	397	32	0%
	$t = 100$	145	32	100%	261	24	0%	533	42	0%	488	40	0%
Setting 4	$t = 25$	67	15	100%	130	29	0%	334	27	0%	306	26	0%
	$t = 50$	103	19	100%	196	38	0%	414	31	0%	395	29	0%
	$t = 100$	164	29	100%	297	55	0%	512	38	0%	499	35	0%
Setting 5	$t = 25$	112	18	27%	99	12	73%	345	30	0%	321	25	0%
	$t = 50$	157	22	83%	180	13	17%	479	39	0%	457	30	0%
	$t = 100$	222	24	100%	331	14	0%	648	62	0%	627	36	0%
Setting 6	$t = 25$	100	15	93%	99	9	7%	272	20	0%	287	24	0%
	$t = 50$	140	19	100%	180	13	0%	370	32	0%	391	28	0%
	$t = 100$	221	32	100%	331	20	0%	480	37	0%	507	34	0%
Setting 7	$t = 25$	100	15	98%	142	14	2%	336	24	0%	344	28	0%
	$t = 50$	145	19	94%	184	13	6%	453	34	0%	456	33	0%
	$t = 100$	220	30	83%	250	14	17%	595	43	0%	587	37	0%
Setting 8	$t = 25$	90	14	98%	144	22	2%	296	30	0%	278	24	0%
	$t = 50$	128	17	99%	220	21	1%	386	30	0%	363	28	0%
	$t = 100$	181	20	100%	303	27	0%	495	35	0%	466	34	0%
Setting 9	$t = 25$	89	17	90%	127	21	10%	334	23	0%	329	26	0%
	$t = 50$	119	23	91%	162	24	9%	417	28	0%	421	34	0%
	$t = 100$	165	33	88%	224	32	12%	510	36	0%	519	40	0%
Setting 10	$t = 25$	91	20	100%	197	10	7%	331	25	0%	302	29	0%
	$t = 50$	138	23	100%	248	16	0%	424	31	0%	402	39	0%
	$t = 100$	214	35	100%	333	22	0%	537	38	0%	518	44	0%

Table 5.3: *Results for Experiment $\#3$: performance of algorithms AdComb-F-TS, AdComb-F-UCB, AdComb-U-TS, AdComb-U-UCB with 10 different random settings at round $t \in \{25, 50, 100\}$.*

of campaigns was used.

The campaigns were optimized by human specialists from week 0 to week 5^{th} and by our algorithm from week 6^{th} on. In addition, the cumulative daily budget was changed at weeks 6^{th} and 29^{th} due to business reasons of the company, as follows:

- 200 Euros per day from week 0^{th} to week 5^{th};
- $1,100$ Euros per day from week 6^{th} to week 28^{th} (the increase in the daily budget was motivated to obtain more leads);
- 700 Euros per day from week 29^{th} on (the decrease in the daily budget was motivated to reduce the cost per lead).

The algorithm was 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. We used a discretization of the bid and daily budget space such that $|X| = 100$ and $|Y| = 500$. Furthermore, the estimates are based on the data collected in the last 20 weeks, thus using a sliding window of 140 days, to discard observations that were considered excessively old by human specialists. The algorithm ran at midnight, collecting the observations of the day before, updating the models, computing the next values bid/daily budget to use, and, finally, setting these values on the corresponding platforms. The total computing time of the algorithm per execution was shorter than 5 minutes.

The cost per lead and the actual costs per week are reported in Figures 5.8 and 5.9, respectively. The cost per lead dramatically reduced thanks to the activation of algorithm at the 6^{th} week, from an average of about 120 Euros per lead to about 65 Euros per lead. Furthermore, the algorithm spent about 4 weeks to reduce the cost per lead from about 65 Euros (at the 6^{th} week) to about 50 Euros (at the 10^{th} week). The algorithm exhibited a rather explorative behavior until the 15^{th} week due to the need to collect samples, whereas, from the 16^{th} week to the 26^{th} week, the algorithm presented a rather stable behavior. Notably, even if the algorithm was rather explorative up to the 15^{th} week, the performance in these weeks was evaluated rather stable by the human specialists. The oscillations from the 15^{th} week to the 28^{th} week were due to seasonability effects and the campaigns of the competitors. More precisely, the human specialists confirmed that from the 12^{th} week (beginning of September, corresponding to the conclusion of the summer holidays in Italy) to the 18^{th} week (middle of October), the users are usually less interested to make loans. This behavior leaded to an increase of the cost per lead. A similar effect was observed from the 22^{th} week to the 27^{th} week (corresponding to the period from the

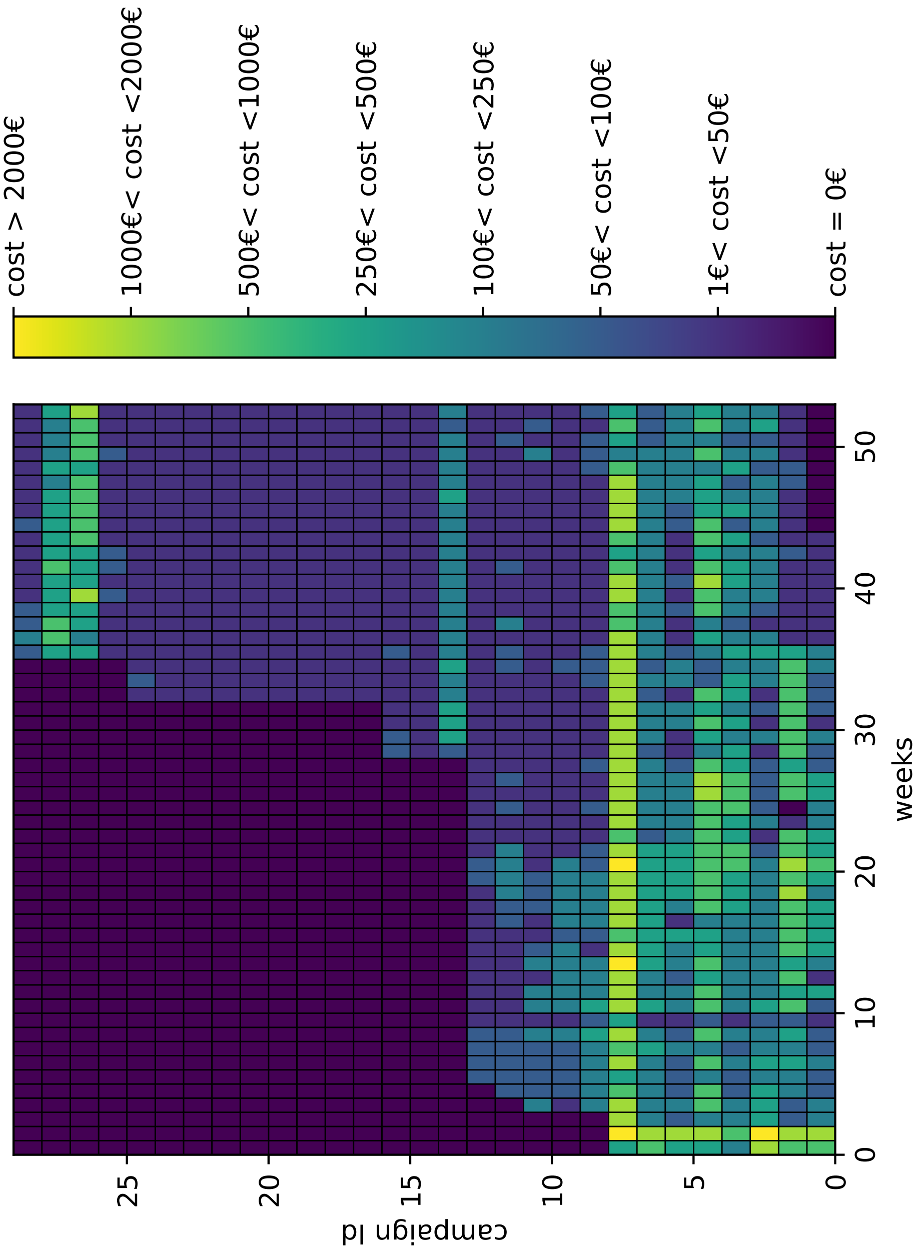


Figure 5.7: *Actual costs for the campaigns. A cost of* 0 *Euros means that the campaign is not active.*

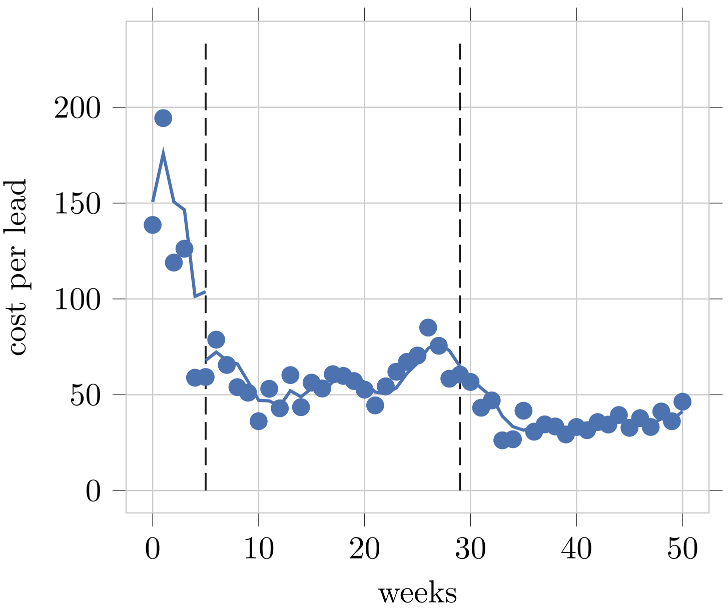


Figure 5.8: *Cost per lead across weeks.*

beginning of December to the middle of January). At the 29th week, the reduction of the daily budget pushed the algorithms to explore better the values of the functions to learn for smaller values of the daily budget. This task required 4 weeks, in which the cost per lead reduced from about 65 Euros to about 35 Euros.

The actual cost per week, differently from the cost per lead, was subject to prominent oscillations. These oscillations were due to seasonal effects and the possibility of overspending on the platform (*i.e.*, even if the platforms allow the introduction of daily budget constraints, these constraints can be violated by the platforms). The evaluation of the performance of our algorithm was very positive for the company, as to motivate its adoption for other sets of campaigns.

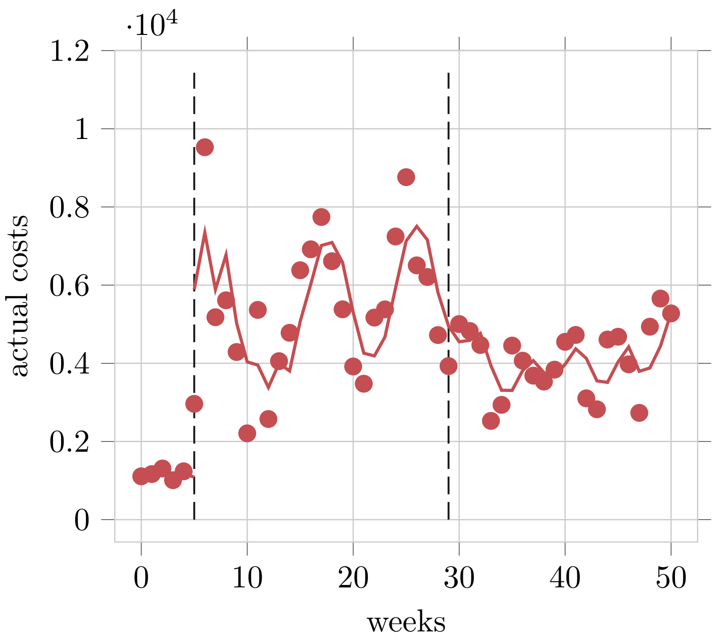


Figure 5.9: *Actual costs across weeks.*

CHAPTER 6

Safe Online Bid Optimization with Return-on-Investment Constraints

Bid optimization is one of the main tasks that advertisers have to face to control campaigns performance. Usually, they have to set this value to balance the tradeoff between achieving *high volumes*, maximizing the sales of the products to advertise, and *high profitability*, maximizing the Return-On-Investment (ROI). However, this problem has been studied in different scenarios and from different point of views. Here, we focus on a specific scenario characterized by two peculiarities. Firstly, we consider a setting in which the spend of a sub-campaign is not limited by the daily budget. This is typical of some platforms (*e.g.,* Google Hotels[1] and other hotel advertising platforms) that do not allow to set a daily budget to limit the daily spend of a sub-campaign. Secondly, we focus on the scenario in which the advertiser has to satisfy some ROI constraints not only at the end but for the whole advertising period. This is a common requirement that media-agencies have to satisfy to meet the their customers' goals.

In this chapter, we propose a novel formulation of the Internet campaign optimization with ROI constraints, and present an online learning algorithm

[1]`https://www.google.com/travel/hotels`

for safe bidding optimization, called $\mathsf{GCB}_{\texttt{safe}}$that satisfies the constraints with high probability.

We initiate the investigation of combinatorial learning algorithms for bidding in advertising scenarios with budget and ROI constraints when Gaussian Processes (GPs) are used to model the problem's parameters. In particular, we show that the optimization problem with ROI and budget constraints cannot be approximated within any strictly positive factor unless $\mathsf{P} = \mathsf{NP}$ when the values of all the parameters are known. However, when dealing with a bids' discretized space as it happens in practice, the problem admits a pseudo-polynomial time algorithm based on dynamic programming, computing the optimal solution. Remarkably, we prove that no online learning algorithm violates the ROI constraint less than a linear number of times while guaranteeing sublinear pseudo-regret. We show that we can obtain sublinear pseudo-regret adopting the GCB algorithm, a specific version of the combinatorial bandit algorithm proposed by [3]. However, it violates the ROI constraints during the learning process and at convergence. Then, we propose $\mathsf{GCB}_{\texttt{safe}}$, a novel algorithm that guarantees the satisfaction of the ROI constraints with high probability, but provides linear pseudo-regret. We experimentally evaluate the performances of the GCB and $\mathsf{GCB}_{\texttt{safe}}$ algorithms, showing the tradeoff between pseudo-regret and constraint-violation, and, finally, we analyze the $\mathsf{GCB}_{\texttt{safe}}$ sensitivity.

Problem Formulation

We are given an advertising campaign $\mathcal{C} = \{C_1, \dots, C_N\}$, where C_j is the j-th sub-campaign, and a finite time horizon of $T \in \mathbb{N}$ days.[2] For each day $t \in \{1, \dots, T\}$ and for every sub-campaign C_j, the advertiser needs to specify the bid $x_{j,t} \in X_j$, where $X_j \subset \mathbb{R}^+$ is a finite set of bids we can set in sub-campaign C_j. The goal is, for every day $t \in \{1, \dots, T\}$, to find the values of bids that maximizes the overall cumulative expected revenue while keeping the overall ROI above a fixed value $\lambda^* \in \mathbb{R}^+$ and the overall budget below a daily value $y_t \in \mathbb{R}^+$. This setting is modeled as

[2] In this work, as common in the literature on ad allocation optimization, we refer to a sub-campaign as a single ad or a group of homogeneous ads requiring to set the same bid.

a constrained optimization problem at a day t, as follows:

$$\max_{x_{j,t} \in X_j} \sum_{j=1}^{N} v_j \; n_j(x_{j,t}) \tag{6.1a}$$

$$\text{s.t.} \; \frac{\sum_{j=1}^{N} v_j \; n_j(x_{j,t})}{\sum_{j=1}^{N} \; c_j(x_{j,t})} \geq \lambda^* \tag{6.1b}$$

$$\sum_{j=1}^{N} c_j(x_{j,t}) \leq \overline{y} \tag{6.1c}$$

where $n_j(x_{j,t})$ and $co_j(x_{j,t})$ are the expected number of clicks and the expected cost given the bid $x_{j,t}$ for sub-campaign C_j, respectively, and v_j is the value per click for sub-campaign C_j. Moreover, Constraint (6.1b) is the ROI constraint, forcing the revenue to be at least λ^* times the incurred costs, and Constraint (6.1c) keeps the daily spend under a predefined overall budget y_t.[3]

In our setting, the available options consist in the different values of the bid $x_{j,t} \in X_j$ satisfying the combinatorial constraints of the optimization problem, while $n_j(\cdot)$ and $c_j(\cdot)$ are unknown functions, defined on the feasible region of the variables, that we need to estimate within the time horizon T.[4] A learning policy $\mathfrak{U}$ solving such a problem is an algorithm returning, for each day t, a set of bid $\{\hat{x}_{j,t}\}_{j=1}^{N}$. The policy $\mathfrak{U}$ can only use estimates of the unknown number-of-click and cost functions built during the learning process. Therefore, the returned solutions may not be optimal and/or violate Constraints (6.1b) and (6.1c) computed on the true functions.[5] We are interested in evaluating learning policies $\mathfrak{U}$ in terms of both loss of revenue (a.k.a. pseudo-regret) and violation of those constraints. The pseudo-regret and safety of a learning policy $\mathfrak{U}$ are defined as follows: [Learning policy pseudo-regret] Given a learning policy $\mathfrak{U}$, we define the *pseudo-regret* as:

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

where $G^* := \sum_{j=1}^{N} v_j \; n_j(x_j^*)$ is the expected value provided by a clairvoyant algorithm, the set of bid $\left\{x_j^*\right\}_{j=1}^{N}$ is the optimal clairvoyant solution to

[3] In economic literature, it is also used an alternative definition of ROI: $\frac{\sum_{j=1}^{N}[v_j \; n_j(x_{j,t}) - co_j(x_{j,t})]}{\sum_{j=1}^{N} \; co_j(x_{j,t})}$. To capture this case, it is sufficient to substitute the right hand of Constraint (6.1b) with $\lambda^* + 1$.

[4] Here, we assume the value per click v_j is known. In the case one needs its estimates, refer to [41] for details.

[5] Some platforms allow the advertisers to set a daily budget constraint. If this feature is allowed, Constraint (6.1c) is always satisfied, and no safety requirement for that constraint is necessary.

Algorithm 2 Meta-algorithm

```
1:  Input: sets $X_j$ of bid values, ROI threshold $\lambda^*$
2:  Initialize the GPs for the number of clicks and costs
3:  for $t \in \{1, \dots, T\}$ do
4:      for $j \in \{1, \dots, N\}$ do
5:          for $x \in X_j$ do
6:              Compute $\hat{n}_{j,t}(x)$ and $\hat{\sigma}^n_{j,t}(x)$ using the GP on the number of clicks
7:              Compute $\hat{c}_{j,t}(x)$ and $\hat{\sigma}^c_{j,t}(x)$ using the GP on costs
8:      Compute $\boldsymbol{\mu}$ using the GPs estimates
9:      Run the $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ procedure to get a solution $\{\hat{x}_{j,t}\}_{j=1}^N$
10:     Set the prescribed allocation during day $t$
11:     Get revenue
12:     Update the GPs using the new information $\tilde{n}_{j,t}(\hat{x}_{j,t})$ and $\tilde{c}_{j,t}(\hat{x}_{j,t})$
```

the problem in Equations (6.1a)–(6.1c), and the expectation $\mathbb{E}[\cdot]$ is taken w.r.t. the stochasticity of the learning policy $\mathfrak{U}$.

[η-safe learning policy] Given $\eta \in (0,\ 1)$, a learning policy $\mathfrak{U}$ is *η-safe* if $\{\hat{x}_{j,t}\}_{j=1}^N$, *i.e.,* the allocations it selects during the days $t \in \{1, \dots, T\}$, violate the Constraints (6.1b) and (6.1c) with probability less than η or, formally:

$$\sum_{t=1}^{T} \mathbb{P}\left(\frac{\sum_{j=1}^N v_j \; n_j(\hat{x}_{j,t})}{\sum_{j=1}^N \; co_j(\hat{x}_{j,t})} < \lambda^* \vee \sum_{j=1}^N c_j(\hat{x}_{j,t}) > y_t \right) \leq \eta.$$

Hence, our goal is the design of algorithms minimizing the pseudo-regret $R_T(\mathfrak{U})$ while selecting each day t bids $\{\hat{x}_{j,t}\}_{j=1}^N$ which are feasible solution to the problem in Equations (6.1a)–(6.1c).

Proposed Method: the GCB$_{\texttt{safe}}$ Algorithm

Meta-algorithm

We provide the pseudo-code of our meta-algorithm in Algorithm 2. It solves the optimization problem stated in the previous section in an online fashion. Algorithm 2 is based on three components: Gaussian Processes (GPs) [48] to model the parameters whose values are unknown, an *estimation subroutine* to generate estimates of the parameters from the GPs, and an *optimization subroutine* to solve the optimization problem given the estimates.

In particular, GPs are used to model the functions $n_j(\cdot)$ and $c_j(\cdot)$ describing the number of clicks and the costs, respectively. The application of GPs to model these functions provide several advantages w.r.t. other regression techniques. More precisely, GPs provide an estimate on the entire bid domain X_j relying on a finite set of samples. For every bid value $x \in X_j$, they return a probability distribution over the functions' possible values, which is an uncertainty measure of the estimates. GPs use the noisy realization of the number of clicks $\tilde{n}_{j,h}(\hat{x}_{j,h})$ collected from each sub-campaign C_j for each past day $h \in \{1, \ldots, t-1\}$ to generate $\hat{n}_{j,t-1}(x)$ and $\hat{\sigma}^n_{j,t-1}(x)$, *i.e.*, the estimates for the expected value and the standard deviation of the number of clicks, respectively, for each bid $x \in X_j$. The same holds for the GPs used to model the cost functions $c_j(\cdot)$, that provides, $\hat{c}_{j,t-1}(x)$ and $\hat{\sigma}^c_{j,t-1}(x)$, *i.e.*, the estimates for the expected value and the standard deviation of the costs for sub-campaign C_j, respectively, using the noisy realizations of the cost function $\tilde{c}_{j,h}(\hat{x}_{j,h})$, with $h \in \{1, \ldots, t-1\}$. Details on the initialization and adoption of the GPs are provided by Rasmussen and Williams [48]. The estimation subroutine returns the vector $\boldsymbol{\mu}$ composed of the estimates generated from the GPs. In the following sections, we provide two subroutines to compute $\boldsymbol{\mu}$, and we analyze their impact in terms of pseudo-regret and safety. Then, the vector $\boldsymbol{\mu}$ is given in input to the optimization subroutine, called $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$, that solves the problems stated in Equations (6.1a)–(6.1c) and returns the bid strategy $\{\hat{x}_{j,t}\}_{j=1}^N$ to play the next day t. Finally, once the strategy has been applied, the stochastic realization of the number of clicks $\tilde{n}_{j,t}(\hat{x}_{j,t})$ and costs $\tilde{c}_{j,t}(\hat{x}_{j,t})$ are observed and provided to the GPs to update the models used for the next day $t+1$.

For the sake of presentation, we first present the $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ subroutine and, then, we describe some estimation subroutines.

Optimization Subroutine

At first, we show that, even if all the values of the parameters of the optimization problem are known, the optimal solution cannot be approximated in polynomial time within any strictly positive factor, unless $\mathsf{P} = \mathsf{NP}$. We reduce from the NP-hard problem SUBSET-SUM. Given a set S of integers $u_i \in \mathbb{N}$ and an integer $z \in \mathbb{N}^+$, SUBSET-SUM requires to decide whether there is a set $S^* \subseteq S$ with $\sum_{i \in S^*} u_i = z$.

Theorem 5 (Inapproximability of the optimization problem)**.** *For any $\rho \in (0, 1]$, there is no polynomial-time algorithm returning a ρ-approximation to the problem in Equations* (6.1a)–(6.1c)*, unless* $\mathsf{P} = \mathsf{NP}$.

It is well known that SUBSET-SUM is a weakly NP-hard problem and, therefore, there is an algorithm whose running time is polynomial in the dimension of the problem and the magnitudes of the data involved rather than

Algorithm 3 $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ subroutine

```
1: Input: sets $X_j$ of bid values, set $Y$ of cumulative cost values, set $R$ of revenue values,
   vector $\boldsymbol{\mu}$, ROI threshold $\lambda^*$
2: Initialize $M$ empty matrix with dimension $|Y| \times |R|$
3: Initialize $\mathbf{x}^{y,r} = \mathbf{x}^{y,r}_{\text{next}} = [\,], \forall y \in Y, r \in R$
4: $A(y,r) = \bigcup \{x \in X_1 | \, \overline{c}_1(x) \leq y \wedge \underline{w}_1(x) \geq r\} \;\; \forall y \in Y, r \in R$
5: $\mathbf{x}^{y,r} = \arg\max_{x \in S} \overline{w}_1(x) \;\; \forall y \in Y, r \in R$
6: $M(y,r) = \max_{x \in S} \overline{w}_1(x) \;\; \forall y \in Y, r \in R$
7: for $j \in \{2, \dots N\}$ do
8:     for $y \in Y$ do
9:         for $r \in R$ do
10:            Update $A(y,r)$ according to Equation (6.2)
11:            $\mathbf{x}^{y,r}_{\text{next}} = \arg\max_{\mathbf{s} \in S(y,r)} \sum_{i=1}^{j} \overline{w}_i(s_i)$
12:            $M(y,r) = \max_{\mathbf{s} \in S(y,r)} \sum_{i=1}^{j} \overline{w}_i(s_i)$
13:    $\mathbf{x}^{y,r} = \mathbf{x}^{y,\lambda}_{\text{next}}$
14: Select $(y^*, r^*)$ according to Equation (6.3)
15: Output: $\mathbf{x}^{y^*,r^*}$
```

the base-two logarithms of their magnitudes. The same can be showed for our problem. Indeed, we can design a pseudo-polynomial-time algorithm to find the optimal solution in polynomial time w.r.t. the number of possible values of revenues and costs. In real-world settings, the values of revenue and cost are in limited ranges and rounded to the nearest cent, allowing the problem to be solved in a reasonable time. From now on, we assume that the discretization of the ranges of the values of the daily cost Y and revenue R is evenly spaced.

The pseudo-code of the $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ subroutine solving the problem in Equations (6.1a)–(6.1c) with a dynamic programming approach, is provided in Algorithm 3. It takes as input the set of the possible bid values X_j for each sub-campaign C_j, the set of the possible cumulative cost values Y, the set of the possible revenue values R, a ROI threshold λ^*, and a vector of parameters characterizing the specific instance of the optimization problem:

$$\boldsymbol{\mu} := \Big[\overline{w}_1(x_1), \dots, \overline{w}_N(x_{|X_N|}), \underline{w}_1(x_1), \dots, \\ \underline{w}_N(x_{|X_N|}), -\overline{c}_1(x_1), \dots, -\overline{c}_N(x_{|X_N|})\Big],$$

where $w_j(x_j) := v_j \, n_j(x_j)$ denotes the revenue for a sub-campaign C_j. We use $\overline{h}$ and $\underline{h}$ to denote potentially different estimated values of a generic function h used by the learning algorithms in the next sections. In particular, if the functions are known beforehand, then it holds $\overline{h} = \underline{h} = h$ for both $h = w_j$ and $h = c_j$. For the sake of clarity, $\overline{w}_j(x)$ is used in the evaluation of the revenue of a solution, while $\underline{w}_j(x)$ and $\overline{c}_j(x)$ are used in the constraints. Notice that we do not need to check the budget constraint as long as the condition $\max(Y) = y_t$ does not allow solutions with unfeasible cumulated costs. At first, the subroutine initializes a matrix M in which it stores the optimal solution for each combination of values in $y \in Y$ and in $r \in R$, and initializes the vectors $\mathbf{x}^{y,r} = \mathbf{x}^{y,r}_{\text{next}} = [\ ], \forall y \in Y, r \in R$ (Lines 2–3). Then, the subroutine generates the set $A(y, r)$ of bids for sub-campaign C_1 (Line 4). More precisely, the set $A(y, r)$ contains only the bids x that induce the overall costs to be lower than y and the overall revenue to be higher than r. The bid in $A(y, r)$ that maximizes the revenue calculated with parameters $\overline{w}_j$ is included in the vector $\mathbf{x}^{y,r}$, while the corresponding revenue is stored in the matrix M. Then, the subroutine iterates over sub-campaigns C_j with $j \in \{2, \dots, N\}$, over all the values $y \in Y$, and over all the values $r \in R$ (Lines 10–12). At each iteration, the subroutine, for every pair (y, r), stores in $\mathbf{x}^{y,r}$ the optimal set of bids for sub-campaigns $C_1, \dots, C_j$ that maximizes the objective function and stores the corresponding optimum value in $M(y, r)$. At every j-th iteration, the computation of the optimal bids is performed by evaluating a set of candidate solutions $A(y, r)$ computed as follows:

$$S(y,r) := \bigcup \Big\{ \mathbf{s} = [\mathbf{x}^{y',r'}, x] \text{ s.t. } y' + \overline{c}_j(x) \leq y \wedge \\ r' + \underline{w}_j(x) \geq r \wedge x \in X_j \wedge y' \in Y \wedge r' \in R \Big\}. \tag{6.2}$$

This set is built by combining the optimal bids $\mathbf{x}^{y',r'}$ computed at the $(j-1)$-th iteration with one of the bids $x \in X_j$ available for the j-th sub-campaign, s.t. these combinations satisfy the ROI and budget constraints. Then, the subroutine assigns the element of $A(y, r)$ that maximizes the revenue to $\mathbf{x}^{y,r}_{\text{next}}$ and the corresponding revenue to $M(y, r)$. At the end, the subroutine computes the optimal pair (y^*, r^*) as follows:

$$(y^*, r^*) = \Big\{ y \in Y, r \in R \text{ s.t. } \frac{r}{y} \geq \lambda^* \wedge \\ M(y, r) \geq M(y', r') \quad \forall y' \in Y, \forall r' \in R \Big\}, \tag{6.3}$$

as well as the corresponding set of bids $\mathbf{x}^{y^*,r^*}$, containing one bid for each sub-campaign. We can state the following:

Theorem 6 (Optimality). *Subroutine* $\textsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ *using* $\overline{w}_j(x) = \underline{w}_j(x) = v_j\, n_j(x)$ *and* $\overline{c}_j(x) = c_j(x)$ *for each* $j \in \{1, \dots, N\}$ *returns the optimal* *solution to the problem in Equations* (6.1a)–(6.1c) *when the values of revenues and costs are in R and Y, respectively.*

The above algorithm is exact. Thus, according to [14], it corresponds to an (α, β)-approximation oracle with $\alpha = \beta = 1$.

Estimation Subroutine

In this section, we first provide an impossibility result and, subsequently, two estimation subroutines.

Impossibility Result We show that no online learning algorithm can provide a sublinear pseudo-regret while guaranteeing to be safe with high probability. For simplicity, our result is based on the violation of the budget constraint (*i.e.*, Constraint (6.1c)), but it can be extended to the ROI constraint (*i.e.*, Constraint (6.1b)).

Theorem 7 (Pseudo-regret/safety tradeoff)**.** *For every $\epsilon > 0$ and time horizon T, there is no algorithm with pseudo-regret smaller than $T(1/2 - \epsilon)$ that violates the constraint on the budget less than $T(1/2 - \epsilon)$ times in expectation.*

In the following, we provide two estimation subroutine, the former providing sublinear pseudo-regret, the latter guaranteeing safety with high probability.

Guaranteeing Sublinear Pseudo-regret: GCB [3] provide a combinatorial bandit algorithm in which the reward is modeled by a single GP. The extension of the GCB algorithm to the case in which multiple parameters of the optimization problem are modeled by independent GPs is direct. To guarantee a sublinear pseudo-regret, we need that a few assumptions are satisfied. More specifically, we need *Lipschitz continuity* assumption between the parameter vector $\boldsymbol{\mu}$ and the value returned by the objective function in Equation (6.1a), and a *monotonicity property*, stating that the value of the objective function increases as the values of the elements in $\boldsymbol{\mu}$ increase. While it is easy to show that Lipschitz continuity holds with constant $\Lambda = N$ (number of sub-campaigns), the monotonicity property holds by definition of $\boldsymbol{\mu}$, as the increase of one value of $\overline{w}_j(x)$ would increase the value of the objective function, and the increase of the values of $\underline{w}_j(x)$ or $\overline{c}_j(x)$ would enlarge the feasibility region of the problem, thus not excluding optimal solutions.

Furthermore, we need that, given $\delta \in (0, 1)$, $\overline{w}_j(x)$ and $\underline{w}_j(x)$ are statistical upper bounds for the actual values $n_j(x)$ and that $\overline{c}_j(x)$ are statistical lower bounds for the actual values $c_j(x)$ holding for all $x \in X_j$ and for all $j \in \{1, \dots, N\}$ with probability at least $1 - \delta$ for $t \in \{1, \dots, T\}$. This last requirement is satisfied if we define:

$$\overline{w}_j(x) = \underline{w}_j(x) := v_j \left[\hat{n}_{j,t-1}(x) + \sqrt{b_{t-1}} \hat{\sigma}^n_{j,t-1}(x) \right], \tag{6.4}$$

$$\overline{c}_j(x) := \hat{c}_{j,t-1}(x) - \sqrt{b_{t-1}} \hat{\sigma}^c_{j,t-1}(x), \tag{6.5}$$

where $b_t := 2 \ln \left(\frac{12NTt^2}{\delta \pi^2} \right)$ is an uncertainty term used to guarantee the confidence level required by GCB.[6] The pseudo-code of the GCB algorithm is the one in Algorithm 2 when using the above $\overline{w}_j(x)$, $\underline{w}_j(x)$, and $\overline{c}_j(x)$ for the vector of parameters $\boldsymbol{\mu}$. It guarantees sublinear pseudo-regret as follows:

Theorem 8 (GCB pesudo-regret)**.** *Given $\delta \in (0,\ 1)$, the GCB algorithm applied to the problem in Equations* (6.1a)–(6.1c)*, with probability at least $1 - \delta$, suffers from a pseudo-regret of:*

$$R_T(\textit{GCB}) \leq \sqrt{\frac{16TN^3 b_t}{\ln(1 + \sigma^2)} \sum_{j=1}^{N} \gamma_{j,T}},$$

where $\sigma \in \mathbb{R}^+$ and $\gamma_{j,T} \in \mathbb{R}^+$ are the noise standard deviation and the maximum information gain of a generic set of T samples for the GP modeling the number of clicks of sub-campaign C_j, respectively.

On the other hand, the GCB algorithm is not safe.

Theorem 9 (GCB safety)**.** *Given $\delta \in (0,\ 1)$, the probability that for at least a $t \in \{1, \dots, T\}$ the allocation returned by the GCB algorithm applied to the problem in Equations* (6.1a)–(6.1c) *violates at least one of the constraints is at least $1 - \frac{\delta}{2NT}$.*

Guaranteeing Safety: GCB$_{\texttt{safe}}$ We propose GCB$_{\texttt{safe}}$, a variant of the GCB algorithm relying on different bounds in $\boldsymbol{\mu}$. More specifically, we employ optimistic estimates for the parameters used in the objective function and

[6] For the sake of simplicity, we assume that the values of the bounds correspond to values in R and Y, respectively. If the bound values for $\overline{w}_j(x)$ are not in the set R, we need to round them up to the nearest value belonging to R. Instead, if $\underline{c}_j(x)$ are not in the set Y, a rounding down should be performed to the nearest value in Y.

pessimistic estimates for the parameters used in the constraints. Formally, in $\mathsf{GCB}_{\texttt{safe}}$, the estimates are chosen as follows:

$$\overline{w}_j(x) := v_j \left[\hat{n}_{j,t-1}(x) + \sqrt{b_{t-1}}\hat{\sigma}^n_{j,t-1}(x)\right],$$
$$\underline{w}_j(x) := v_j \left[\hat{n}_{j,t-1}(x) - \sqrt{b_{t-1}}\hat{\sigma}^n_{j,t-1}(x)\right],$$
$$\overline{c}_j(x) := \hat{c}_{j,t-1}(x) + \sqrt{b_{t-1}}\hat{\sigma}^c_{j,t-1}(x).$$

Furthermore, $\mathsf{GCB}_{\texttt{safe}}$ needs a default set of bids $\left\{x^{\mathsf{d}}_{j,t}\right\}_{j=1}^{N}$, that is known *a priori* to be feasible for the problem in Equations (6.1a)–(6.1c) with the actual values of the parameters.[7] The pseudo-code of $\mathsf{GCB}_{\texttt{safe}}$ is provided in Algorithm 2 with the above definition of the parameters of vector $\boldsymbol{\mu}$, except that it returns $\{\hat{x}_{j,t}\}_{j=1}^{N} = \left\{x^{\mathsf{d}}_{j,t}\right\}_{j=1}^{N}$ if the optimization problem does not admit any feasible solution with the current estimates.

It is easy to check that the probability with which the bounds $\underline{w}_j(x)$ and $\overline{c}_j(x)$ are larger than their true values is smaller than $\frac{\delta\pi^2}{6NTt^2}$ in both cases. This result does not allow the adoption of Theorem 1 by [3] to bound the pseudo-regret. However, this choice of the estimates allows us to show the following:

Theorem 10 ($\mathsf{GCB}_{\texttt{safe}}$ safety)**.** *Given $\delta \in (0,\ 1)$, the $\mathsf{GCB}_{\texttt{safe}}$ algorithm applied to the problem in Equations* (6.1a)–(6.1c) *is δ-safe.*

The safety property comes at the cost that the $\mathsf{GCB}_{\texttt{safe}}$ algorithm may suffer from a much larger pseudo-regret than GCB. More specifically, we show the following:

Theorem 11 ($\mathsf{GCB}_{\texttt{safe}}$ pseudo-regret)**.** *Given $\delta \in (0,\ 1)$, the $\mathsf{GCB}_{\texttt{safe}}$ algorithm applied to the problem in Equations* (6.1a)–(6.1c) *problem suffers from a pseudo-regret $R_t(\mathsf{GCB}_{\texttt{safe}}) = \mathcal{O}(T)$.*

Experimental Evaluation

We compare the GCB algorithm with $\mathsf{GCB}_{\texttt{safe}}$ in a synthetic setting in terms of revenue and safety.

Experiment #1 We simulate $N = 5$ sub-campaigns, with $|X| = 201$ bid values evenly spaced in $[0,\ 2]$, $|Y| = 101$ cost values evenly spaced in $[0,\ 100]$, and $|R| = 151$ revenue values evenly spaced in $[0,\ 1200]$. For

[7] A trivial default feasible bid allocation is $\left\{x^{\mathsf{d}}_{j,t} = 0\right\}_{j=1}^{N}$.

a generic sub-campaign C_j, we generate the number of clicks using the function $\tilde{n}_j(x) := \beta_j(1 - e^{-x/\delta_j}) + \xi_j^r$ and the cost $\tilde{c}_j(x) = \alpha_j(1 - e^{-x/\gamma_j}) + \xi_j^c$, where $\alpha_j \in \mathbb{R}^+$ and $\beta_j \in \mathbb{R}^+$ represent the maximum achievable cost and revenue for sub-campaign C_j in a single day, $\delta_j \in \mathbb{R}^+$ and $\gamma_j \in \mathbb{R}^+$ characterize how fast the two functions reach a saturation point, and ξ_j^r and ξ_j^c are noise terms drawn from a $\mathcal{N}(0, 1)$ Gaussian distribution.[8] We assume a unitary value for the clicks, *i.e.*, $v_j = 1$ for each $j \in \{1, \dots, N\}$. The values of the parameters of cost and revenue functions of the sub-campaigns are specified in Table B.1 reported in the Supplementary Material. We set a constant daily budget $y_t = 100$ for every t, and $\lambda^* = 10$ in ROI constraint, and a time horizon $T = 60$. The peculiarity of this setting is that, at the optimal solution, the budget constraint is active, while the ROI constraint is not (see Experiment #3 for a setting where the ROI constraint is active at the optimal solution).

For both GCB and GCB$_{\texttt{safe}}$, we use GPs with a squared exponential kernel of the form $k(x, x') := \sigma_f^2 \exp\left\{-\frac{(x-x')^2}{l}\right\}$ for each $x \in X_j$, where the parameters $\sigma_f \in \mathbb{R}^+$ and $l \in \mathbb{R}^+$ are estimated from observed data as indicated by the GP literature, see [48] for details. The confidence for the two algorithms is set to $\delta = 0.2$.

We evaluate the algorithms in terms of:

- daily revenue: $P_t(\mathfrak{U}) := \sum_{j=1}^N v_j n_j(\hat{x}_{j,t}, \hat{y}_{j,t})$;
- daily ROI: $ROI_t(\mathfrak{U}) := \frac{\sum_{j=1}^N v_j \ n_j(\hat{x}_{j,t})}{\sum_{j=1}^N \ co_j(\hat{x}_{j,t})}$;
- daily spend: $S_t(\mathfrak{U}) := \sum_{j=1}^N c_j(\hat{x}_{j,t})$.

We show the results obtained over 100 independent runs of the two algorithms.

In Figure B.7, for the daily revenue, ROI, and spend achieved by GCB and GCB$_{\texttt{safe}}$ at every t, we show the 50_{th} percentile (*i.e.*, the median) with solid lines and the 90_{th} and 10_{th} percentiles with dashed lines surrounding the semi-transparent area. While GCB achieves larger values of revenue, it violates the budget constraint over the entire time horizon and the ROI constraint in the first 7 days in more than 50% of the experiments. This happens because, in the optimal solution, the ROI constraint is not active, while the budget constraint is. Conversely, GCB$_{\texttt{safe}}$ satisfies the budget and ROI constraints over the time horizon for more than 90% of the executions, and has a slower convergence to the optimum revenue. If we focus on

[8] These functions are customarily used in the advertising literature, *e.g.*, by [31].

the median revenue, $\mathsf{GCB}_{\texttt{safe}}$ has similar behaviour to the GCB algorithm for $t > 15$. This makes $\mathsf{GCB}_{\texttt{safe}}$ a good choice even in terms of overall revenue.

However, it is worth to notice that, in the 10% of the executions, $\mathsf{GCB}_{\texttt{safe}}$ does not converge to the optimum before the end of the learning period. These results confirm the theoretical analysis showing that limiting the exploration to safe regions can lead the algorithm to get stuck in local optima.

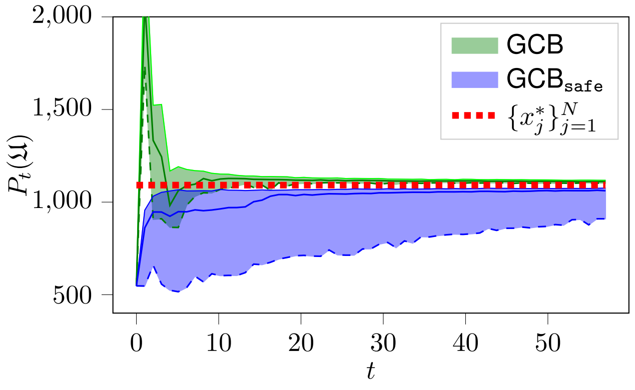
Experiment #2 In real-world scenarios, the business goals in terms of volumes/profitability tradeoff are often blurred, and sometimes can be desirable to slightly violate the constraints (usually, the ROI constraint) in favor of a significant volumes increase. However, analyzing and acquiring information about these tradeoff curves requires exploring volumes opportunities by relaxing the constraints. In this experiment, we show how our approach can be adjusted to address this problem in practice. We use the same setting of Experiment #1, except for the input we pass to the $\mathsf{GCB}_{\texttt{safe}}$ algorithm. More precisely, we relax the ROI constraint by multiplying λ^* by a value $\epsilon_x \in \{1.00, 0.95, 0.90, 0.85\}$, and we run 4 instances of $\mathsf{GCB}_{\texttt{safe}}$ each associated to a different ϵ_x value. As a result, except for the first instance, we allow $\mathsf{GCB}_{\texttt{safe}}$ to violate the ROI constraint, but, with high probability, the violation is bounded by at most 5%, 10%, 15% of λ^*, respectively. Instead, we do not introduce any tolerance for the daily budget constraint y_t, as it happens in practice.

In Figure 6.2, we show the median values, on 100 independent runs, of the performance in terms of daily revenue, ROI, and spend of $\mathsf{GCB}_{\texttt{safe}}$ for every value of ϵ_x.[9] The results show that, in practice, allowing a small tolerance in the ROI constraint violation, we can improve the exploration and, therefore, lead to faster convergence. We note that if we set a value of $\epsilon_x \leq 0.95$, we achieve significantly better performance in the first learning steps ($t < 20$) still maintaining a robust behavior in terms of constraints violation. Most importantly, a small tolerance leads only to a violation of the ROI constraint in the early learning stages, but the behavior at convergence is the same obtained without any tolerance.

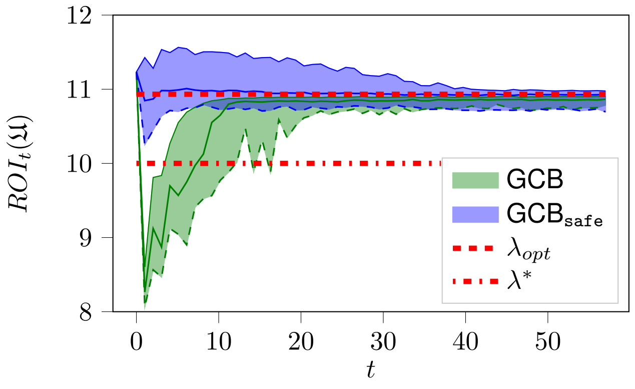
Experiment #3 We study a setting in which the ROI constraint is active at the optimal solution, *i.e.*, $\lambda^* = \lambda_{opt}$ and the budget constraint is not. This means that at the optimal solution, the advertiser would have an extra budget to spend, but she does not, otherwise, the ROI constraint would be

[9] Quantiles for these quantities have been omitted for visualization purposes. For the sake of completeness, they are provided in the Supplementary Material.

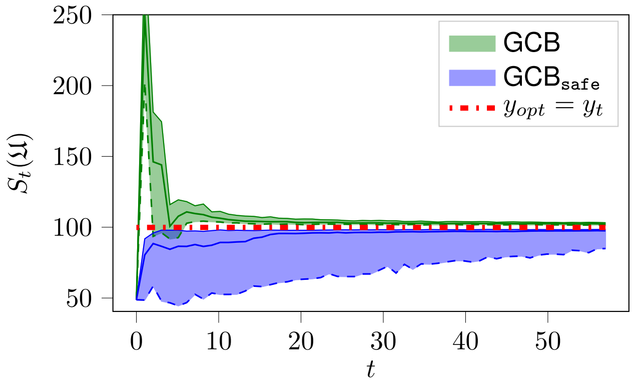
violated. The experimental setting is the same used in Experiments #1 and #2, except that we set a ROI constraint $\lambda^* = 10$ and a budget constraint $y(t) = 300$. The optimal daily spend is $y_{opt} = 161$. In Figure 6.3, we show the median values of the daily revenue, ROI, and spend of GCB, GCB$_{\texttt{safe}}$, GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$. We notice that also in this setting, GCB violates the ROI constraint for the entire time horizon, and the budget constraint in $t = 6$ and $t = 7$, however, it achieves revenue values larger than the optimum. Conversely, GCB$_{\texttt{safe}}$ always satisfies both the constraints, but it does not perform a sufficient exploration to converge quickly to the optimal solution. However, as also suggested by the results obtained in Experiment #2, it is sufficient to allow a tolerance in the ROI constraint violation by slightly perturbing the input value λ^* ($\epsilon_x = 0.95$, corresponding to a violation of the constraint by at most 5%) to make GCB$_{\texttt{safe}}$ capable of approaching the optimal solution while satisfying both constraints for every $t \in \{0, \ldots, T\}$. This suggests that, in real-world applications, GCB$_{\texttt{safe}}$ with a given tolerance may represent the best solution, providing guarantees on the violation of the constraints while returning empirically high values of revenue.



(a)

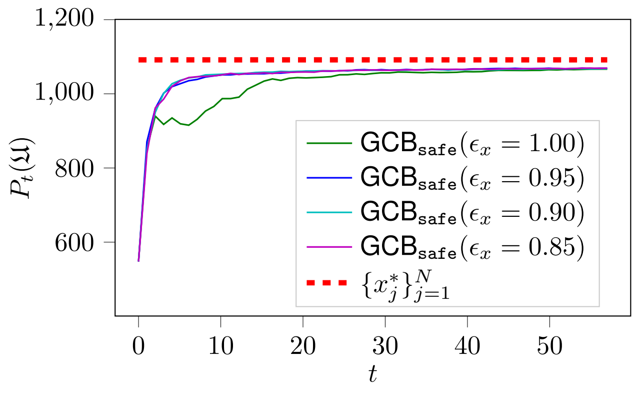


(b)

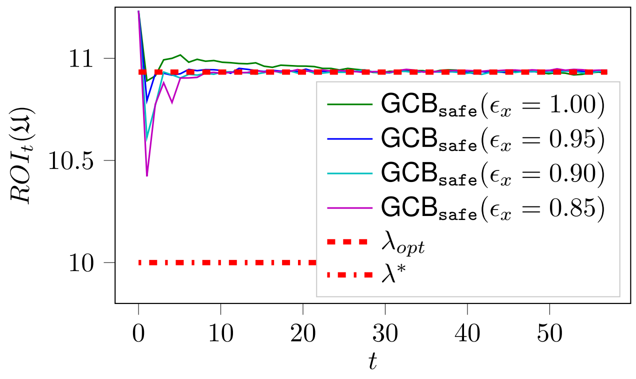


(c)

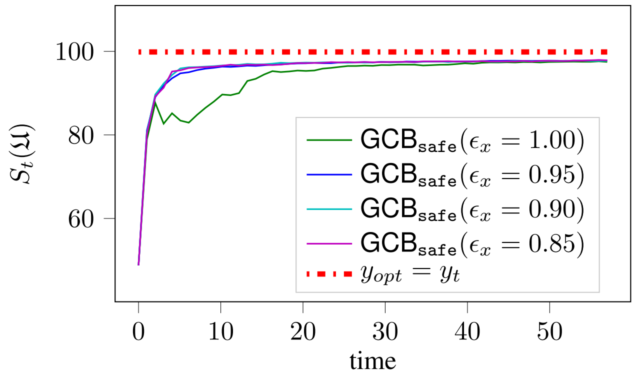
Figure 6.1: *Results of Experiment #1: daily revenue (a), ROI (b), and spend (c) obtained by GCB and GCB$_{\texttt{safe}}$. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*



(a)

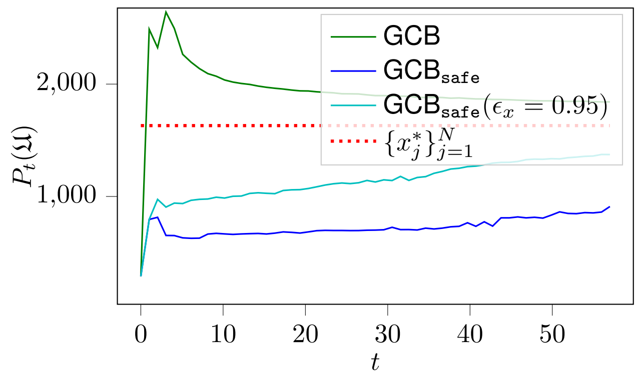


(b)

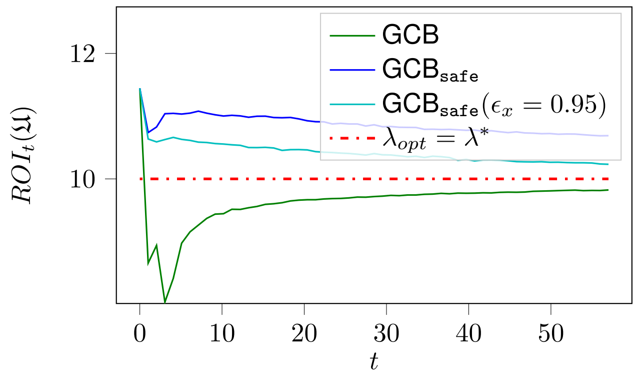


(c)

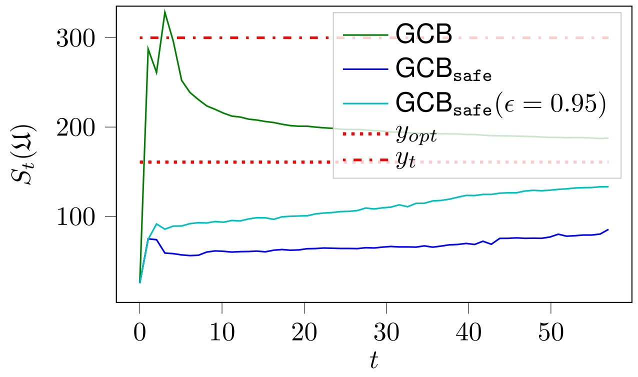
Figure 6.2: *Results of Experiment #2: Median values of the daily revenue (a), ROI (b) and spend (c) obtained by* $\mathsf{GCB}_{\mathtt{safe}}$ *with different values of* ϵ_x.



(a)



(b)



(c)

Figure 6.3: *Results of Experiment #3: Median values of the daily revenue (a), ROI (b) and spend (c) obtained by GCB, GCB*$_{\texttt{safe}}$*, and GCB*$_{\texttt{safe}}(\epsilon_x = 0.95)$*.*

CHAPTER 7

Off-line Targeting Optimization

A key to success of Internet advertising is the possibility of targeting very the ads to the users very accurately, thanks to a huge amount of data on the users' behavior available to the advertisement platforms [47]. However, such amount of data makes the problem of finding the best targeting unfeasible for humans, thus needing automatic methods. To this purpose, in this chapter, we develop an automatic method to find the best user targets that a media agency can adopt to boost the performance of an advertising campaign. While the creation of the ads is usually left to marketing experts, we focus on the joint task of selecting a suitable target for an advertising campaign and, at the same time, optimizing the bid/budget pairs of each sub-campaign. Such a problem is addressed in literature as Learning from Logged Bandit Feedback (LLBF). Nonetheless, our problem presents two peculiarities: first, the data about users, which are available to the media agencies, are aggregated (i.e., the behavior of a single user cannot be perfectly tracked); second, the problem of jointly optimizing bid and budget in a campaign is combinatorial. In the present work, we formulate the problem of target optimization as an LLBF problem, and we propose the TargOpt algorithm, which uses a tree expansion of the target space to learn the partition providing the maximum number of conversions efficiently. In doing so, we use a risk-averse approach. Furthermore, since the problem of finding the optimal target is intrinsically exponential in the number of the features, any algorithm may require exponential time. To cope with this issue, we propose a tree-search method, called A-TargOpt and two heuristics to drive the tree expansion, aiming at providing an anytime solution. Finally, we provide empirical evidence, on both synthetically generated and real-world data, that our algorithms provide an effective solution to find optimal targets for Internet advertising.

Problem Formulation

An Internet advertising campaign $\mathcal{C} := \{C_1, \dots, C_N\}$ is described by a set of N sub-campaigns C_j, each of which is identified by a tuple of K features $C_j := (z_{1j}, \dots, z_{Kj})$, *e.g.,* specifying the gender, age or the interests of the users we target by the sub-campaign C_j. Each feature $z_{ij} \subseteq Z_i$ is a nonempty set of values characterising the sub-campaign, where Z_i is the set of the feasible values for the i-th feature. For instance, if the i-th feature corresponds to the gender, with values M for male and F for female, we have $Z_i = \{M, F\}$ as the set of feasible values, and, thus, the corresponding feature can be $z_{ij} = \{M\}$ if the sub-campaign C_j targets only male users, $z_{ij} = \{F\}$ if it targets only the female ones, and $z_{ij} = \{M, F\}$ if it targets both. We assume that the sub-campaigns are targeting different sets of users. This implies that, for each pair of sub-campaigns in $\mathcal{C}$, these are disjoint, formally:

Definition 1. *Two sub-campaigns C_j and C_k are* disjoint *($C_j \cap C_k = \emptyset$) if it exists an index $i \in \{1, \dots, K\}$ s.t. $z_{ij} \cap z_{ik} = \emptyset$.*

To optimally set the target of the advertising campaign $\mathcal{C}$, we are provided with a set of logged bandit feedbacks generated by the application of an unknown decision policy $\mathfrak{U}$ over a time horizon of length T. At a generic round $t \in \{1, \dots, T\}$, the policy $\mathfrak{U}$ selects a bid/budget pair for each sub-campaign $C_j \in \mathcal{C}$.[1] Consider the following definitions:

Definition 2. *A sub-campaign C_j is called* atomic *if $|z_{ij}| = 1$ for each $i \in \{1, \dots K\}$, i.e., in which each feature has a single element.*

[1]We do not make any assumption on the policy $\mathfrak{U}$, except that it should provide feedback about all the possible bid/budget pairs and all the sub-campaigns $C_j \in \mathcal{C}$ we want to analyse for target optimization. For instance, a policy $\mathfrak{U}$ which never allocates budget on a specific sub-campaign $C_{\hat{j}}$ over the whole time horizon does not allow the optimization of the target for $C_{\hat{j}}$.

Definition 3. *Given two sub-campaigns C_j and C_k we say that C_j is included in or equal to C_k ($C_j \subseteq C_k$) if:*

$$z_{ij} \subseteq z_{ik} \quad \forall i \in \{1, \dots, K\}.$$

We assume to have, at each round t during which the policy $\mathfrak{U}_0$ runs, the following information on every atomic sub-campaign C_j such that there is $C_j \in \mathcal{C}$ with $C_j \subseteq C_j$:

- $\tilde{x}_t(C_j)$ is the bid which has been selected;
- $\tilde{c}_t(C_j)$ is the amount of budget spent;
- $\tilde{n}_t(C_j)$ is the number of clicks obtained;
- $\tilde{v}_t(C_j)$ is a cumulative revenue obtained by the conversions.

Finally, there exists a function $n(C_j, x, y)$ returning the average number of clicks for a generic sub-campaign C_j obtained when setting bid x and budget y and a parameter $v(C_j)$ denoting the average value per click of sub-campaign C_j.

The problem of jointly optimizing the values of the bid and daily budget for each sub-campaign of a given advertising campaign $\mathcal{C}$ has already been addressed in Chapter 5, where we cast such a problem as a MCK problem [29]. More formally, the problem aims at finding a bid/budget pair $(x(C_j), y(C_j))$ for each sub-campaign C_j such that:

$$\begin{aligned} J^*(\mathcal{C}) = & \max_{\{(x(C_j), y(C_j))\}_{i=1}^N} J(\mathcal{C}) \\ & \text{s.t.} \sum_{i=1}^N y(C_j) \leq \overline{B} \\ & \underline{x}(C_j) \leq x(C_j) \leq \overline{x}(C_j) \quad \forall i \in \{1, \dots N\} \\ & \underline{y}(C_j) \leq y(C_j) \leq \overline{y}(C_j) \quad \forall i \in \{1, \dots N\} \end{aligned}$$

where

$$J(\mathcal{C}) := \sum_{i=1}^N v(C_j)\, n(C_j, x(C_j), y(C_j))$$

is the revenue generated by the advertising campaign in a single day, $\overline{B}$ is the cumulative daily budget spent for all the sub-campaigns in a day, $\underline{x}(C_j)$ and $\overline{x}(C_j)$ are the minimum and maximum bid values available for the sub-campaign C_j, $\underline{y}(C_j)$ and $\overline{y}(C_j)$ are the minimum and maximum daily budget values that can be allocated to the sub-campaign C_j.

The optimization problem described above cannot control the structure of the campaign $\mathcal{C}$. This means that campaign $\mathcal{C}$ keeps to be unchanged during the whole time horizon. Conversely, in the present work, we face the problem of changing the configuration of the sub-campaigns to maximize the expected objective function $J^*(\mathcal{C})$. The space Ω in which we search for the optimal configuration of sub-campaigns is:

$$\Omega = \{\mathcal{C} = \{C_1, \dots, C_N\} \text{ s.t. } \forall i, j, i \neq j \quad C_j \cap C_j = \emptyset\},$$

i.e., the space of all the advertising campaigns whose sub-campaigns are disjoint. The optimization problem we study is:

$$\mathcal{C}^* := \arg\max_{\mathcal{C} \in \Omega} J^*(\mathcal{C}).$$

To fulfill this goal we will make use of the information over the time horizon T provided by the policy $\mathfrak{U}_0$.

Proposed Method: The TargOpt Algorithm

In this section, we describe an algorithm that, given a set of bids and budgets and a cumulative budget $\overline{B}$, finds an advertising campaign $\mathcal{C}^*$ which maximizes the expected revenue. This will be done by an exploration of the space Ω over a specifically designed tree and the use of a novel algorithm, called Target Optimization (TargOpt) able to work on it. In the case the space of all the possible feasible solutions is extremely large (due to a high number of features), a complete exploration of the space Ω is generally unfeasible. Therefore, we also provide a tree search algorithm, called Any-time Target Optimization (A-TargOpt), and effective heuristics able to efficiently visit the space Ω.

Number of Click Function Approximation

At first, we discretize of the bid/budget space. For the sake of concision and without loss of generality, we adopt the same discretization grid over the bid/budget space for all the atomic sub-campaigns C. Specifically, we have $\underline{x}(s) = \underline{x} > 0$, $\overline{x}(s) = \overline{x}$, $\underline{y}(s) = \underline{y} > 0$ and $\overline{y}(s) = \overline{B}$, and we use a uniform grid over the bid/budget space $X \times \overline{B}$ as follows:

$$X = \left\{ \underline{x} + \frac{h}{N_x}(\overline{x} - \underline{x}) \right\}_{h=0}^{N_x},$$

$$\overline{B} = \left\{ \underline{y} + \frac{h}{N_y}(\overline{y} - \underline{y}) \right\}_{h=0}^{N_y},$$

where $N_x \in \mathbb{N}^+$ and $N_y \in \mathbb{N}^+$ determine the granularity of the discretization for the bid and budget, respectively. Furthermore, we use lower bounds on the number of clicks in place of the empirical average value. Indeed, the maximization of the empirical average is not a suitable criterion to design a policy [49], since it might be arbitrarily far from the actual expected value. Instead, lower bounds take into account the uncertainty that affects the actual value, providing a risk-averse policies that minimize the probability of realizing a very low revenue or even a loss.

We compute the lower bounds $\underline{n}(C_j, x, y)$ and $\underline{v}(C_j)$ for the number of clicks $n(C_j, x, y)$ and the click values $v_i(C_j)$, respectively, with a given confidence δ, for each bid/budget pair in $X \times \overline{B}$ as follows. Let us focus on the number of clicks. Given a sub-campaign C_j, we compute a function $\underline{n}(C_j, x, y, \delta)$ that, for each element $(x, y) \in X \times \overline{B}$, returns a lower bound holding with probability δ on the number of clicks $n(C_j, x, y)$. This task is solved by the algorithm proposed in [41], where the estimation of the number of clicks is performed by means of Gaussian Processes [48]. More specifically, the number of clicks $n(C_j, x, y)$ corresponding to a specific bid/budget pair $(x, y) \in X \times \overline{B}$ is modeled as a Gaussian distribution, whose parameters, mean $\mu(C_j, x, y)$ and variance $\sigma^2(C_j, x, y)$, are estimated relying on historical observations $(\tilde{x}_t(C_j), \tilde{c}_t(C_j), \tilde{n}_t(C_j))_{t=1}^{T}$.[2] The lower bound on the number of clicks is computed as:

$$\underline{n}(C_j, x, y) := \hat{\mu}(C_j, x, y) - z_\delta \hat{\sigma}(C_j, x, y),$$

where $\hat{\mu}(C_j, x, y)$ is the estimates for the mean, $\hat{\sigma}(C_j, x, y)$ is the estimates of the standard deviation and z_δ the quantile of order δ of the standard Gaussian distribution. The same methodology can be applied to estimate a lower bound $\underline{v}(C_j)$ on the value $v(C_j)$ of sub-campaign C_j resorting to the sequence of samples $(v_t(c_i))_{t=1}^{T}$. See [41] for details. We underline that the procedure to obtain lower bounds we describe above can be substituted by any other suitable procedure. For instance, one can adopt a procedure using only historical data on a specific bid/budget pair (x, y) to estimate the lower bound $\underline{n}(C_j, x, y)$ employing, e.g., the Hoeffding bound [59].

Moreover, we remove the dependency of our optimization problem on the bid x by finding the best bid for every campaign C_j and every value of the daily budget y. We denote by $\underline{n}(C_j, y)$ the lower bound on the number of clicks of campaign C_j when the daily budget is y and the best bid is used, formally:

$$\underline{n}(C_j, y) := \max_{x \in X} \underline{n}(C_j, x, y).$$

[2] In this section, we summarize the procedure to estimate the number of clicks, and we refer to [41] for technical details.

In the next section, we denote the function returning the (lower bounds of the) revenue generated by a sub-campaign C_j with:

$$P(C_j, y) := \underline{v}(C_j)\,\underline{n}(C_j, y). \tag{7.1}$$

Tree Construction and Optimization

Let us define two operators working with campaigns and sub-campaigns used in what follows.

Definition 4. *Given a campaign $\mathcal{C}_i \in \mathcal{M}$ and a sub-campaign C_j we say that $\mathcal{C}_i$ is included in C_j ($\mathcal{C}_i \subseteq C_j$) if:*

$$\forall C_k \in \mathcal{C}_i \quad C_k \subseteq C_j.$$

Definition 5. *Given a sub-campaign C_j and a feature index $f \in \{1, \dots, K\}$, the partition operator $D := d(C_j, f)$ returns the set D of all the possible campaigns that can be generated by partitioning the sub-campaign C_j w.r.t. the feature x_{fj}. Formally, $D := \{\mathcal{C}_1, \dots, \mathcal{C}_{part(x_{fj})}\}$ s.t. each $\mathcal{C}_i \subseteq C_j$ and $\forall C_k \in \mathcal{C}_i$ $x_{hi} = x_{hj}, \forall h \neq f$, where $part(x_{fj})$ is the number of partitions of the set x_{fj}.*

For instance, given a sub-campaign $C_j = \{\{M, F\}, \{Y, A\}\}$—where the features are gender (M for male, F for female) and age (Y for young, A for adult)—and index $f = 1$, the partition operator $d(C_j, f)$ returns $D = \{\{C_1, C_2\}, \{C_3\}\}$, where $C_1 = (\{M\}, \{Y, A\})$, $C_2 = (\{F\}, \{Y, A\})$, and $C_3 = (\{M, F\}, \{Y, A\})$. This means that D is composed of two campaigns, the first composed, in its turn, of two sub-campaigns, the second composed of a single sub-campaign.

We use the tree $\mathcal{T} := (\mathcal{E}, \mathcal{O})$, composed of two different sets of nodes: an even level nodes set $\mathcal{E} := \{E_i\}$, in which each node E_i corresponds to a different campaign, and an odd level nodes set $\mathcal{O} := \{O_j\}$, in which each node O_j corresponds to a sub-campaign which is a child to some even node E_i. Each level of the tree corresponds to a single feature. The even levels nodes $E_i := (\mathcal{C}_i, Child_i, f_i, \boldsymbol{J}_i, Arg_i)$ and odd levels nodes $O_j := (C_j, Child_j, f_j, \boldsymbol{J}_j, Arg_j)$ are defined as tuples in which:

- $\mathcal{C}_i$ is a campaign;
- C_j is a sub-campaign;
- $Child_i$ and $Child_j$ are the sets of the children nodes;
- $f_i, f_j \in \{0, \dots, K\}$ are feature indexes indicating the level of the node and, at the same time, which feature has been selected;

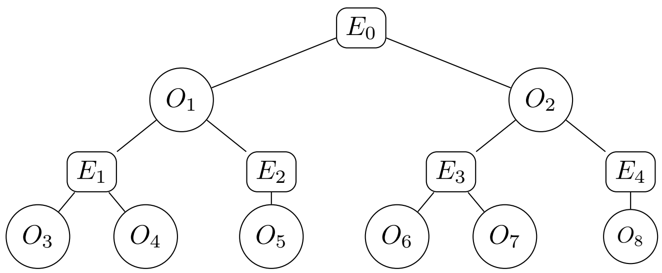


Figure 7.1: *An example of a tree representing a completely expanded advertising campaign, in which the original campaign $\mathcal{C}$ had two subcampaigns $C_1 = (\{M\}, \{Y, A\})$ and $C_2 = (\{F\}, \{Y, A\})$, where the subcampaigns are defined by gender (M for Male, F for Female) and age (Y for Young, A for Adult).*

- $\boldsymbol{J}_i := (\boldsymbol{J}_i(y))_{y \in \overline{B}}, \boldsymbol{J}_j = (\boldsymbol{J}_j(y))_{y \in \overline{B}}$, is the vector of the lower bound of the revenues for each $y \in \overline{B}$ (initially empty, which is used in the optimization procedure);
- $Arg_i, Arg_j \in \mathcal{M}^{N_y}$ is a vector of campaigns (initially empty, which is used in the optimization procedure).

The root of the tree $\mathcal{T}$ is the node $E_0 = (\mathcal{C}, Child_0, 0, \boldsymbol{J}_0, Arg_0)$, where $\mathcal{C} := \{C_1, \ldots, C_N\}$ is the original advertising campaign, $C_j \in \mathcal{C}$ are its sub-campaigns, and the set of children nodes $Child_0$ is composed of odd nodes $O_j = (C_j, Child_j, 0, \boldsymbol{J}_j, Arg_j)$. Given a non-atomic sub-campaign C_j, the set of the children $Child_j$ of a generic odd node $O_j = (C_j, Child_j, f_j, \boldsymbol{J}_j, Arg_j)$ is composed of even nodes $E_i = (\mathcal{C}_i, Child_i, f_j + 1, \boldsymbol{J}_i, Arg_i)$ s.t. the campaign $\mathcal{C}_i$ is in $D = d(C_j, f_j)$. Instead, if C_j is atomic, $Child_j$ is empty, meaning that O_j is a leaf. Given a campaign $\mathcal{C}_i$, the set of the children $Child_i$ of a generic even node $E_i = (\mathcal{C}_i, Child_i, f_i,$
$\boldsymbol{J}_i, Arg_i)$ is composed of odd nodes $O_j = (C_j, Child_j, f_i, \boldsymbol{J}_j, Arg_j)$ in which C_j is one of the sub-campaigns contained in $\mathcal{C}_i$. The construction of the tree $\mathcal{T}$ consists in the successive expansion of the root node E_0 until no odd node can be expanded further.

In Figure 7.1, we show an example of a fully expanded tree $\mathcal{T}$, generated starting from the campaign $\mathcal{C} = \{C_1, C_2\}$, with $C_1 = (\{M\}, \{Y, A\})$ and $C_2 = (\{F\}, \{Y, A\})$. The root of the tree E_0 (even rectangular node) corresponds to the original campaign $\mathcal{C}$, and its children nodes O_1 and O_2 (circular odd nodes) corresponds to the sub-campaigns C_1 and C_2, respectively. The next level contains the nodes representing all the possible campaigns and sub-campaigns that can be generated by partitioning the target of the sub-campaigns C_1 for O_1 and C_2 for O_2. More specifically, nodes O_3 and O_4 correspond to sub-campaign $C_3 = (\{M\},\{Y\})$ and $C_4 = (\{M\},\{A\})$, respectively, node E_1 corresponds to campaign $\mathcal{C}_1 = \{C_3, C_4\}$; node O_5 corresponds to sub-campaign $C_5 = C_1$ and E_2 corresponds to campaign $\mathcal{C}_2 = \{C_5\}$; nodes O_6 and O_7 correspond to sub-campaign $C_6 = (\{F\},\{Y\})$ and $C_7 = (\{F\},\{A\})$, respectively and node E_3 corresponds to campaign $\mathcal{C}_3 = \{C_6, C_7\}$; node O_8 corresponds to sub-campaign $C_8 = C_2$ and E_4 corresponds to campaign $\mathcal{C}_4 = \{C_8\}$.

Algorithm 4 TargOpt Algorithm

```
1: **Input:** tree $\mathcal{T} = (\mathcal{E}, \mathcal{O})$
2: **Output:** campaign $\mathcal{C}^*$
3: $l \leftarrow K$
4: **while** $l \geq 0$ **do**
5:     **for all** $O_j \in \mathcal{O} \mid f_j = l$ **do**
6:         $\boldsymbol{J}_{new}, Arg_{new} \leftarrow$ OddNodeUpdate$(O_j)$
7:         $O_j \leftarrow (C_j, Child_j, f_j, \boldsymbol{J}_{new}, Arg_{new})$
8:     **for all** $E_i \in \mathcal{E} \mid f_i = l$ **do**
9:         $\boldsymbol{J}'_{new}, Arg'_{new} \leftarrow$ EvenNodeUpdate$(E_i)$
10:        $E_i \leftarrow (\mathcal{C}_i, Child_i, f_i, \boldsymbol{J}'_{new}, Arg'_{new})$
11:    $l \leftarrow l - 1$
12: $\mathcal{C}^* \leftarrow Arg_0(\overline{B})$
13: **return** $\mathcal{C}^*$
```

We describe TargOpt algorithm to find the campaign maximizing the revenue. The algorithm traverses the tree from the leaves to the root and computes the lower bound of the revenue for each campaign in $\mathcal{T}$. The pseudocode of the TargOpt algorithm is presented in Algorithm 4. It takes as input a tree $\mathcal{T} = (\mathcal{E}, \mathcal{O})$ and it returns the optimal campaign $\mathcal{C}^*$. The algorithm starts from the lowermost level of the tree ($l = K$) and applies the procedure OddNodeUpdate(O_j), detailed in what follows, for all the odd nodes O_j s.t. $f_j = l$ (Line 6). This procedure fills the vector $\boldsymbol{J}_{new}$ with the revenues corresponding to each budget $y \in \overline{B}$ and the corresponding campaign Arg_{new}, i.e., a vector containing campaigns which provides the revenues in $\boldsymbol{J}_{new}$. The nodes are then updated to include this new information (Line 7). After that, these results are used to update the even nodes E_i s.t. $f_i = l$ with the subroutine EvenNodeUpdate(E_i) (Line 9), detailed in what follows. The subroutine aggregates the revenues $\boldsymbol{J}_j$ provided by the children $O_j \in Child_i$, executing a variation of the MCK solving algorithm. Indeed, it provides the vector of the lower bounds on the revenue $\boldsymbol{J}'_{new}$ and the vector of the campaigns Arg'_{new} corresponding to those revenues for each budget $y \in \overline{B}$. After that, the algorithm updates each analysed even node E_j (Line 10). The procedure moves to the upper level ($l \leftarrow l - 1$) and iterates until it reaches the root ($l = 0$), where it returns the campaign $Arg_0(\overline{B})$, contained in the root node E_0, corresponding to the optimal revenue $\boldsymbol{J}_0(\overline{\overline{B}})$ given a total budget of $\overline{B}$.

Algorithm 5 EvenNodeUpdate Subroutine

```
1:  Input: even node $E_i$
2:  Output: vector of the lower bounds of the revenues $\boldsymbol{J}_{new}$, vector of the campaigns $Arg_{new}$
3:  $\boldsymbol{J}_{new} \leftarrow \mathbf{0}$
4:  for $y \in \overline{B}$ do
5:      $Arg_{new}(y) \leftarrow \emptyset$
6:  for $O_j \in Child_i$ do
7:      $\boldsymbol{J}_{old} \leftarrow \boldsymbol{J}_{new}$
8:      $Arg_{old} \leftarrow Arg_{new}$
9:      for $y \in \overline{B}$ do
10:         $y^* = \arg \max_{y' \in \overline{B}, y' \leq y} [\boldsymbol{J}_{old}(y') + \boldsymbol{J}_j(y - y')]$
11:         $\boldsymbol{J}_{new}(y) = \boldsymbol{J}_{old}(y^*) + \boldsymbol{J}_j(y - y^*)]$
12:         if $y^* = 0$ then
13:             $Arg_{new}(y) = Arg_j(y)$
14:         else
15:             if $y^* = y$ then
16:                 $Arg_{new}(y) = Arg_{old}(y)$
17:             else
18:                 $Arg_{new}(y) \leftarrow Arg_{old}(y) \cup Arg_j(y)$
19: return $\boldsymbol{J}_{new}, Arg_{new}$
```

The pseudocode of the subroutine EvenNodeUpdate(E_i) is presented in Algorithm 5. It is a variation of the dynamic-programming algorithm used to solve the MCK problem [29] specifically crafted for our scenario. Given an even node E_i it computes the vector of the optimal lower bound of the revenue $\boldsymbol{J}_{new}$ corresponding to each budget $y \in \overline{B}$ and the corresponding vector of campaigns Arg_{new}. At first, it sets null revenue and empty optimal campaign for all the budgets $y \in \overline{B}$ (Lines 3-5). Then, it analyses each odd node O_j in the set of children $Child_i$ and evaluates for each budget $y \in \overline{B}$ whether the lower bound of the revenue $\boldsymbol{J}_j(y)$ provided by the campaign C_j in the child O_j is better than any other one computed so far, or whether there exists an allocation of budgets over more than one sub-campaign that can perform better (Line 10). Then, it stores in $Arg_{new}(y)$ the set of sub-campaigns which is providing the largest lower bound on the revenue $\boldsymbol{J}_j(y)$ for the budget y (Lines 12-18). Once this process is repeated for all the nodes $O_j \in Child_i$, the subroutine returns the vector of the largest lower bound on the revenue $\boldsymbol{J}_{new}$ and the vector of the corresponding campaigns Arg_{new} (Line 19).

Algorithm 6 OddNodeUpdate Subroutine

```
1: Input: odd node $O_j$
2: Output: vector of the lower bounds of the revenues $\boldsymbol{J}_{new}$, vector of the campaigns $Arg_{new}$
3: if $Child_j = \emptyset$ then
4:     for $y \in \overline{B}$ do
5:         $\boldsymbol{J}_{new}(y) \leftarrow P(C_j, y)$
6:         $Arg_{new}(y) \leftarrow s_j$
7: else
8:     for $y \in \overline{B}$ do
9:         $i^* \leftarrow \arg\max_{i | E_i \in Child_j} \boldsymbol{J}_i(y)$
10:        $\boldsymbol{J}_{new}(y) \leftarrow \boldsymbol{J}_i^*(y)$
11:        $Arg_{new}(y) \leftarrow Arg_{i^*}(y)$
12: return $\boldsymbol{J}_{new}$, $Arg_{new}$
```

The subroutine OddNodeUpdate is provided in Algorithm 6. It requires as input an odd node O_j. If O_j does not have any child, the subroutine fills each element of the vector of the lower bounds of the revenue $\boldsymbol{J}_{new}(y)$ using the function $P(C_j, y)$ defined in Equation (8.4) (Lines 3-6). Conversely, if the set of the children $Child_j$ is not empty , the vector $\boldsymbol{J}_{new}$ is computed by choosing for each $y \in \overline{B}$ the maximum value of the revenue $\boldsymbol{J}_i(y)$ among the even nodes $E_i \in Child_j$ (Lines 8-11). The subroutine also stores the campaign vector Arg_{new}, whose elements are the one providing the revenues in the vector $\boldsymbol{J}_{new}$. At last, it returns the vector of the lower bound of the revenue $\boldsymbol{J}_{new}$ and the vector of the campaigns Arg_{new} (Line 12).

Approximated Algorithm for Large Feature Space

In the case the feature space is such that $K \gg 1$, the expansion of the tree $\mathcal{T}$ up to the atomic sub-campaigns might be computationally expensive, since the number of the odd nodes scales as $O(2^{|Z|K})$, where $|Z| := \min_i |Z_i|$ is the minimum cardinality of the sub-campaign features. To perform the target optimization also when the execution of the TargOpt algorithm is unfeasible, we design a variation of the TargOpt algorithm that, starting from a tree composed only by the root node E_0, iteratively expands the nodes in a classical tree-search fashion.

The pseudocode of our algorithm, called A-TargOpt, is presented in Algorithm 7. It takes as input a campaign $\mathcal{C}$ and a maximum number of odd

Algorithm 7 A-TargOpt Algorithm

```
 1: Input: campaign $\mathcal{C}$, maximum number of nodes to expand $N_{\max}$.
 2: Output: best campaign discovered $\mathcal{C}^*$
 3: $\mathcal{O} \leftarrow \{(C_i, \emptyset, 0, \emptyset, \emptyset)\}_{i=1}^{N}$
 4: $\mathcal{E} \leftarrow \{(\mathcal{C}, \{O_1, \dots O_N\}, 0, \emptyset, \emptyset)\}$
 5: $\mathcal{C}^* \leftarrow$ TargOpt$((\mathcal{E}, \mathcal{O}))$
 6: while $|\mathcal{O}| < N_{\max}$ do
 7:     $\mathcal{L} \leftarrow \{O_k \in \mathcal{O} \,|\, Child_k = \emptyset\}$
 8:     Select $O_i \leftarrow H(\mathcal{L})$
 9:     $D \leftarrow d(C_i, f_i + 1)$
10:     $Child_i \leftarrow \emptyset$
11:     for $\overline{\mathcal{C}} \in D$ do
12:         $\overline{Child} \leftarrow \emptyset$
13:         for $\hat{C} \in \overline{\mathcal{C}}$ do
14:             $\underline{O} \leftarrow (\hat{C}, \emptyset, f_i + 1, \emptyset, \emptyset)$
15:             $\overline{Child} \leftarrow \mathcal{O} \cup O$
16:         $\mathcal{O} \leftarrow \mathcal{O} \cup \overline{Child}$
17:         $E \leftarrow (\overline{\mathcal{C}}, \overline{Child}, f_j + 1, \emptyset, \emptyset)$
18:         $Child_i \leftarrow Child_i \cup E$
19:         $\mathcal{E} \leftarrow \mathcal{E} \cup E$
20:     $\mathcal{O} \leftarrow \mathcal{O} \setminus O_i$
21:     $O_i = (C_i, Child_i, f_i, \emptyset, \emptyset)$
22:     $\mathcal{O} \leftarrow \mathcal{O} \cup O_i$
23:     $\mathcal{C}^* =$ TargOpt$((\mathcal{E}, \mathcal{O}))$
24: return $\mathcal{C}^*$
```

nodes $N_{\max} \in \mathbb{N}$ to expand.[3] At first, the algorithm initializes the tree with an odd node for each sub-campaign in $\mathcal{C}$ (Line 3) and a single even node E_0 for the original campaign (Line 4). In the case the expansion is already too computationally expensive for the available budget ($|\mathcal{O}| > N_{\max}$), it executes the TargOpt algorithm on $\mathcal{T}$ to provide a tentative solution $\mathcal{C}^*$ (Line 5). At each iteration, it expands the more promising nodes according to a strategy function $H(\mathcal{L})$, which takes a set of odd leaf nodes $\mathcal{L}$, i.e., the odd nodes $O_k \in \mathcal{O}$ s.t. $Child_k = \emptyset$, and returns a single odd node $O_i \in \mathcal{L}$ which will be expanded. Here, we propose two different strategies $H(\cdot)$ we adopt to select the more promising node to be expanded:

- Breadth-First Search (BFS), which in our specific setting consists in expanding one of the nodes with the largest target, i.e., $O_i \in \mathcal{L}$ s.t. $\nexists\, O_k \in \mathcal{L}$ s.t. $f_k < f_i$;
- Optimistic Search (OS), which expands the nodes with the maximum

[3]We limit the number of the odd nodes since they are the most computationally expensive, being those that execute the algorithm solving the MCK problem.

average revenue or, formally, $O_i \in \mathcal{L}$ s.t. $i = \arg\max_h \frac{\sum_{y \in \overline{B}} \boldsymbol{J}_h(y)}{N_y}$.

Once an odd node O_i has been chosen (Line 8), the algorithm uses the partition operator $D = d(C_j, f_i)$ to expand the sub-campaign C_j (Line 9). Moreover, it generates and adds in $Child_j$ all the even nodes $E = (\overline{\mathcal{C}}, \overline{Child}, f_j + 1, \emptyset, \emptyset)$ corresponding to campaigns in D (Lines 17-18) and it adds to the child set $\overline{Child}$ each node E all the odd nodes O corresponding to the sub-campaign $\hat{s} \in \bar{\mathcal{C}}$ (Lines 13-15). Finally, the algorithm adds to the tree $\mathcal{T}$ the newly generated node and its children (Lines 19-22), and executes the TargOpt algorithm on the updated version of the tree. These operations are repeated until the tree $\mathcal{T}$ has at most $N_{\max}$ odd nodes.

Experimental Evaluation

We experimentally evaluate our algorithm on both real-world and synthetic problems. At first, we show the improvement provided by the TargOpt algorithm on a real-world advertising problem and, after that, we evaluate the performance of the A-TargOpt algorithm in a synthetically generated environment.

Evaluation in a Real-world Setting

Experimental Setting. We evaluate the TargOpt algorithm on a real-world dataset, provided by the company MediaMatic [39]. The database corresponds to an advertising campaign for a financial product, whose name is omitted for privacy reasons.[4] The original campaign $\mathcal{C}$ is composed of $N = 8$ sub-campaigns. The dataset we use for the experiments consists of the data recorded from $01/09/17$ to $08/11/17$, for a total of $T = 69$ days. The (single) feature z_{1j} that we analyze in this experiment is the hour of the day the ad is displayed to users. More specifically, for this feature, we consider a set of $|Z_1| = 8$ values corresponding to 3-hours-long time slots (0.00 a.m.-3.00 a.m., 3.01 a.m.-6.00 a.m., etc.). A finer granularity would make the optimization problem intractable. Since a preliminary analysis suggests that the behaviour of users is different during weekdays and weekends, e.g., the total volumes of the search are significantly different between the twos, we separate the gathered data to apply the TargOpt algorithm independently to the two scenarios. The policy used for the collection of the data is the AdComb algorithm [41], where we set a budget $\overline{B} = 1100$, while we discretize the bid space evenly in $(\underline{x}, \overline{x}) = (0.1, 5)$

[4] To fulfill the NDA we have with the media agency, some of the values reported in what follows of the experiment have been scaled, given that the transformation we applied does not change the conclusion we draw.

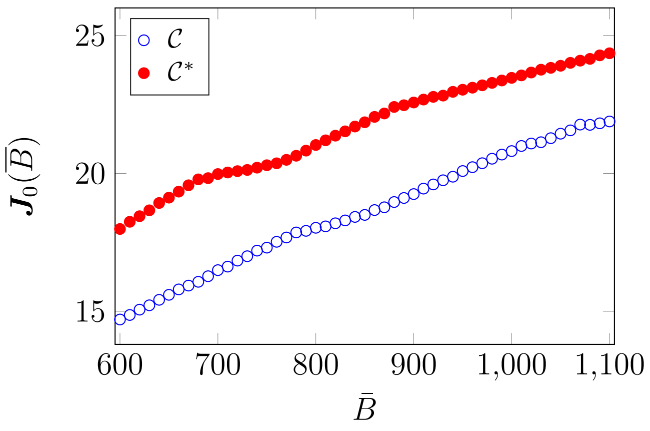


Figure 7.2: *Revenue of the* TargOpt *on the real-world dataset for the weekdays.*

with $|X| = 50$ values, and the budget space evenly in $(\underline{y}, \overline{y}) = (2, 1100)$ with $|\overline{B}| = 550$ possible budgets. We use the same discretization of the bid/budget space also while executing the TargOpt algorithm. We compare the optimal campaign $\mathcal{C}^*$, resulting from the execution of the TargOpt algorithm, with the one provided by the original campaign $\mathcal{C}$. The performance index we use to evaluate the performance of the two campaigns is the lower bound on the revenue $\boldsymbol{J}_0(\overline{\overline{B}})$, where we assume a unitary value for the conversion.

Results. The results of the experiments are presented in Figure 7.2 for the weekdays and Figure 7.3 for the weekends. In both the cases, the TargOpt algorithm provides a significant improvement over the original campaign $\mathcal{C}^*$, i.e., about 13% more conversions during the weekdays and about 30% during the weekend. This improvement does not reduce as the budget invested per day $\overline{B}$ increases. In the weekend scenario, the revenue for campaign $\mathcal{C}$ is almost constant for $\overline{B} \geq 230$ ($\boldsymbol{J}_0(\overline{B}) \approx 9$). This phenomenon is due to the fact that, as we increase the total daily budget, part of what is spent is targeting sub-campaigns in which no conversion occurs, therefore even by increasing the budget we do not have any further conversion. This issue is partially overcome with the use of a more fine-grained targeting provided by the campaign $\mathcal{C}^*$, whose performance are bounded to $\boldsymbol{J}_0(\overline{B}) \approx 12$ for $\overline{B} \geq 260$. We suppose that an even finer-grained targeting, e.g., using 1-hour-long slots, could further improve the performance of the optimal advertising campaign.

Evaluation in a Synthetically Generated Setting

Experimental Setting. In this experiment, we compare the performance of the proposed exploration strategies for the A-TargOpt algorithm in a synthetically generated setting. The original campaign $\mathcal{C} = \{C_0\}$ is composed of a single sub-campaign C_0 having $K = 3$ features, each of which has cardinality 3. Given an advertising period of $T = 100$ days, we generate for each day $t \in \{1, \dots, T\}$ and for each atomic sub-campaign $C_j \subseteq C_0$, the daily observations about the selected bid $\tilde{x}_t(C_j)$, selected budget $\tilde{y}_t(C_j)$, number of clicks $\tilde{n}_t(C_j)$ and revenue per click $\tilde{v}_t(C_j)$, in the following way. For each atomic campaign C_j, the policy $\mathfrak{U}$ selects a budget $y_t(C_j)$ uniformly over $\overline{B} = \{0, \dots, 100\}$, with $|\overline{B}| = 10$, and keeps the bid $x_t(C_j) = \bar{c}$ constant during the period and the sub-campaigns. This provides an average number of 10 observations for each sub-campaign and each budget. For each sub-campaign C_j and for each day t, the daily num-

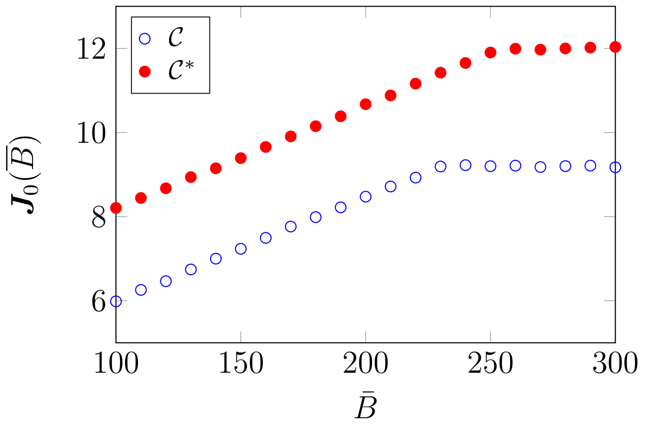


Figure 7.3: *Revenue of the TargOpt on the real-world dataset for the weekends.*

ber of clicks is computed as $\tilde{n}_t(C_j) := \frac{\tilde{c}_t(C_j)}{cpc_t(C_j)}$, where the cost per click $cpc_t(C_j)$ is extracted from $\mathcal{N}(0.5, 0.1)$ and $\mathcal{N}(\mu, \sigma)$ is the Gaussian distribution with mean μ and standard deviation σ. We generate the cumulative value obtained by the conversions $v_t(C_j)$ as $v_t(C_j) := cr(C_j)\, n_t(C_j)$ where the conversion rate $cr(C_j)$ for an atomic sub-campaign C_j is extracted from $0.5\, B(0.5)$, being $B(\mu)$ the Bernoulli distribution of parameter μ. This cumulative value modeling exemplifies the case in which, on average, half of the atomic sub-campaigns is not profitable at all. From the data corresponding to each atomic sub-campaign C_j we estimate the lower bound on the number of clicks $\underline{n}(C_i, y)$ as described in Section 7.2.1. Moreover, the lower bound on the number of clicks $\underline{n}(C_j, y)$ corresponding for the non-atomic sub-campaign C_j is computed by aggregating the observations of

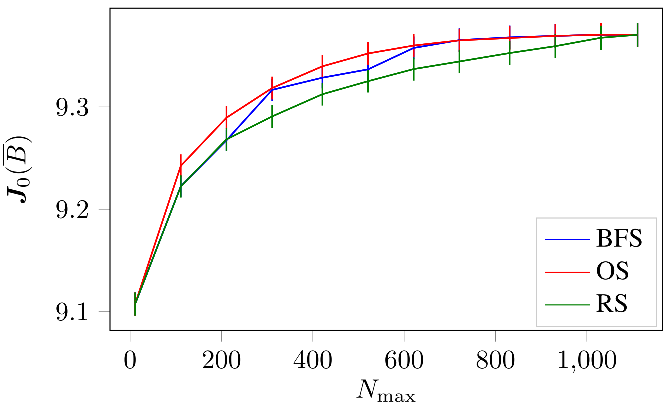


Figure 7.4: *Revenue obtained by the BFS, OP and RS heuristics by setting a different number of maximum nodes to expand $N_{\max}$. 95% confidence interval are represented as error bars.*

those atomic sub-campaigns s.t. $C_j \subseteq C_i$ and computing the lower bound as in Section 7.2.1.[5]

In the execution of the A-TargOpt algorithm, we set a total budget to spend of $\overline{B} = 100$ and a number of budget $N_y = 10$. As for performance index, we use the lower bound of the revenue $\boldsymbol{J}_0(\overline{B})$ obtained at the end of the procedure. We compare the performance of our two heuristics with a baseline, called Random Strategy (RS), which expands the node by selecting at random from the leaf nodes. We average the results over 200 independent runs.

Results. The results of the synthetically generated setting are provided in Figure 7.4. It is possible to see that for $200 \leq N_{\max} \leq 700$ there is statistical evidence that the BFS and OS heuristics are performing better than the RS one. Conversely, with a few nodes ($N_{\max} < 200$) or when the tree is almost completely expanded ($N_{\max} > 700$) there is no significant difference among the performance of the three heuristics. This suggests that our two heuristics provide an advantage when the problem allows the expansion of a considerable portion of the tree, while if we explore only a few nodes, a random exploration might be valid as well.

In general, looking at the average performance, one should follow the OS heuristic when expanding the nodes, since it is always providing the largest revenue on average. Nonetheless, the heuristic BFS provides solutions which are more compact than OS since the BFS heuristic builds the sub-campaigns trying to keep their targets as the large as possible. There-

[5]The aggregation of the data as performed in this experiment is correct under the assumption that all the atomic sub-campaigns C_j interacted with the same number of users, thus they contributes to the data of sub-campaigns C_j in the same way.

fore, if the marketing experts require a limited number of sub-campaigns, e.g., to perform further business analysis on the advertising campaign, the BFS heuristic is to be preferred, while, if we are more concerned about the revenue, we should rely on the OS heuristic when expanding the tree $\mathcal{T}$.

CHAPTER 8

Dealing with Interdependencies in Online Advertising Campaigns Optimization

The sub-campaigns interdependence is customarily exploited by experts in the field, e.g., setting up sub-campaigns (called *assist*) not providing direct conversions but increasing the number of conversions on the search engine channel. Besides, capturing the interdependence can provide a direct method for comparing and optimizing the performance of sub-campaigns on different channels. Indeed, sub-campaigns on different channels need to be evaluated using different performance metrics, and how to combine them is still an open issue. For instance, display and social ads provide very few conversions compared to search ads but allow search ads to generate a larger number of conversions, and therefore an optimization method based only on the number of conversions might not provide optimal allocations. The same holds in the search channel for branding and no-branding sub-campaigns. Although this problem is central in advertising, to the best of our knowledge, no model in the economic literature captures such interdependence.

We design an algorithm based on both learning and optimization techniques that can be adopted for the optimization of real-world Internet advertising campaigns and that, exploiting sub-campaigns interdependence, outperforms the algorithms known so far. To do that, we provide a novel model that, on one side, is expressive enough to capture the interdependences and, on the other side, is simple enough to require few data for its estimation.[1] From of the proposed model, we design a data-driven algorithm, called IDL, which consists of two phases: the *Interdependence Graph Learning Phase* and the *Estimation and Optimization Phase*. In the former phase, the IDL algorithm learns the sub-campaigns interdependence structure (represented as a graph), identifying the pairs of sub-campaigns with the most significant interdependences by applying the Granger Causality test. This is crucial since the number of pairs of interdependent sub-campaigns dramatically increases the amount of data required to have accurate estimates of the model parameters. In the latter phase, the IDL algorithm computes the optimal joint bid/daily budget allocation exploiting Gaussian Process modeling [48] and an *ad hoc* dynamic programming procedure. In particular, Gaussian Processes exploit some form of functional regularity to describe the relationships among the problem parameters, without forcing them belong to a specific family of curves.

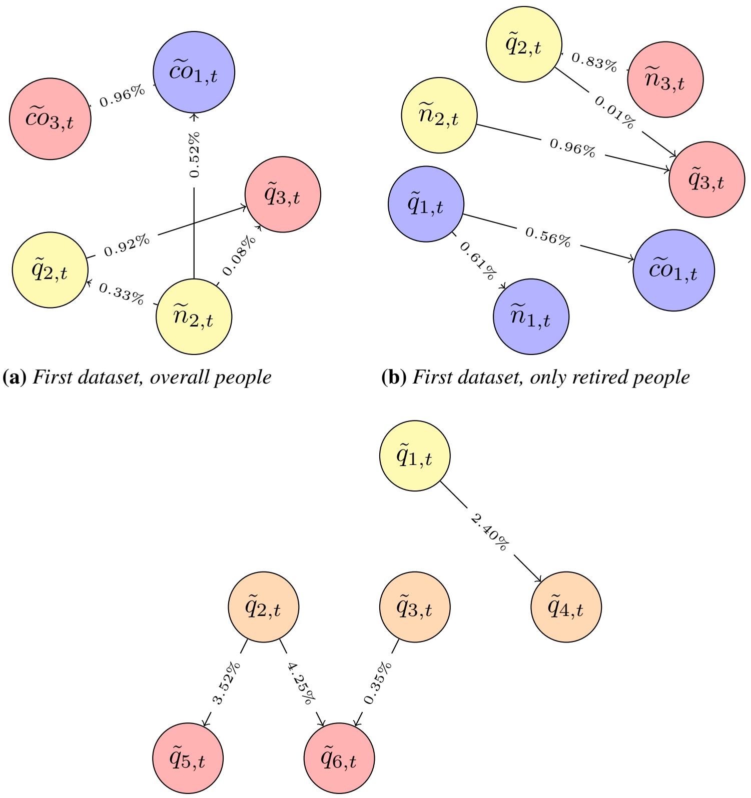
Finally, we show that neglecting the sub-campaigns interdependence can lead to massive losses even in simple and common scenarios and we theoretically bound the loss of our algorithm. Furthermore, we experimentally evaluate its performance in both realistic and real-world settings, showing the superiority of its performance compared to the previous approaches that neglect the sub-campaigns interdependence.

Problem Formulation

Model

Assume to have an Internet advertising campaign $\mathcal{C} = \{C_1, \dots, C_N\}$, with $N \in \mathbb{N}$, where C_j is the j-th sub-campaign. At day t, we are asked to set for each sub-campaign C_j a bid $x_{j,t} \in [\underline{x}_j, \overline{x}_j]$, and a daily budget $y_{j,t} \in [\underline{y}_j, \overline{y}_j]$, subject to that the daily cumulative budget of all the sub-campaigns cannot exceed $\overline{B} \in \mathbb{R}^+$. At day $t+1$, we get a report on the performance of the campaign $\mathcal{C}$ at the previous day t, which specifies, for every C_j, the tuple $(\widetilde{q}_{j,t}, \widetilde{n}_{j,t}, \widetilde{co}_{j,t}, \widetilde{c}_{j,t})$, where $\widetilde{q}_{j,t}$ denotes the number of impressions, $\widetilde{n}_{j,t}$ denotes the number of received clicks, $\widetilde{co}_{j,t}$ denotes the cumulate value of

[1]Due to learning, a sophisticated model may provide poor performance. This is because an excessively complex model could need a large amount of data for the training and collecting such data could require a long time, even longer than the time horizon considered in the optimization process.



(a) *First dataset, overall people* **(b)** *First dataset, only retired people*

(c) *Second dataset*

Figure 8.1: *Graphs representing the interdependences of real-world Internet advertising sub-campaigns inferred by Granger Causality test from real data. The numbers on the edges are the p-values (in terms of $\%$) of the Granger Causality test; display ads are depicted in blue, social ads in yellow, and search ads in orange (for branding sub-campaigns) and red (for other search sub-campaigns). The second dataset graph refers to interdependence among impressions $\tilde{n}_{j,t}$ of different sub-campaigns C_j.*

the conversions, and $\widetilde{c}_{j,t}$ denotes the amount of money spent for it.[2]

As aforementioned, both experts in the field of Internet advertising and studies in the Internet economic field, *e.g.,* [30] and [25], demonstrate that impressions, clicks, and conversions of a sub-campaign might be influenced by the same kind of quantities of the other sub-campaigns. We extend the previous studies on the sub-campaigns interdependence, applying the Granger Causality test [24, 55] to two real-world Internet advertising campaigns optimized by an Italian web media agency using the AdComb-TS

[2]We recall that the money spent in one day for a sub-campaign may be different from the daily budget previously allocated.

algorithm (see Chapter 5). The algorithm, being online, produces policies explorative enough to make the test significant.

At first, we test for Granger causality the data collected for 8 months (from $1/1/2018$ to $1/8/2018$) from an Internet advertising campaign for a financial service of an insurance company: data correspond to $N = 12$ sub-campaigns, on Google AdWords (search), Facebook (social), and Google display with a cumulative budget of $Y = 600$ Euros. The results obtained from the Granger Causality test are shown in Figure 8.1, where the most significant elements of $(\widetilde{q}_{j,t}, \widetilde{n}_{j,t}, \widetilde{co}_{j,t}, \widetilde{c}_{j,t})$ are represented as nodes of different colors according to their specific channel (as detailed in the caption of the figure) and the detected interdependences (with a p-value less than 5%) are represented as directed edges. In particular, Figure 8.1a shows the results when all the sub-campaigns data are aggregated, while Figure 8.1b focuses on a specific subset of sub-campaigns who share the same targeting (retired people). These results confirm the presence of the interdependence between display and search advertising as previously observed in the literature. They also show that social and search advertising are interdependent and that the interdependences may be targeting specific. Moreover, the interdependence between clicks and impressions of the social channels and the impressions of the search one in this specific scenario seems to be more relevant than others, since they appear in both graphs. Furthermore, the Granger Causality test detects that interdependence dynamics between sub-campaigns are delayed up to 2 days. At second, we test for Granger causality the data collected for 3 months (from $20/7/2018$ to $20/10/2018$) from an Internet advertising campaign of a different financial product of the same company with about $Y = 1100$ Euros. There are $N = 14$ sub-campaigns belonging to social and search advertising channels. The resulting graph is depicted in Figure 8.1c (with a p-value less than 5%). As in the previous dataset, many sub-campaigns are subject to interdependence. In particular, in this case, the interdependence phenomenon is only among impressions, suggesting that these can be the most significant in practice. Moreover, differently from the previous case, we distinguish search sub-campaigns into two subclasses which are at different depths in the marketing funnel: branding (orange nodes) or no-branding (red nodes). Finally, the delay of the interdependence dynamics is up to 3.

Optimization Problem

We provide our optimization problem capturing the sub-campaigns interdependence. For the sake of presentation, we focus on the interdependence between the impressions of different sub-campaigns.[3] Our goal is the maximization of the revenue earned each day from an Internet advertising campaign subject to a cumulative budget constraint. Formally, given a campaign $\mathcal{C}$ and a cumulative daily budget of $\overline{B}$, we aim to find, at day t, the value of bid $x_{j,t}$ and the value of daily budget $y_{j,t}$ for every sub-campaign C_j that maximise the revenue by solving the following optimization problem:

$$\max_{x_{j,t}, y_{j,t}} \sum_{j=1}^{N} v_j \; r_j \; q_j(x_{j,t}, y_{j,t}, u_{j,t}) \tag{8.1a}$$

$$\text{s.t.} \sum_{j=1}^{N} y_{j,t} \leq \overline{B} \tag{8.1b}$$

$$\underline{x}_j \leq x_{j,t} \leq \overline{x}_j \qquad \forall j \tag{8.1c}$$

$$\underline{y}_j \leq y_{j,t} \leq \overline{y}_j \qquad \forall j \tag{8.1d}$$

where $q_j(x_{j,t}, y_{j,t}, u_{j,t})$ is the expected number of impressions given bid $x_{j,t}$, daily budget $y_{j,t}$, and *influence index* $u_{j,t}$, representing the influence of other sub-campaigns towards C_j and computed by using the number of impressions of those sub-campaigns that are interdependent with sub-campaign C_j (see below); r_j and v_j are the click-trough rate and the value per click for the sub-campaign C_j, respectively, and, therefore, $v_j \; r_j \; q_j(x_{j,t}, y_{j,t}, u_{j,t})$ is the revenue provided by sub-campaign C_j.[4] We denote with $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{u}^*)$ the optimal solution to the optimization problem.

To model the sub-campaigns interdependence, we define:

Definition 6. *Given an advertising campaign $\mathcal{C}$, an* interdependence graph *$\mathcal{G} := (\mathcal{C}, D)$ is a graph in which the adjacency matrix $D = \{d_{ij}\}, D \in \{0,1\}^{N \times N}$ has elements $d_{ij} = 1$ iff the sub-campaign C_i influences the performance of the sub-campaign C_j.*

We assume that the graph $\mathcal{G}$ is a Directed Acyclic Graph (DAG), i.e., there are no dependency cycles among the sub-campaigns. This assumption is supported by the model of the marketing funnel, in which the majority of the users flows from the top to the bottom, and different advertising channels are positioned at different levels of the funnel. Without loss of

[3]The use of such quantities is also supported by the experimental results of Section 8.1, where the interdependence among impressions is the most significant. However, different models, e.g., including the interdependence between the clicks and the conversions, are straightforward extensions of what is proposed in this section.

[4]The optimization problem in Equations (8.1a)–(8.1d) reduces to the one proposed in Chapter 5 when there is no interdependence, i.e., if $q_j(x_{j,t}, y_{j,t}, u_{j,t}) = q_j(x_{j,t}, y_{j,t})$, for every C_j.

generality, we assume that the order over the indices of the sub-campaigns is one of the topological orders induced by $\mathcal{G}$. Given the interdependence graph $\mathcal{G}$, a formal definition of the influence index $u_{j,t}$ is:

$$u_{j,t} := \frac{1}{K} \sum_{i=1}^{j-1} \sum_{h=t-1}^{t-K} d_{ij}\, q_i(x_{i,h}, y_{i,h}, u_{i,h}), \tag{8.2}$$

where K is a maximum lag order, meaning that users are influenced by ads at most for K consecutive days. Notice that the first sub-campaign $\mathcal{C}_1$, being influenced by no other sub-campaign, has $u_{1,t} = 0$ since the first summation in Equation (8.2) is over an empty set. The above definition of $u_{j,t}$ is based on the assumption that the increase in the number of impressions provided by a user coming from any sub-campaign influences the number of impressions of C_j in the same way. While this assumption might seem simplistic, it is necessary to keep at a pace the complexity of training the model. Indeed, a more complex model, e.g., where there is a different influence index for every pair of sub-campaigns, might be an option, but this would require an excessively large amount of data for the training of the model, which is not a viable option within the time horizon of the optimization process.

The optimization problem in Equations (8.1a)-(8.1d) can be solved using dynamic programming techniques, once all its parameters are known. However, the advertiser does not know the function $q_j(\cdot, \cdot, \cdot)$ that returns the number of impressions for sub-campaign C_j, as well as its click-trough rate r_j and its value per click v_j. Therefore, we resort to learning techniques to produce estimates of these parameters relying on historical data. We assume to have a dataset $Z := \{\boldsymbol{z}_{j,t}\}$ of τ samples that provides, for each day $t \in \{1, \dots, \tau\}$ and each sub-campaign C_j with $j \in \{1, \dots, N\}$, the following values: $\boldsymbol{z}_{j,t} := (\tilde{x}_{j,t}, \tilde{y}_{j,t}, \tilde{q}_{j,t}, \tilde{n}_{j,t}, \tilde{co}_{j,t}, \tilde{c}_{j,t})$. This is a tuple with the used bid $\tilde{x}_{j,t}$ and daily budget $\tilde{y}_{j,t}$, the received impressions $\tilde{q}_{j,t}$, clicks $\tilde{n}_{j,t}$, values of the conversions $\tilde{co}_{j,t}$, and costs $\tilde{c}_{j,t}$. We require that the data collected up to day τ to be exploratory enough to properly model the sub-campaigns interdependences.

Proposed Method: The IDL Algorithm

The pseudo-code of the IDL algorithm is provided in Algorithm 8. It requires a dataset Z and two confidence levels $\alpha_{ADF} \in (0, 1)$ and $\alpha_{GC} \in (0, 1)$ in input. The first phase of the algorithm (Lines 1–8) is called *Interdependence Graph Learning Phase* and is devoted to learning the interdependence graph of the sub-campaigns. The output of this phase is an

Algorithm 8 IDL

```
    Input: dataset $Z$, confidence $\alpha_{ADF}$, confidence $\alpha_{GC}$
    Output: optimal bid/budget/new user allocation $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$

    ▷ Interdependence Graph Learning Phase
 1: for $j \in \{1, \dots, N\}$ do
 2:     $adf_j \leftarrow \mathsf{ADF}(\tilde{\boldsymbol{q}}_{\boldsymbol{j}}, \alpha_{ADF})$
 3: $d_{\max} \leftarrow \max_j\{adf_j\}$
 4: $\hat{P} \leftarrow 0$
 5: for $j \in \{1, \dots, N\}$ do
 6:     for $i \in \{j+1, \dots, N\}$ do
 7:         $\hat{p}_{i,j} \leftarrow \mathsf{GCT}\,(\boldsymbol{q}, i, j)$
 8: $\hat{D} \leftarrow \mathsf{DAG}(\hat{P}, \alpha_{GC})$

    ▷ Estimation and Optimization Phase
 9: for $j \in \{1, \dots, N\}$ do
10:     $\hat{q}_j(\cdot,\cdot,\cdot) \leftarrow \mathsf{GP}(Z, \hat{D}, j)$
11:     $\hat{v}_j \leftarrow \frac{1}{\tau}\sum_{h=1}^{\tau} \frac{\tilde{co}_{j,t}}{\tilde{n}_{j,t}}$
12:     $\hat{w}_j \leftarrow \frac{1}{\tau}\sum_{h=1}^{\tau} \frac{\tilde{n}_{j,t}}{\tilde{q}_{j,t}}$
13: $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*) \leftarrow \mathsf{OPT}(\hat{\boldsymbol{q}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{D})$

14: return $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$
```

estimate $\hat{D}$ of the actual adjacency matrix D. The second phase of the algorithm (Lines 9–13) is called *Estimation and Optimization Phase* and is devoted to the estimation of the parameters for each sub-campaign C_j (i.e., $\hat{q}_j(\cdot,\cdot,\cdot), \hat{v}_j, \hat{w}_j$), using Gaussian Process [48] modeling, and solving the optimization problem in Equations (8.1a)-(8.1d), once the parameters have been replaced with their estimates. The outputs of this phase are $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$, i.e., the optimal bid, daily budget, and influence index for each sub-campaign.

Interdependence Graph Learning Phase

The task of learning $\hat{D}$ is obtained by resorting to the Granger Causality test [24]. This test has been used in many different fields to infer the structure among datastreams, e.g., sensor networks by [4] and economics by [11]. While in its original formulation the test assumes that the analysed time series are stationary, we rely on a generalization of this test, proposed by [55], which is suitable for integrated and cointegrated time series.

The basic idea of this approach is to estimate a Vector AutoRegressive model of order $K_{GR} + d_{\max}$ for the vector $(\widetilde{q}_{1,t}, \dots, \widetilde{q}_{N,t})$, where $d_{\max} \in \mathbb{N}$ is the maximum integration order of the time series that we analyse and $K_{GR} \in \mathbb{N}$ is a lag order which is estimated from the data.[5] The use of $K_{GR} + d_{\max}$ lags ensures that the test statistic used in the Granger Causality test for stationary time series has the same asymptotic distribution of the stationary case and, therefore, statistically valid conclusions can be drawn. More specifically, to test if the impressions of the campaign C_i influence the impressions of the campaign C_j, we estimate the parameters a_{jlm}, for each $m \in \{1, \dots, K_{GR} + d_{\max}\}$, of the model:

$$\widetilde{q}_{j,t} = \sum_{l=1}^{N} \sum_{m=1}^{K_{GR}+d_{\max}} a_{jlm} \, \widetilde{q}_{l,t-m} \;\; \forall h \in \{1, \dots, N\}$$

and we test for the hypothesis:

$$H_0 : \forall m \in \{1, \dots, K_{Gr}\} \; a_{jim} = 0,$$
$$H_1 : \exists m \in \{1, \dots, K_{Gr}\} \mid a_{jim} \neq 0.$$

The complete description of this procedure is provided by [55]. The test states that if we reject H_0 there is evidence, with confidence α_{GC}, that the impressions from C_i are influencing those of C_j.

The IDL algorithm works as follows. For each sub-campaign, we estimate $d_{\max}$ performing the Augmented Dickey Fuller test $\mathsf{ADF}(\tilde{\boldsymbol{q}}_{\boldsymbol{j}}, \alpha_{ADF})$ on the time series $\tilde{\boldsymbol{q}}_{\boldsymbol{j}} := (\widetilde{q}_{j,t}, \dots, \widetilde{q}_{j,\tau})$ with confidence α_{ADF} (Lines 1–3), and inferring the time series order adf_j, and, finally, we perform the Granger Causality test on each pair of sub-campaigns (Lines 5–7). The result of this procedure is a matrix $\hat{P}$ containing the p-values of the pairwise tests, which is used to generate a valid estimate of the adjacency matrix $\hat{D} \in \{0,1\}^{N \times N}$. This operation is performed by $\mathsf{DAG}(\hat{P}, \alpha_{GC})$ (Line 8) by selecting the largest subset S of the p-values $\hat{p}_{ij} < \frac{2\alpha_{GC}}{N(N-1)}$ s.t. the matrix $\hat{D} := \{d_{ij} = 1 \text{ iff } p_{ij} \in S\}$ to correspond to a DAG.[6] This procedure ensures an overall confidence α_{GC} on the Granger Causality test, thanks to the Bonferroni correction for multiple tests, and it avoids that the presence of false positives in the detection of interdependences. Indeed, the edges generated by false positive detections might provide adjacency matrices $\hat{D}$ whose corresponding graph presents cycles, which would compromise the execution of the following optimization procedure.

[5]d_{max} can be estimated using the Augmented Dickey Fuller test [17], which requires a confidence level $\alpha_{ADF} \in (0,1)$, while K_{GR} can be estimated from the dataset Z by standard techniques, see [43] for details.

[6]An adjacency matrix $\hat{D}$ identifies a DAG if and only if a depth-first search of the corresponding graph yields no back edges.

Algorithm 9 OPT$(\hat{\boldsymbol{q}}, \hat{\boldsymbol{v}}, \hat{\boldsymbol{w}}, \hat{D})$

```
Input: estimated adjacency matrix $\hat{D}$, estimated models $\hat{q}_j(\cdot,\cdot,\cdot)$, $\hat{w}_j$, $\hat{v}_j$, discretization of the total budget $\{b_1, \ldots, b_B\}$
Output: optimal bid/budget allocation $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$

1: for $i \in \{1, \ldots, B\}$ do
2:     $\Pi_{1,i,1} \leftarrow (b_i \; \mathbf{0}_{N-1})$
3:     $L_{1,i,1} \leftarrow \hat{v}_1 \; \hat{w}_1 \; \hat{q}_1(\chi_1, b_i, 0)$
4:     $M_{1,i,1} \leftarrow \hat{q}_1(\chi_1, b_i, 0) \; \hat{\boldsymbol{d}}_1$
5: for $j \in \{2, \ldots, N\}$ do
6:     for $i \in \{1, \ldots, B\}$ do
7:         $c \leftarrow 1$
8:         for $k \in \{1, \ldots, i\}$ do
9:             $m = |\{\Pi_{j-1,k,h}\}_h|$
10:            for $h \in \{1, \ldots, m\}$ do
11:                $l \leftarrow \hat{q}_j(\chi_j, b_i - b_k, M_{j-1,k,h}(j))$
12:                $\bar{\Pi}_c \leftarrow (\mathbf{0}_{j-1} \; (b_i - b_k) \; \mathbf{0}_{N-j}) + \Pi_{j-1,k,h}$
13:                $\bar{L}_c \leftarrow \hat{v}_j \; \hat{w}_j \; l + L_{j-1,k,h}$
14:                $\bar{M}_c \leftarrow \; l \; \hat{\boldsymbol{d}}_j + M_{j-1,k,h}$
15:                $c \leftarrow c + 1$
16:        $c \leftarrow 1$
17:        for $h \in \{1, \ldots, |\{Lt_c\}_c|\}$ do
18:            if $\nexists k \mid \bar{L}_h < \bar{L}_k \wedge \forall p \in \{j+1, \ldots, N\} | \bar{M}_h(p) < \bar{M}_k(p)$ then
19:                $\Pi_{j,i,c} \leftarrow \bar{\Pi}_h$
20:                $L_{j,i,c} \leftarrow \bar{L}_h$
21:                $M_{j,i,c} \leftarrow \bar{M}_h$
22:                $c \leftarrow c + 1$
23: for $j \in \{1, \ldots, N\}$ do
24:     $\hat{y}_j^* \leftarrow \max_i \Pi_{N,i,1}(j)$
25:     $\hat{u}_j^*$ computed as in Equation (8.3)
26:     $\hat{x}_j^* = x_j^*(\hat{y}_j^*, \hat{u}_j^*)$
27: return $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$
```

Estimation and Optimization Phase

The second phase of the IDL algorithm exploits predictive models to estimate unknown functions and quantities in the optimization problem defined in Equations (8.1a)-(8.1d), and solves it in a dynamic programming fashion with an *ad hoc* procedure.[7]

We use Gaussian Processes (GPs) to compute, for each sub-campaign

[7]For the sake of presentation in what follows we assume that the number of impressions is monotonically increasing in the influence index. A version of the optimization procedure able to handle general cases is discussed in the final part of this section.

C_j, the function $\hat{q}_j(x, y, u)$ estimating the expected number of impressions $q_j(x, y, u)$, given the chosen bid x, the allocated budget y, and the influence index u generated by the sub-campaigns influencing the sub-campaign C_j (Line 10). The estimate $\hat{r}_j$ of the click-through rate r_j and the estimate $\hat{v}_j$ of the value per click v_j are the average ratios between the number of clicks and the number of impressions and between the number of conversions and the number of clicks, respectively (Lines 9-12). Finally, the estimated influence index is computed as follows:

$$\hat{u}_{j,t} := \frac{1}{K_{GR}} \sum_{i=1}^{j-1} \sum_{h=t-1}^{t-K_{GR}} \hat{d}_{ij}\, \hat{q}_i(x_{i,h}, y_{i,h}, \hat{u}_{i,h}), \tag{8.3}$$

where we use K_{GR}, obtained from the Granger Causality test, as an estimate of the actual lag K.

The optimization procedure is an extension of the optimization algorithm in Chapter 5, to handle also campaigns in which the revenue given by a budget allocated to a sub-campaign depends on the budget allocated to other sub-campaigns. The OPT algorithm, presented in Algorithm 9, takes in input the estimates of the adjacency matrix $\hat{D}$, the number of impressions function $\hat{q}_j(\cdot, \cdot, \cdot)$, the click-trough rate $\hat{r}_j$, the value per click $\hat{v}_j$, and a set of available daily budget values $\{b_1, \dots, b_B\}$, which are, for simplicity, evenly spaced in the range $[0, \overline{B}]$.

The OPT algorithm uses three structures Π, L, and M defined as follows: $\Pi_{j,i,h}$ is a vector that specifies a partial budget allocation with cumulative budget of b_i among the sub-campaigns $C_1, \dots, C_j$; $L_{j,i,h}$ is the revenue provided by the partial budget allocation $\Pi_{j,i,h}$; $M_{j,i,h}$ is a vector that specifies the value of the influence index of the sub-campaigns $C_{j+1}, \dots, C_N$ provided by the sub-campaigns $C_1, \dots, C_j$ when the partial allocation $\Pi_{j,i,h}$ is used. The third index h in the structures mentioned above is necessary since the algorithm may need to store multiple partial budget allocations for each j and i. More precisely, the set $\{\Pi_{j,i,h}\}_h$ contains Pareto-efficient partial budget allocations, where the optimality criteria are the revenue and the influence indices of campaigns $C_{j+1}, \dots, C_N$. For instance, given two partial budget allocations Π_{j,i,h_1} and Π_{j,i,h_2}, where the former has high revenue and a small number of impressions and the latter *vice versa*, it is not possible to decide which one is the optimal before evaluating their influence on the sub-campaigns $C_{j+1}, \dots, C_N$ and therefore we need to store both.

At first, the algorithm initializes the values of the structures for $j = 1$ (Lines 1–4), corresponding to the allocations of the partial budget to the sub-campaign C_1. For each budget b_i, we allocate it to C_1, formally,

$\Pi_{1,i,1} = (b_i, \mathbf{0}_{N-1})$, where $\mathbf{0}_{N-1}$ denotes a null vector of size $N-1$. The sub-campaign C_1, being the first in the topological ordering induced by $\hat{D}$, is not subject to any interdependence from other sub-campaigns. Therefore, the computation of the revenue $\{L_{1,i,1}\}_i$ and the influence index vector $\{M_{1,i,1}\}_i$ is performed using the previously estimated models.[8] The vector $M_{1,i,1}$ is computed as $M_{1,i,1} = n_1(\chi_1, b_i, 0)\ \hat{\boldsymbol{d}}_1$, where $\hat{\boldsymbol{d}}_i$ is the i-th row of the adjacency matrix $\hat{D}$. This means that $M_{1,i,1}(j)$, i.e., the j-th element of $M_{1,i,1}$, is equal to $n_1(\chi_1, b_i, 0)$ if the sub-campaign C_1 influences the campaign C_j and zero otherwise.

For all the $j \in \{2, \dots, N\}$, the algorithm computes the elements of the three structures Π, L, and M using the values previously computed at the $j-1$-th step, in a dynamic programming fashion (Lines 5–22). For each daily budget b_i and for each daily budget $b_k \leq b_i$, we compute the revenue and the influence index provided by the allocation of a daily budget of $b_i - b_k$ to the sub-campaign C_j and the remaining daily budget of b_k to the sub-campaigns $C_1, \dots, C_{j-1}$. We do this by enumerating all the Pareto-efficient partial allocations $\Pi_{j-1,k,1}, \Pi_{j-1,k,2}, \dots$ of the first $j-1$ sub-campaigns, then allocating daily budget $b_i - b_k$ to the sub-campaign C_j and, finally, we evaluate the total revenue $\bar{M}_c$ and the influence indices vector $\bar{L}_c$ provided by the partial allocations obtained, denoted with $\bar{\Pi}_c$ (Lines 9–15).[9] After that, the algorithm discards all the candidate partial allocations which are Pareto dominated (Lines 16-22); see [20] for details on Pareto efficiency and dominance.[10] Finally, the algorithm returns the optimal allocation (Lines 23–27): the optimal budgets $\hat{y}_j^*$ are the elements of $\max_i \Pi_{N,i,1}(j)$; the optimal influence indices $\hat{u}_j^*$ are computed using Equation (8.3); the optimal bids $\hat{y}_j^*$ are computed using the impressions models $\hat{n}_j(\cdot,\cdot,\cdot)$. The complexity of the OPT algorithm is $O\left(\sum_j B^{\sum_i \hat{d}_{ij}+2}\right) \leq O\left(N\, B^2\, B^{\max_j \sum_i \hat{d}_{ij}}\right)$ and strictly depends on the maximum indegree of the interdependence graph corresponding to $\hat{D}$. The complexity reduces to that one of the algorithm proposed by [41] when the sub-campaigns are not interdependent. Notice that capturing only the pairs of sub-campaigns with the most significant interdependence is a crucial issue from a computational point of view since, taking into ac-

[8]We define χ_j as the bid that maximise the number of impressions given a budget y and a influence index u, or, formally, $\chi_j := \chi_j(y,u) = \arg\max_x \hat{q}_j(x,y,u)$.

[9]In the pseudo-code, we denoted the number of Pareto optimal allocations at the $j-1$-th row with a budget of b_i with $|\{\Pi_{j-1,i,h}\}_h|$.

[10]Notice that the inequality in Line 18 is designed for settings in which the number of impressions is monotonically increasing in the influence index. However, removing the condition in Line 18, the proposed method also applies to problems without such a monotonicity assumption. This comes at at the cost of storing a larger number of partial allocations in $\{\Pi_{j,i,h}\}_h$.

count all the possible pairs of sub-campaigns, the complexity is bounded by $NB^2\frac{B^{N+1}-1}{B-1}$, which is intractable when N is large as it happens in real-world applications.

Theoretical Properties

We analyse the properties of our problem and those of the IDL algorithm. Initially, we analyse the suboptimality of any algorithm ignoring the sub-campaigns interdependencies w.r.t. our algorithm, i.e., when the learner uses an adjacency matrix $\hat{D} = 0$, and the real one D is non-null. The following theorem shows that ignoring the sub-campaigns interdependences might be arbitrarily suboptimal.

Theorem 12. *Given the problem of optimizing an advertising campaign $\mathcal{C}$, employing a model $\hat{n}_j(x, y)$ for the number of impressions that ignores the sub-campaigns interdependence may result in an arbitrary large loss in terms of* revenue*, defined as:*

$$R_t = \sum_{j=1}^{N} v_j \; r_j \; q_j(x_{j,t}, y_{j,t}, u_{j,t}). \tag{8.4}$$

Proof. Consider a campaign $\mathcal{C}$ with $N = 2$ sub-campaigns, adjacency matrix $D = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, click-through rates $w_1 = w_2 = 1$, values per click $v_1 = 0$ and $v_2 = 1$, lag $K = 1$, and impression functions identified by two GPs having the following mean value:

$$\begin{aligned} &n_1(x_{1,t}, y_{1,t}, u_{1,t}) = H\, y_{1,t}, \\ &n_2(x_{2,t}, y_{2,t}, u_{2,t}) = \Upsilon\, u_{2,t}\, y_{2,t} = \Upsilon\, n_1(x_{1,t-1}, y_{1,t-1}, u_{1,t-1})\, y_{2,t}, \end{aligned}$$

where $\Upsilon \in [0, 1]$, $H \in \mathbb{R}^+$. Moreover, assume to have a total budget of $\overline{B}$ and to sample the solution space uniformly during training. Asymptotically (when we have an infinite number of samples or $\tau \to \infty$), a model that provides a stationary allocation and knows the sub-campaign interdependencies would compute the revenue:

$$\begin{aligned} R_t &= \sum_{j=1}^{N} v_j \; r_j \; q_j(x_{j,t}, y_{j,t}, u_{j,t}) = \Upsilon\, n_1(x_1, y_1, u_1)\, y_2 \\ &= \Upsilon\, H\, y_1\, y_2 = \Upsilon\, H\, (\overline{B} - y_2)\, y_2, \end{aligned}$$

where we drop the temporal indices since the proposed solution is stationary and $y_1 + y_2 = \overline{B}$. The budget allocation maximizing the revenue is $y_2 = \frac{\overline{B}}{2}$, with a revenue $R_t = \Upsilon\, H\, \frac{\overline{B}^2}{4}$.

On the other hand, a model ignoring the sub-campaigns interdependence estimates the impressions of the sub-campaign C_2 as:

$$\begin{aligned}\tilde{n}_2(x_{2,t}, y_{2,t}) &= \int_0^{H\overline{B}} n_2(x_{2,t}, y_{2,t}, u) du \\ &= \int_0^{H\overline{B}} \Upsilon u\, y_{2,t} du = \frac{\Upsilon H^2 \overline{B}^2}{2} y_{2,t},\end{aligned}$$

and the revenue one maximises becomes:

$$\tilde{R}_t = \sum_{j=1}^{N} v_j\, r_j\, q_j(x_{j,t}, y_{j,t}, u_{j,t}) = \tilde{n}_2(x_{2,t}, y_{2,t}) = \frac{\Upsilon H^2 \overline{B}^2}{2} y_2.$$

The budget allocation maximising $\tilde{R}_t$ is $y_2 = \overline{B}$, with a real revenue $R_t = 0$. Hence, the difference of revenue of the two algorithms is $\Upsilon H \frac{\overline{B}^2}{4} - 0$, which is arbitrarily large as H goes to ∞. □

When the model is flexible enough to model the actual process properly, we can bound its error, formally, defined as follows:

Definition 7. *Given a dataset Z, the* total (estimation) error *is:*

$$E_\tau := \sum_{j=1}^{N} \left[v_j r_j q_j(x_j^*, y_j^*, u_j^*) - \hat{v}_j \hat{r}_j \hat{q}_j(\hat{x}_j^*, \hat{y}_j^*, \hat{u}_j^*) \right],$$

where the tuples $(\hat{x}_j^*, \hat{y}_j^*, \hat{u}_j^*)$ are elements of the output $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$ of the IDL *algorithm using the estimates of the parameters, and (x_j^*, y_j^*, u_j^*) are elements of the output of the* IDL *algorithm using the real parameters.*

We can show the following:

Theorem 13. *When the expected number of impressions $n_j(\cdot, \cdot, \cdot)$ of every sub-campaign C_j is distributed as a Gaussian Process, the total error between the real revenue and the estimated one using the output of the* IDL *algorithm is upper bounded, with a probability of at least $1 - \delta$, as follows:*

$$\begin{aligned}E_\tau \leq\ & 2N v^{(\max)} \sqrt{\frac{1}{2\tau} \log \frac{6N}{\delta}} \left(\hat{q}^{(\max)} + \hat{\sigma}_\tau^{(\max)} \sqrt{2 \log \frac{3N}{\delta}} \right) \\ & + N v^{(\max)}\, \hat{\sigma}_\tau^{(\max)} \sqrt{2 \log \frac{3N}{2\delta}},\end{aligned}$$

where $\hat{q}^{(\max)} := \max_j \max_{(x,y,u)} \hat{q}_j(x,y,u)$ *is the maximum number of estimated expected impressions over all the sub-campaigns,*
$\hat{\sigma}_\tau^{(\max)} := \max_j \max_{(x,y,u)} \hat{\sigma}_{j,\tau}(x,y,u)$ *is the maximum estimated standard deviation, and* $v^{(\max)}$ *is the maximum value per click.*

We remark that [48] show that, in a generic GP, $\hat{\sigma}_\tau^{(\max)} \to 0$ as $\tau \to \infty$. Therefore, the total error E_τ decreases as the number of samples τ in the training set Z increases.

Proof. Since estimates for click-through rate $\hat{r}_j$ and value per click $\hat{v}_j$ are sum of i.i.d. random variables with finite support $[0,1]$ and $[0, v^{(\max)}]$, respectively, we can apply the Hoeffding's bound [59] and state that, with a probability of at least $1-\delta$:

$$r_j - \hat{r}_j \leq \sqrt{\frac{1}{2\tau}\log\frac{1}{\delta}}, \tag{8.5}$$

$$v_j - \hat{v}_j \leq v^{(\max)}\sqrt{\frac{1}{2\tau}\log\frac{1}{\delta}}, \tag{8.6}$$

where τ is the number of samples we use to compute the estimates. By assumption, the number of impressions are generated by a GP and we have that for each input (x,y,u) in the GP domain $\frac{q_j(x,y,u)-\hat{q}_j(x,y,u)}{\hat{\sigma}_{j,\tau}(x,y,u)} \sim \mathcal{N}(0,1)$, where $\hat{\sigma}_{j,\tau}(x,y,u)$ is the standard deviation computed by the GP at the point (x,y,u) by relying on τ samples in the training set Z. This implies that, with a probability of at least $1-\delta$, it holds:

$$q_j(x,y,u) \leq \hat{q}_j(x,y,u) + \hat{\sigma}_{j,\tau}(x,y,u)\sqrt{2\log\frac{1}{2\delta}}, \tag{8.7}$$

where the last inequality is due to the fact that, for a Gaussian random variable $X \sim \mathcal{N}(0,1)$, it holds $\forall\, x > 0,\ \mathbb{P}(X > x) \leq \frac{1}{2}e^{-\frac{x^2}{2}}$.

Let us focus on E_τ. If we sum and subtract from it the following quantities: $\hat{r}_j v_j q_j(x_j^*, y_j^*, u_j^*)$, $\hat{r}_j \hat{v}_j q_j(x_j^*, y_j^*, u_j^*)$, and $\hat{r}_j \hat{v}_j \hat{q}_j(x_j^*, y_j^*, u_j^*)$ for each

$j \in \{1, \dots, N\}$, the total error can be decomposed as:

$$E_\tau = \sum_{j=1}^{N} \Bigg(\underbrace{(r_j - \hat{r}_j) v_j q_j(x_j^*, y_j^*, u_j^*)}_{E_{1j}} + \underbrace{\hat{r}_j (v_j - \hat{v}_j) q_j(x_j^*, y_j^*, u_j^*)}_{E_{2j}} + \underbrace{\hat{r}_j \hat{v}_j [q_j(x_j^*, y_j^*, u_j^*) - \hat{q}_j(x_j^*, y_j^*, u_j^*)]}_{E_{3j}} \Bigg) + \underbrace{\sum_{j=1}^{N} \hat{r}_j \hat{v}_j \hat{q}_j(x_j^*, y_j^*, u_j^*) - \sum_{j=1}^{N} \hat{r}_j \hat{v}_j \hat{q}_j(\hat{x}_j^*, \hat{y}_j^*, \hat{u}_j^*)}_{E_4}.$$

If we focus on E_{1j}, with probability at least $1 - \delta$, it holds:

$$\begin{aligned} E_{1j} &= (r_j - \hat{r}_j)\, v_j\, q_j(x_j^*, y_j^*, u_j^*) \\ &\le v^{(\max)} \sqrt{\frac{1}{2\tau} \log \frac{2}{\delta}} \left(\hat{q}_j(x_j^*, y_j^*, u_j^*) + \hat{\sigma}_{j,\tau}(x_j^*, y_j^*, u_j^*) \sqrt{2 \log \frac{1}{\delta}} \right) \\ &\le v^{(\max)} \sqrt{\frac{2}{\tau} \log \frac{2}{\delta}} \left(\hat{q}_j^{(\max)} + \hat{\sigma}_{j,\tau}^{(\max)} \sqrt{2 \log \frac{1}{\delta}} \right), \end{aligned}$$

by relying on the inequalities in Equations (8.5) and (8.7), using a union bound over these two events, then defining $\hat{q}_j^{(\max)} := \max_{(x,y,u)} \hat{q}_j(x, y, u)$ and $\hat{\sigma}_{j,\tau}^{(\max)} := \max_{(x,y,u)} \hat{\sigma}_{j,\tau}(x, y, u)$, and finally since $v_j \le v^{(\max)}$.

Similarly, we derive the following bound holding with probability at least $1 - \delta$ for E_{2j}:

$$E_{2j} \le v^{(\max)} \sqrt{\frac{1}{2\tau} \log \frac{2}{\delta}} \left(\hat{q}_j^{(\max)} + \hat{\sigma}_{j,\tau}^{(\max)} \sqrt{2 \log \frac{1}{\delta}} \right),$$

by relying on the inequality in Equation (8.6), and the fact that $r_j \le 1$.

Let us focus on E_{3j}. Using the inequality in Equation (8.7), we have that with probability at least $1 - \delta$:

$$\begin{aligned} E_{3j} &= \hat{r}_j\, \hat{v}_j\, [q_j(x_j^*, y_j^*, u_j^*) - \hat{q}_j(x_j^*, y_j^*, u_j^*)] \\ &\le v^{(\max)}\, \hat{\sigma}_{j,\tau}(x_j^*, y_j^*, u_j^*) \sqrt{2 \log \frac{1}{2\delta}} \le v^{(\max)}\, \hat{\sigma}_{j,\tau}^{(\max)} \sqrt{2 \log \frac{1}{2\delta}}, \end{aligned}$$

where we used $\hat{r}_j \le 1$, and $\hat{v}_j \le v^{(\max)}$.

Finally, let us focus on E_4. The vector $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$ is the optimal solution of the optimization problem stated in Equations (8.1a)-(8.1d). Therefore, by definition, we have that for each $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{u})$ satisfying the constraints in Equations (8.1a)-(8.1d) the following holds:

$$\sum_{j=1}^{N} \hat{r}_j \hat{v}_j \hat{q}_j(x_j, y_j, u_j) - \sum_{j=1}^{N} \hat{r}_j \hat{v}_j \hat{q}_j(\hat{x}_j^*, \hat{y}_j^*, \hat{u}_j^*) \leq 0,$$

which holds also for $(\boldsymbol{x}^*, \boldsymbol{y}^*, \boldsymbol{u}^*)$ and, therefore, E_4 is negative.

Recalling that $\hat{q}^{(\max)} := \max_j \hat{q}_j^{(\max)}$ and $\hat{\sigma}_\tau^{(\max)} := \max_j \hat{\sigma}_{j,\tau}^{(\max)}$, it holds, with probability at least $1 - \delta$:

$$\begin{aligned} E_\tau &= \sum_{j=1}^{N} (E_{1j} + E_{2j} + E_{3j}) + E_4 \\ &\leq 2N v^{(\max)} \sqrt{\frac{1}{2\tau} \log \frac{6N}{\delta}} \left(\hat{q}^{(\max)} + \hat{\sigma}_\tau^{(\max)} \sqrt{2 \log \frac{3N}{\delta}} \right) + \\ &\quad N v^{(\max)}\, \hat{\sigma}_\tau^{(\max)} \sqrt{2 \log \frac{3N}{2\delta}}, \end{aligned}$$

which concludes the proof. □

Our analysis has, so far, focused on the static properties of our problem. However, the scenario we are studying is a dynamical system due to the potentially delayed effects induced by the sub-campaigns interdependence. Therefore, it is crucial to show that, whenever a stationary allocation is used, the dynamics always reach a steady state in finite time and how their length is upper bounded. In this context, a steady state allocation provides a constant number of impressions for each sub-campaign for at least K consecutive days. We can show the following:

Theorem 14. *Using the stationary allocation $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$ we reach a steady state after at most $K\,\Gamma + 1$ days, where K is the maximum lag of the influence index $u_{j,t}$ and Γ is the length of the longest path of the graph $\mathcal{G}$.*

The above theorem states that the more complex the process (e.g., presenting a cascade of interdependences), the more we have to wait to completely remove the effects of a suboptimal allocation.

Proof. Consider an adjacency matrix $\hat{D}$ and assume that the lag K is the same for all the performance indices $u_{j,t}$. Moreover, consider the sub-campaigns $C_{i_1}, \ldots, C_{i_\Gamma}$ that make up the longest path on the graph $\mathcal{G}$. To achieve the steady state revenue provided by the allocation $(\hat{x}^*_{i_\Gamma}, \hat{y}^*_{i_\Gamma}, \hat{u}^*_{i_\Gamma})$, an algorithm needs that all the incoming-neighbour sub-campaigns have reached the optimal allocation for K consecutive days, so as to provide exactly $\hat{u}^*_{i_\Gamma}$ as influence index. In particular, this also happens for the sub-campaign C_{i_Γ}.

By induction, this reasoning can be applied for all nodes C_{i_h} in the longest path up to C_{i_2}. Therefore, every time we traverse a node, we require K days to conclude the transient for that node, for a total of $K\,\Gamma$ days. Instead, the node C_{i_1}, being at the beginning of the longest path, has no incoming neighbours and, therefore, the allocation prescribed by the optimal solution is $(\hat{x}^*_{i_\Gamma}, \hat{y}^*_{i_\Gamma}, 0)$. This implies that the allocation is achieved on the same day that the stationary allocation is used, leading to a total number of $K\,\Gamma + 1$ days to reach the desired allocation on the longest path $C_{i_1}, \dots, C_{i_\Gamma}$.

The same reasoning can be replicated on any other path, but since their length is shorter or equal to the longest one, the maximum number of days required to reach the allocation $(\hat{\boldsymbol{x}}^*, \hat{\boldsymbol{y}}^*, \hat{\boldsymbol{u}}^*)$ takes no longer than $K\,\Gamma + 1$ days. $\square$

Experimental Evaluation

We experimentally evaluate the IDL algorithm in a real-world setting and in a synthetic setting, generated by using a realistic simulator. We compare the revenue R_t produced by IDL and AdComB-Mean (an off-line version of the algorithm proposed in Chapter 5 neglecting any sub-campaign interdependence).

Evaaluation in a Real-world Setting

In this experiment, we rely on the data of the second campaign described in Section 8.1 to train our model. We recall that the length of the dataset is $\tau = 93$ days (from 20/7/2018 to 20/10/2018), the advertising campaign is composed of $N = 14$ sub-campaigns belonging to both social and search advertising channels. The corresponding estimated interdependence graph is provided in Figure 8.1c. From 21/10/2018 to 4/11/2018 (15 days), the campaign optimization has been performed by the IDL algorithm.

When comparing the policies produced by IDL with those produced by AdComB-Mean (the off-line version of AdComB-TS), the former policies appear more suitable then the latter ones, as a more significant portion of the budget is allocated to social sub-campaigns and branding search sub-campaigns. The interdependence suggested by the Granger Causality Test

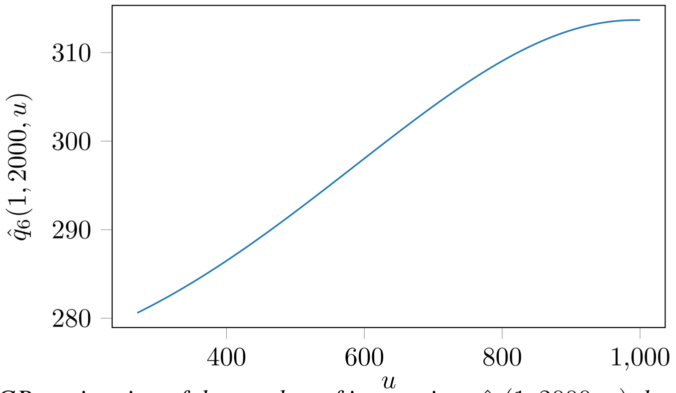


Figure 8.2: *GPs estimation of the number of impressions $\hat{q}_6(1, 2000, u)$ depending on the influence index u.*

are confirmed by estimations provided by the GPs. Indeed, in Figure 8.2, we show the expected value of the prediction provided by GPs of the number of impressions for the sub-campaign C_6 with a bid value of $x = 1$ (i.e., one of the most frequent choice during the training set) and $y = 2000$ (i.e., a budget large enough to capture all the available user for this sub-campaign). The number of impressions increases as the value of the influence index increases, suggesting that a positive correlation between C_2 and C_3 impressions, and C_6 ones exist. However, since in a real setting we cannot exclude the presence of negative interdependence, to compute the optimal allocation with the IDL algorithm, we remove the condition in Line 18 of Algorithm 9, to be able to provide the optimal allocation even if generic interdependence among sub-campaigns are present. In Figure 8.3, we show the expected revenue given by optimal policies computed by AdComB-Mean and IDL for different values of the total budget Y. In this scenario, the exploitation of the sub-campaigns interdependence can lead to a potential revenue increase up to 13%.

In the 15 days of campaign optimization performed by the IDL algorithm, the number of daily conversions increased by 11% w.r.t. the average of the previous 30 days (the result is compatible with our prediction, given that AdComB-TS/Mean provide very close performance). Although this is a promising result, there is no statistical significance that the IDL algorithm outperforms in practice AdComB-TS/Mean. Due to the impossibility to directly compare the performance of the two algorithms online (e.g., by using an A/B testing system), we resort to a realistic synthetic environment.

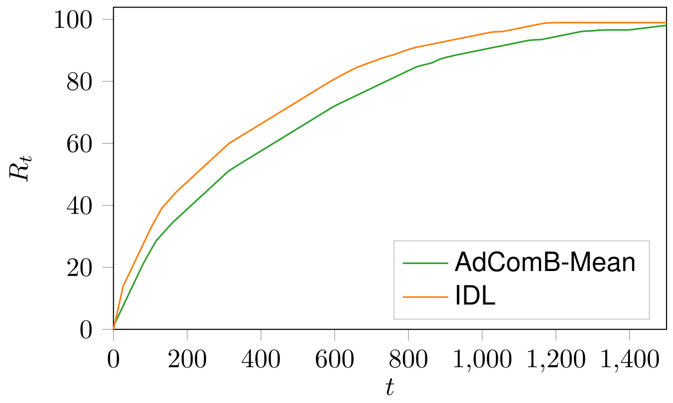


Figure 8.3: *Comparison of the expected revenue R_t given by the AdComB-Mean and IDL algorithms.*

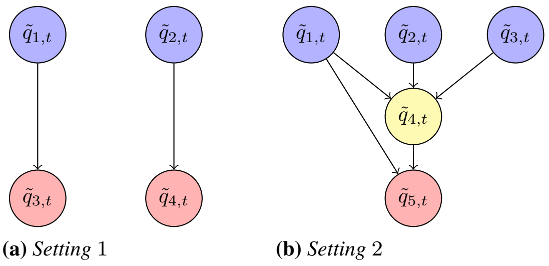


Figure 8.4: *Interdependence graph $\mathcal{G}$ for the two synthetic experimental settings.*

Evaluation in Synthetic Settings

We evaluate the performance of the IDL algorithm in two synthetic settings, generated by a realistic simulator, comparing the revenue R_t produced by the following algorithms: IDL, DA-IDL (Dependency Aware-IDL), a variation of the IDL algorithm *a priori* knowing the dependency matrix D, and AdComB-Mean.

Synthetic Data Generation The synthetic settings are generated as follows. At day t, each sub-campaign C_j is characterized by the set of the users $S_{j,t} = s_{j,t} \cup \left(\bigcup_{i \neq j} s_{ij,t} \right)$ that could potentially visualize the ad of the sub-campaign C_j. More precisely, we distinguish the set of the users $s_{j,t}$, that would visualize the ad of C_j without having previously visualized the ads of the other interdependent sub-campaigns, from the set of the users $s_{ij,t}$, that would visualize the ad of C_j only after having visualized the ad of

C_i. Notice that $s_{ij,t}$ is non-empty only if the sub-campaigns C_i and C_j are interdependent and, more precisely, if $d_{ij} \neq 0$.

The number of users $|s_{j,t}|$ is sampled from $\mathcal{N}(\mu_j, \sigma_j^2)$, i.e., a Gaussian distribution with mean μ_j and variance σ_j^2. Each user in $s_{j,t}$ is characterized by a click probability $p_j^{(cl)}$ and a conversion probability $p_j^{(co)}$ specific for the sub-campaign C_j. Conversely, the number of users $|s_{ij,t}|$ is modeled trough a linear combination of the number of daily impressions $q_{i,t-1}, \dots, q_{i,t-K}$ (whose generation is described in what follows), where K represents the maximum delay in the interdependence dynamics. Formally, we have that $s_{ij,t} := p_{ij}^{(res)} \sum\limits_{k=1}^{K} \beta_k \, n_{i,t-k}$, where $\beta_k \in [0, 1]$ are randomly sampled coefficients and $p_{ij}^{(res)}$ is the probability that a user having visualized ad of C_i is a potential user that may visualize C_j. Each user in $s_{ij,t}$ is characterized by a click probability $p_{ij}^{(cl)}$ and a conversion probability $p_{ij}^{(co)}$.

At each day t, setting the bid/budget pairs on each sub-campaign allows the advertiser to take part to $A_j \leq |S_{j,t}|$ auctions based on the Vickrey-Clarke-Groves mechanism [38], in which γ_j available ad slots are allocated to a subset of δ_j advertisers ($\gamma_j \leq \delta_j$). More specifically, each advertiser submits her bid b_h and those with the first γ_j highest values $b_h \; \rho_h$ are allocated in the γ_j slots, where ρ_h is the probability that h-th ad is clicked given it has been observed. The bids b_h of the other ads participating in the auctions are drawn from a truncated Normal distribution $\mathcal{N}(\mu^{(b)}, \sigma^{(b)})$, and the click probabilities ρ_h are uniformly sampled in $[0, 1]$. In the case the advertiser wins the m-th auction, the ad gets an impression ($q_{m,j,t} = 1$, otherwise $q_{m,j,t} = 0$). The ad is allocated in a the l-th slot, the ad can be visualized by an user $S_{j,t}$ according to the probability of being observed $p^{(obs)}(l)$. After the impression, the user can click on the ad and generate a conversion according to the click $p_j^{(cl)}$ and conversion $p_j^{(co)}$ probabilities if the user belongs to $s_{j,t}$, and according to the click $p_{ij}^{(cl)}$ and conversion $p_{ij}^{(co)}$ probabilities if the user belongs to $s_{ij,t}$. A click on the ad of C_j provided by the user corresponding to the m-th auction is denoted by $n_{m,j,t} = 1$ ($n_{m,j,t} = 0$ otherwise), and imposes a payment of $CPC_{m,j,t}$, as specified by the VCG auction (see [38] for details). The auctions are generated until the daily budget $y_{j,t}$ allocated on the sub-campaign C_j is totally spent, i.e., the total number of auctions A_j is s.t. $\sum_{m=1}^{A_j} CPC_{m,j,t} = y_{j,t}$ or until $A_j = |S_{j,t}|$. Finally, in the case a click happen, the m-th user may convert ($co_{m,j,t} = 1$) or not ($co_{m,j,t} = 0$). The daily impressions, the daily clicks, the daily conversions (assuming unitary value per conversion), and

Table 8.1: *Parameters of the synthetic settings.*

	Setting 1				Setting 2				
	C_1	C_2	C_3	C_4	C_1	C_2	C_3	C_4	C_5
μ_j	5000	5000	200	1300	10000	10000	10000	700	500
σ_j	10	10	10	10	10	10	10	5	2
γ_j^2	5	5	3	3	5	5	5	3	4
δ_j	5	6	4	5	5	5	6	4	5
$\mu^{(b)}$	0.89	1.19	1.59	1.59	0.10	0.10	0.10	1.0	1.5
$\sigma^{(b)}$	0.32	0.12	0.2	0.2	0.032	0.032	0.012	0.2	0.2
$p^{(obs)}(1)$	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
$p^{(obs)}(2)$	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8	0.8
$p^{(obs)}(3)$	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7	0.7
$p^{(obs)}(4)$	0.6	0.6	-	-	0.6	0.6	0.6	-	0.65
$p^{(obs)}(5)$	0.5	0.5	-	-	0.5	0.5	0.5	-	-
$p_j^{(cl)}$	0.2	0.2	0.4	0.4	0.2	0.2	0.2	0.3	0.4
$p_j^{(co)}$	0.001	0.05	0.2	0.2	0.001	0.001	0.011	0.06	0.3

the daily costs are computed as $q_{j,t} = \sum_{m=1}^{A_j} q_{m,j,t}$, $n_{j,t} = \sum_{m=1}^{A_j} n_{m,j,t}$, $co_{j,t} = \sum_{m=1}^{A_j} co_{m,j,t}$, respectively.

We report in Table 8.1 the values of the main parameters used in the two synthetic settings in which we test our algorithm.

Experiment #1 There are $N = 4$ sub-campaigns, with delayed dynamics of $K = 5$ days, whose interdependence graph is shown in Figure 8.4a. The longest path of the interdependence graph $\mathcal{G}$ is $\Gamma = 1$. C_1 and C_2 are on the display advertising channel and are targeted to a wide range of daily users, thus generating a large number of daily auctions, but their conversion probability is low. C_3 and C_4 are on the search advertising channel, generating a small number of daily auctions, but their conversion probability is high.

We use $\overline{B} = 500$ and $B = 10$ daily budget values evenly spaced in the range $[0, 500]$. The GPs used to estimate the impressions model of the sub-campaigns adopt a squared exponential kernel in which the kernel parameters are chosen as recommended by [48]. We evaluate the performance of the algorithms with different numbers of samples $\tau \in \{60, 80, 100\}$ in the training set Z. In the first τ days, a uniformly random allocation is used to collect data and, after that, the algorithms compute the optimal solution based on their estimates and then set it. In Figure 8.5a, we report the average (over 100 repetitions) revenue R_t produced by the algorithms with a training of $\tau = 100$ samples. From $t = 100$ on, the optimal stationary solution is used. The average revenue of each algorithm peaks at $t = 101$ and, for $t > 101$, decreases by converging to a steady state within $K\,\Gamma + 1 = 6$ days. The peak is generated by the presence of a large number of residual users who have observed display ads during training and who, after $t = 100$, observe search ads. These residual users decrease for $t > 101$ until they reach a steady state. Thus, (temporary) peaks may be achieved with non-stationary policies.

The DA-IDL algorithm exhibits the best performance, exploiting the *a priori* knowledge of the adjacency graph D. The gap between the revenue produced by the IDL and DA-IDL algorithms, due to the estimation error introduced on $\hat{D}$, is sufficiently small, showing that the Granger Causality test used by the IDL algorithm works well in practice. Instead, the revenue produced by the AdComB-Mean algorithm, neglecting the interdependence among sub-campaigns, is much smaller than that produced by the other two algorithms. This is due to the very different budget allocations chosen by the three algorithms: the IDL and DA-IDL algorithms optimally balance the budget on all the sub-campaigns, while the AdComB-Mean algorithm greedily invests the budget only in the search sub-campaigns C_3 and C_4. Interestingly, the performance of the AdComB-Mean algorithm is quite similar to that of the uniformly random allocation used during training.

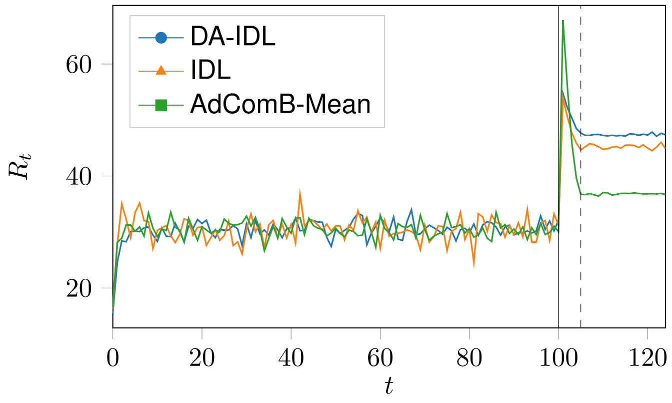
In Figure 8.5b, we report the average revenue R_t at the steady-state (averaged over the 100 independent repetitions and over $t \in \{106, \dots, 120\}$) and the 95% confidence intervals as the number of samples τ used for training increases. All algorithms always perform better than the uniformly random allocation. The performance of both the IDL and DA-IDL algorithms is significantly better than the one provided by AdComB-Mean (confidence intervals do not overlap). The use of more training samples provides an improvement in terms of steady-state revenue for the IDL and DA-IDL algorithms. On the other hand, the performance of the AdComB-Mean algorithm does not benefit from having more samples, which is probably due to the presence of a model bias induced by the fact that it neglects the sub-campaign interdependence.

Experiment #2 There are $N = 5$ sub-campaigns, whose interdependence graph is shown in Figure 8.4b. The longest path of the interdependence graph $\mathcal{G}$ is $\Gamma = 2$. C_1, C_2, and C_3 are display sub-campaigns directed to a wide audience and have a low cost per impression, but a low conversion rate. C_4 is a social sub-campaign, whose number of impressions is influenced by the influence index of the display sub-campaigns. Finally, C_5 is a search sub-campaign, whose impressions depend on the influence index of

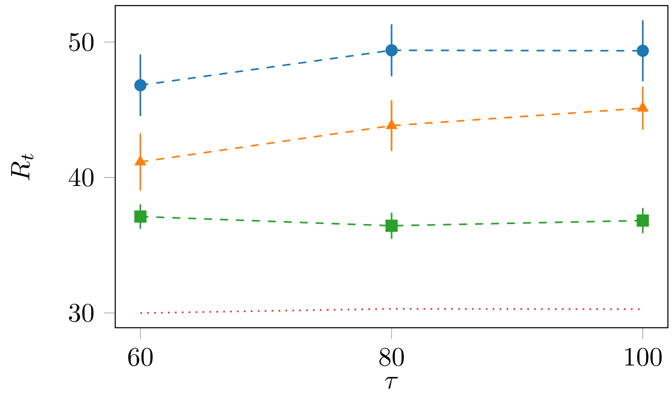
C_1 and C_4. The interdependence among the sub-campaigns occurs within $K = 3$ days and is modeled as in Setting 1. We set a cumulative budget of $\overline{B} = 500$ and the budget discretization from the interval $[0, 500]$ with $B = 100$. The number of samples for training is $\tau \in \{100, 150, 200\}$.

In Figure 8.5c, we report the average (over 100 repetitions and over $t \in \{107, \ldots, 120\}$) revenue of the algorithms. The performance of AdComB-Mean is worse than the one of the uniformly random allocation and gets worse as τ increases. This is an empirical confirmation of the statement of Theorem 12, showing that a solution that is optimal without interdependence might perform arbitrarily bad. Conversely, the performance of IDL and DA-IDL are significantly larger than that of the uniformly random allocation and increase as the number of samples increases.

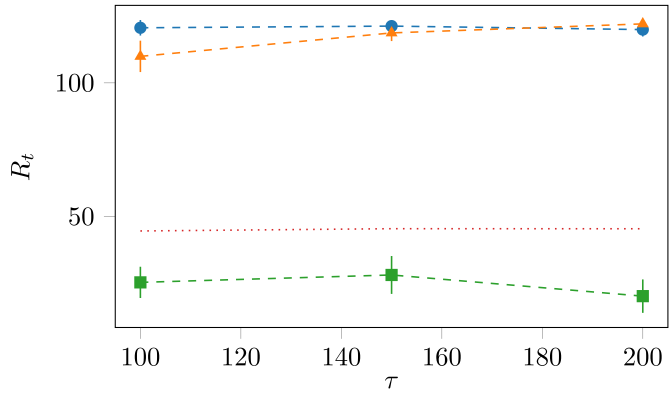
Final Remarks Results obtained in synthetic settings show that this model, relying on a training time which is reasonable for the application, provides a significant improvement in terms of revenue of an Internet advertising campaign. Experts in the marketing field confirmed the feasibility of what proposed in terms of learning time. Conversely, adopting more complex models would most likely result unaffordable in most of the cases, since accurate estimations would require a larger training set and, therefore, excessively long learning periods in real-world scenarios.



(a)



(b)



(c)

Figure 8.5: *Results for the Settings* 1 *and* 2. *(a) Revenue* R_t *over time for the Setting* 1. *(b) Revenue* R_t *in steady state conditions for different training sizes* τ *in Setting* 1. *(c) Revenue* R_t *in steady state conditions for different training sizes* τ *in Setting* 2. *In (b) and (c), the revenue of the random allocation is reported with a dotted magenta line and the vertical lines represent the* 95% *confidence intervals for the algorithms revenue.*

CHAPTER 9

Conclusions and Future Research

Internet advertising campaigns optimization is a challenging problem involving different tasks and sub-problems that can not be efficiently addressed by human agents. The optimization of these tasks is of paramount importance for companies and media agencies that need automatic tools to support marketers in campaigns management and increase their revenue. In this thesis, we presented a set of algorithms addressing the most important problems involved in the internet advertising optimization. First, we focused on the joint optimization of the bid and the budget of a set of advertising campaigns. We presented the AdComB algorithm, a method capable of choosing the values of the bid and the daily budget of a set of Internet advertising sub-campaigns to maximize, in an online fashion, the revenue under a budget constraint. The algorithm exploits Gaussian Processes to estimate the campaigns performance, combinatorial bandit techniques to address the exploration/exploitation dilemma in the bid/daily budget choice, and a dynamic programming procedure to solve the allocation optimization problem. We propose four flavors of our approach: AdComB-U-UCB which uses an unfactorized model for the bid/daily budget space and confidence bounds for exploration, AdComB-U-TS which uses an unfactorized model and sampling as exploration strategy, AdComB-F-UCB which uses a factorized model and upper confidence bounds, and AdComB-F-TS which uses a factorized model and sampling for exploration. We theoretically analyze our algorithms and we provide high probability bounds on the regret of $\tilde{O}(\sqrt{T})$, where T is the time horizon of the learning process. Our experimental results, on both synthetic settings and real-world settings, 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 design an algorithm for the adaptive discretization of the bid and budget space, depending on the complexity of the setting and the time horizon T. Furthermore, 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 seasonalities and sudden changes in the market and in the competitors strategies.

In the second part of the thesis, we addressed the bid optimization problem and we proposed a novel framework that introduces the concept of *safety* for the algorithms choosing the bid allocation each day. More specifically, our approach aims at satisfying, with high probability, some daily ROI and spend constraints fixed by the business units of the companies. We model this setting as a constrained optimization problem. Furthermore, we proved that such a problem is inapproximable within any strict factor, unless $\mathsf{P} = \mathsf{NP}$, but it admits an exact pseudo-polynomial-time algorithm. Most interestingly, we demonstrated that no online learning algorithms can provide sublinear pseudo-regret while guaranteeing to be safe. We showed that the adaption of the GCB algorithm provides a sublinear pseudo-regret; however, it may violate the constraints a linear number of times. Thus, we design GCB$_{\mathtt{safe}}$, a new algorithm that guarantees safety at the cost of a loss in terms of revenue. Finally, we evaluate the empirical performance of the two algorithms on synthetically generated advertising problems. Remarkably, GCB$_{\mathtt{safe}}$ provides good performance in terms of safety, while suffering from a small loss w.r.t. GCB in terms of cumulative revenue.

As future works, an interesting open research direction is to explore how to define a less strict safety property, allowing the algorithm to overcome the impossibility result we showed. A new concept of safety may allow the constraints to hold only in expectation over the time horizon. Another option is to let the constraints change during the learning process, *e.g.,* become stricter over time. In the third part of this thesis, we addressed the targeting optimization problem. We designed a new method to define the optimal target of an advertising campaign, which exploits the information gathered from past interactions between a set of users and the advertising campaign to estimate the performance of all the possible sub-campaigns in the target space and select the campaign providing maximum revenue. We propose the TargOpt algorithm, which follows the risk-averse framework, to solve the optimization problem to explore the target space completely. Moreover, in the case the dimension of the target space is too large, we provided the A-TargOpt algorithm, which allows to iteratively expand the space we analyse, and two different heuristics to effectively explore the target space, thus providing an anytime version of the TargOpt algorithm. Finally, we showed on both synthetically generated and real-world datasets that the proposed algorithms increase the revenue gained from the advertising campaign.

An interesting future work is the study of a criterion to decide what is the optimal number of days T after which we run one of the proposed algorithms. Another challenging extension of this work is the inclusion of the target optimization procedure in an online learning framework.

In the last part of this thesis, we formalized, for the first time, the problem of optimizing an Internet advertising campaign with sub-campaigns interdependence. We presented the IDL algorithm that, given a set of past observations, models these interdependences and returns an optimal allocation of the bid/daily budget on the sub-campaigns maximizing the revenue. We analysed the properties of the IDL algorithm both theoretically, providing a bound on the total error, and empirically, showing that it provides revenues on synthetic datasets significantly better than other approaches that do not exploit sub-campaigns interdependencies.

In the future, we will extend IDL to an online framework, and we will extend the interdependency analysis, including other advertising channels.

List of Figures

List of Tables

Bibliography

[1] (2020). Digital 2020: Glocal digital overview.

[2] Accabi, G. M., Trovò, F., Nuara, A., Gatti, N., and Restelli, M. (2018a). When gaussian processes meet combinatorial bandits: GCB. In *European Workshop on Reinforcement Learning (EWRL)*, pages 1–11.

[3] Accabi, G. M., Trovò, F., Nuara, A., Gatti, N., and Restelli, M. (2018b). When gaussian processes meet combinatorial bandits: Gcb. In *European Workshop on Reinforcement Learning (EWRL)*.

[4] Alippi, C., Roveri, M., and Trovò, F. (2014). Learning causal dependencies to detect and diagnose faults in sensor networks. In *SSCI*, pages 34–41. IEEE.

[5] Amin, K., Kearns, M., Key, P., and Schwaighofer, A. (2012). Budget optimization for sponsored search: Censored learning in mdps. *arXiv preprint arXiv:1210.4847*.

[6] Amit Bhatnagar, P. P. (2001). Identifying locations for targeted advertising on the internet. *International Journal of Electronic Commerce*, 5(3):23–44.

[7] Archak, N., Mirrokni, V. S., and Muthukrishnan, S. (2010). Budget optimization for online advertising campaigns with carryover effects. In *Ad Auctions Workshop*.

[8] Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

[9] Bleier, A. and Eisenbeiss, M. (2015). Personalized online advertising effectiveness: The interplay of what, when, and where. *Marketing Science*, 34(5):669–688.

[10] Braun, M. and Moe, W. W. (2013). Online display advertising: Modeling the effects of multiple creatives and individual impression histories. *Marketing Science*, 32(5):753–767.

[11] Calderón, C. and Liu, L. (2003). The direction of causality between financial development and economic growth. *J DEV ECON*, 72(1):321–334.

[12] Castiglioni, M., Nuara, A., Romano, G., Trovò, F., Gatti, N., and Restelli, M. (2020). Safe online bid optimization with return-oninvestment constraints. In *Under Review at AAAI*.

[13] Cesa-Bianchi, N. and Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge University Press.

[14] Chen, W., Wang, Y., and Yuan, Y. (2013). Combinatorial multi-armed bandit: General framework and applications. In *Proceedings of the Internation Conference on Machine Learning (ICML)*, pages 151–159.

[15] Cheng, H. and Cantú-Paz, E. (2010). Personalized click prediction in sponsored search. In *Proceedings of the ACM conference on web search and data mining (WSDM)*, pages 351–360.

[16] Degenne, R. and Perchet, V. (2016). Combinatorial semi-bandit with known covariance. In *Proceedings of the Neural Information Processing Systems Conference (NeurIPS)*, pages 2972–2980.

[17] Dickey, D. A. and Fuller, W. A. (1979). Distribution of the estimators for autoregressive time series with a unit root. *J AM STAT ASSOC*, 74(366a):427–431.

[18] Ding, W., Qin, T., Zhang, X.-D., and Liu, T. (2013). Multi-armed bandit with budget constraint and variable costs. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 232–238.

[19] Dinner, I. M., Heerde, H. J. V., and Neslin, S. A. (2014). Driving online and offline sales: The cross-channel effects of traditional, online display, and paid search advertising. *J MARKETING RES*, 51(5):527–545.

[20] Ehrgott, M. (2005). *Multicriteria Optimization*. Springer-Verlag, Berlin, Heidelberg.

[21] Gasparini, M., Nuara, A., Trovò, F., Gatti, N., and Restelli, M. (2018). Targeting optimization for internet advertising by learning from logged bandit feedback. In *IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

[22] Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

[23] Geyik, S. C., A-Saxena, and Dasdan, A. (2014). Multi-touch attribution based budget allocation in online advertising. In *Proceedings of the International Workshop on Data Mining for Online Advertising (ADKDD)*, pages 1–9.

[24] Granger, C. W. (1969). Investigating causal relations by econometric models and cross-spectral methods. *ECONOMETRICA*, pages 424–438.

[25] Hoban, P. R. and Bucklin, R. E. (2015). Effects of internet display advertising in the purchase funnel: Model-based insights from a randomized field experiment. *J MARKETING RES*, 52(3):375–393.

[26] Howard, J. and Sheth, J. (1969). *The theory of buyer behavior*. Wiley, New York.

[27] IAB (2019). Iab internet advertising revenue report 2020, first six months results. `https://www.iab.com/wp-content/uploads/2020/05/FY19-IAB-Internet-Ad-Revenue-Report_Final.pdf`. Online; accessed 20 January 2020.

[28] Italia, E. M., Nuara, A., Trovò, F., Restelli, M., Gatti, N., and Dellavalle, E. (2017). Safe online bid optimization with return-on-investment constraints. In *AMEC*, pages 1–15.

[29] Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *The Multiple-Choice Knapsack Problem*, chapter 11, pages 317–347. Springer.

[30] 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.

[31] Kong, D., Fan, X., Shmakov, K., and Yang, J. (2018a). A combinational optimization approach for advertising budget allocation. In *Companion of the The Web Conference*, pages 53–54.

[32] Kong, D., Shmakov, K., and Yang, J. (2018b). Demystifying advertising campaign for cpa goal optimization. In *Companion Proceedings of the The Web Conference*, pages 83–84.

[33] Langheinrich, M., Nakamura, A., Abe, N., Kamba, T., and Koseki, Y. (1999). Unintrusive customization techniques for web advertising. *Computer Networks*, 31(11):1259–1272.

[34] Lee, K.-C., Jalali, A., and Dasdan, A. (2013). Real time bid optimization with smooth budget delivery in online advertising. In *Proceedings of the International Workshop on Data Mining for Online Advertising (ADKDD)*, pages 1–9.

[35] Lewis, R. and Nguyen, D. (2015). Display advertising's competitive spillovers to consumer search. *QME-QUANT MARK ECON*, 13(2):93–115.

[36] Li, P., Hawbani, A., et al. (2018). An efficient budget allocation algorithm for multi-channel advertising. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 886–891.

[37] Markakis, E. and Telelis, O. (2010). Discrete strategies in keyword auctions and their inefficiency for locally aware bidders. In *Proceedings of the Conference on Web and Internet Economics (WINE)*, pages 523–530.

[38] Mas-Colell, A., Whinston, M. D., Green, J. R., et al. (1995). *Microeconomic theory*, volume 1. Oxford university press New York.

[39] Multimedia, M. (2018). Mediamatic.

[40] Nuara, A., Sosio, N., Trovo, F., Zaccardi, M. C., Gatti, N., and Restelli, M. (2019). Dealing with interdependencies and uncertainty in multi-channel advertising campaigns optimization. In *The World Wide Web Conference*, pages 1376–1386.

[41] Nuara, A., Trovò, F., Gatti, N., and Restelli, M. (2018). A combinatorial-bandit algorithm for the online joint bid/budget optimization of pay-per-click advertising campaigns. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, pages 2379–2386.

[42] Nuara, A., Trovò, F., Gatti, N., and Restelli, M. (2020). Online joint bid/daily budget optimization of internet advertising campaigns.

[43] Ozcicek, O. and Douglas Mcmillin, W. (1999). Lag length selection in vector autoregressive models: symmetric and asymmetric lags. *APPL ECON*, 31(4):517–524.

[44] Perlich, C., Dalessandro, B., Raeder, T., Stitelman, O., and Provost, F. (2014). Machine learning for targeted display advertising: Transfer learning in action. *Machine learning*, 95(1):103–127.

[45] Provost, F., Dalessandro, B., Hook, R., Zhang, X., and Murray, A. (2009). Audience selection for on-line brand advertising: privacy-friendly social network targeting. In *SIGKDD*, pages 707–716.

[46] 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.

[47] Raeder, T., Stitelman, O., Dalessandro, B., Perlich, C., and Provost, F. (2012). Design principles of massive, robust prediction systems. In *Proceedings of the ACM conference on knowledge discovery and data mining (SIGKDD)*, pages 1357–1365.

[48] Rasmussen, C. E. and Williams, C. K. (2006). *Gaussian processes for machine learning*, volume 1. MIT Press.

[49] Sani, A., Lazaric, A., and Munos, R. (2012). Risk-aversion in multi-armed bandits. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, pages 3275–3283.

[50] Sankararaman, K. A. and Slivkins, A. (2018). Combinatorial semi-bandits with knapsacks. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1760–1770.

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

[52] Srinivas, N., Krause, A., Seeger, M., and Kakade, S. M. (2010). Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1015–1022.

[53] 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.

[54] 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.

[55] Toda, H. Y. and Yamamoto, T. (1995). Statistical inference in vector autoregressions with possibly integrated processes. *J ECONOMETRICS*, 66(1-2):225–250.

[56] Trovò, F., Paladino, S., Restelli, M., and Gatti, N. (2016). Budgeted multi-armed bandit in continuous action space. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 560–568.

[57] Urban, G. L., Liberali, G., MacDonald, E., Bordley, R., and Hauser, J. R. (2013). Morphing banner advertising. *Marketing Science*, 33(1):27–46.

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

[59] Wassily, H. (1963). Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30.

[60] Weinan, W., Rong, Y., Wang, J., Zhu, T., and Wang, X. (2016). Feedback control of real-time display advertising. In *Proceedings of the Web Search and Data Mining (WSDM)*, pages 407–416.

[61] Wu, D., Chen, X., Yang, X., Wang, H., Tan, Q., Zhang, X., Xu, J., and Gai, K. (2018). Budget constrained bidding by model-free reinforcement learning in display advertising. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1443–1451.

[62] Xia, Y., Li, H., Qin, T., Yu, N., and Liu, T.-Y. (2015). Thompson sampling for budgeted multi-armed bandits. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3960–3966.

[63] Yan, J., Liu, N., Wang, G., Zhang, W., Jiang, Y., and Chen, Z. (2009). How much can behavioral targeting help online advertising? In *WWW*, pages 261–270.

[64] Yang, X., Li, Y., Wang, H., Wu, D., Tan, Q., Xu, J., and Gai, K. (2019). Bid optimization by multivariable control in display advertising. In *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1966–1974.

[65] Zhang, W., Yuan, S., and Wang, J. (2014). Optimal real-time bidding for display advertising. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1077–1086.

[66] 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 *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1177–1185.

Appendices

APPENDIX $\mathcal{A}$

Online Bid/Budget Optimization

Proofs

[From [52]] Given the realization of a GP $f(\cdot)$, the estimates of the mean $\hat{\mu}_{t-1}(x)$ and variance $\hat{\sigma}^2_{t-1}(x)$ for the input x belonging to the input space X, for each $b \in \mathbb{R}^+$ the following condition holds:

$$\mathbb{P}\left(|f(x) - \hat{\mu}_{t-1}(x)| \geq \sqrt{b}\,\hat{\sigma}_{t-1}(x)\right) \leq e^{-\frac{b}{2}},$$

for each $x \in X$.

Proof. Be $r \sim \mathcal{N}(0,1)$ and $c \in \mathbb{R}^+$, we have:

$$\mathbb{P}[r > c] = \frac{1}{\sqrt{2\pi}} e^{-\frac{c^2}{2}} \int_c^\infty e^{-\frac{(r-c)^2}{2} - c(r-c)}\, dr \leq e^{-\frac{c^2}{2}}\, \mathbb{P}[r > 0] = \frac{1}{2} e^{-\frac{c^2}{2}},$$

since $e^{-c(r-c)} \leq 1$ for $r \geq c$. For the symmetry of the Gaussian distribution, we have:

$$\mathbb{P}[|r| > c] \leq e^{-\frac{c^2}{2}}.$$

Applying the above result to $r = \frac{f(x) - \hat{\mu}_{t-1}(x)}{\hat{\sigma}_{t-1}(x)}$ and $c = \sqrt{b}$ concludes the proof. $\square$

Theorem 1. *Let us consider an ABBA problem over T rounds where the functions $n_j(x,y)$ is the realization of a GP. Using the* AdComb-U-UCB *algorithm with the following upper bounds for the number of clicks and of value per click:*

$$\hat{u}^{(n)}_{j,t-1}(x,y) := \hat{\mu}_{j,t-1}(x,y) + \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(x,y),$$
$$\hat{u}^{(v)}_{j,t-1} := \hat{\nu}_{j,t-1} + \sqrt{b'_t}\,\hat{\psi}^2_{j,t-1},$$

respectively, with $b_t := 2\log\left(\frac{\pi^2 N M t^2}{3\delta}\right)$ *and* $b'_t := 2\log\left(\frac{\pi^2 N t^2}{3\delta}\right)$*. For every* $\delta \in (0,1)$*, the following holds with probability at least* $1-\delta$*:*

$$\mathcal{R}_T(\mathfrak{U}) \leq \Bigg\{ 8TNb_T \Bigg[\frac{v_{\max}^2}{\log\left(1+\frac{1}{\lambda}\right)} \sum_{j=1}^{N} \gamma_T(n_j) + \xi(n_{\max} + 2\sqrt{b'_t}\sigma)^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\psi_j^2} + T\right) \Bigg] \Bigg\}^{\frac{1}{2}},$$

where, λ *and* ξ *are variances of the measurement noise of the click functions* $n_j(\cdot)$ *and of the value per click* v_j*, respectively,* $v_{\max} := \max_{j\in\{1,\dots,N\}} v_j$ *is the maximum expected value per click,* $n_{\max} := \max_{x\in X, y\in Y, j\in\{1,\dots,N\}} n_j(x,y)$ *is the maximum expected number of click we might obtain on average over all the campaigns* C_j*, and* $\sigma^2 := k(\boldsymbol{a},\boldsymbol{a}) \geq \hat{\sigma}_{j,t}^2(\boldsymbol{a})$ *for each* j*,* t *and* $\boldsymbol{a}$*. Equivalently, with probability at least* $1-\delta$*, it holds:*

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN\sum_{j=1}^{N}\gamma_T(n_j)}\right),$$

where the notation $\tilde{O}(\cdot)$ *disregards the logarithmic factors.*

Proof. In AdComb-U-UCB, we assume the number of clicks $n_j(x,y) = n_j(\boldsymbol{a})$ of a campaign C_j be the realization of a GP over the space $\mathcal{D}$ of the bid/daily budget pairs $\boldsymbol{a} = (x,y)$. Using the selected input $\boldsymbol{a}_{j,h}$ and the corresponding observations $\tilde{n}_{j,h} = \tilde{n}_{j,h}(\boldsymbol{a}_{j,h})$ for each $h \in \{1,\dots,t-1\}$, the GP provides the estimates of the mean $\hat{\mu}_{j,t-1}(\boldsymbol{a})$ and variance $\hat{\sigma}_{j,t-1}^2(\boldsymbol{a})$ for each $\boldsymbol{a} \in \mathcal{D}$. The sampling phase is based on the upper bounds on the number of clicks and on the value per click, formally:

$$u_{j,t-1}^{(n)}(\mathbf{a}) := \hat{\mu}_{j,t-1}(\mathbf{a}) + \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\mathbf{a}), \tag{A.1}$$

$$u_{j.t-1}^{(v)} := \hat{\nu}_{j,t-1} + \sqrt{b'_t}\,\hat{\psi}_{j,t-1}. \tag{A.2}$$

Applying Lemma A.1 to Equation (A.1) for a generic arm $\mathbf{a}$ and $b = b_t$ we have:

$$\mathbb{P}\left[|n_j(\boldsymbol{a}) - \mu_{j,t-1}(\boldsymbol{a})| > \sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a})\right] \leq e^{-\frac{b_t}{2}}.$$

In the execution of the AdComb-U-UCB algorithm, after $t-1$ rounds, each arm can be chosen a number of times from 0 to $t-1$. Applying the union bound over the rounds ($t \in \{1,\dots,T\}$), the campaigns ($j \in \{1,\dots,N\}$) and the available arms in each campaign $\mathcal{D}$ ($\boldsymbol{a} \in \mathcal{D}$), and exploiting Lemma (A.1), we obtain:

$$\mathbb{P}\left[\bigcup_{t\in\{1,\dots,T\}}\bigcup_{j\in\{1,\dots,N\}}\bigcup_{\boldsymbol{a}\in\mathcal{D}}\Big(|n_j(\boldsymbol{a}) - \hat{\mu}_{j,t-1}(\boldsymbol{a})| > \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\boldsymbol{a})\Big)\right] \leq \sum_{t=1}^{T}\sum_{j=1}^{N} M e^{-\frac{b_t}{2}}.$$

Thus, choosing $b_t = 2\log\left(\frac{\pi^2 N M t^2}{3\delta}\right)$, we obtain:

$$\sum_{t=1}^{T}\sum_{j=1}^{N} M e^{-\frac{b_t}{2}} = \sum_{j=1}^{N}\sum_{t=1}^{T} M \frac{3\delta}{\pi^2 N M t^2} \leq \frac{\delta}{2}\frac{1}{N}\sum_{j=1}^{N}\left(\frac{6}{\pi^2}\sum_{t=1}^{\infty}\frac{1}{t^2}\right) = \frac{\delta}{2}.$$

Similarly, using Lemma A.1, we have:

$$\mathbb{P}\left[|v_j - \hat{\nu}_{j,t-1}| > \sqrt{b'_t}\,\hat{\psi}_{j,t-1}\right] \leq e^{-\frac{b'_t}{2}},$$

which holds for each $j \in \{1, \ldots, N\}$. Choosing $b'_t = 2\log\left(\frac{\pi^2 N t^2}{3\delta}\right)$ and applying an union bound we have:

$$\mathbb{P}\left[\bigcup_{t\in\{1,\ldots,T\}}\bigcup_{j\in\{1,\ldots,N\}}\left(|v_j - \hat{\nu}_{j,t-1}| > \sqrt{b'_t}\,\hat{\psi}_{j,t-1}\right)\right]$$
$$\leq \sum_{j=1}^{N}\sum_{t=1}^{T} e^{-\frac{b_t}{2}} = \sum_{j=1}^{N}\sum_{t=1}^{T}\frac{3\delta}{\pi^2 N t^2} \leq \frac{\delta}{2}.$$

Therefore, the event that at least one of the upper bounds over the number of clicks and the value per click does not hold has probability less than δ.

Assume to be in the event that all the previous bounds hold. The instantaneous pseudo-regret reg_t at round t satisfies the following inequality:

$$reg_t = r^*_{\boldsymbol{\mu}} - r_{\boldsymbol{\mu}}(S_t) \leq r^*_{\boldsymbol{\mu}} - r_{\bar{\boldsymbol{\mu}}_t}(S_t) + r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t),$$

where $\bar{\boldsymbol{\mu}}_t := (u^{(v)}_{1,t-1}u^{(n)}_{1,t-1}(\mathbf{a}_1), \ldots, u^{(v)}_{N,t-1}u^{(n)}_{N,t-1}(\mathbf{a}_M))$ is the vector composed of all the upper bounds of the different arms (of dimension NM). Let us recall that, given a generic superarm S, if all the elements of a vector $\boldsymbol{\mu}$ are larger than the ones of $\boldsymbol{\mu}'$ the following holds:

$$r_{\boldsymbol{\mu}}(S) \geq r_{\boldsymbol{\mu}'}(S).$$

Let us focus on the term $r_{\bar{\boldsymbol{\mu}}_t}(S_t)$. The following inequality holds:

$$r_{\bar{\boldsymbol{\mu}}_t}(S_t) \geq r^*_{\bar{\boldsymbol{\mu}}_t} \geq r_{\bar{\boldsymbol{\mu}}_t}(S^*_{\boldsymbol{\mu}}) \geq r_{\boldsymbol{\mu}}(S^*_{\boldsymbol{\mu}}) = r^*_{\boldsymbol{\mu}}, \tag{A.3}$$

where $S^*_{\boldsymbol{\mu}} \in \arg\max_{S\in\mathcal{S}}(r_{\boldsymbol{\mu}}(S))$ is the super-arm providing the optimum expected reward when the expected rewards are $\boldsymbol{\mu}$. Thus, we have:

$$\begin{aligned} reg_t &\leq r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t) \\ &\leq r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}_t}(S_t) + r_{\boldsymbol{\mu}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t), \end{aligned}$$

where $\boldsymbol{\mu}_t := (\hat{\eta}_{1,t-1}\hat{\mu}_{1,t-1}(\mathbf{a}_1), \ldots, \hat{\eta}_{N,t-1}\hat{\mu}_{N,t-1}(\mathbf{a}_M))$ is the vector composed of the estimated average payoffs for each arm $\boldsymbol{a} \in \mathcal{D}$.

A bound the terms $(r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}_t}(S_t))$ is provided by the definition of the upper confidence bounds:

$$
\begin{aligned}
&r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}_t}(S_t) = \sum_{j=1}^{N} \left[u^{(v)}_{j,t-1} u^{(n)}_{j,t-1}(\mathbf{a}_{j,t}) - \hat{\nu}_{j,t-1}\hat{\mu}_{j,t-1}(\mathbf{a}_{j,t}) \right] \\
&= \sum_{j=1}^{N} \left[\hat{\nu}_{j,t-1}\sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\mathbf{a}_{j,t}) + \hat{\mu}_{j,t-1}(\mathbf{a}_{j,t})\sqrt{b'_t}\,\hat{\psi}_{j,t-1} + \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\mathbf{a}_{j,t})\sqrt{b'_t}\,\hat{\psi}_{j,t-1} \right] \\
&\leq \sum_{j=1}^{N} \left\{ \left[v_j + \sqrt{b'_t}\,\hat{\psi}_{j,t-1} \right] \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\mathbf{a}_{j,t}) \right. \\
&\left. + \left[n_j(\mathbf{a}_{j,t}) + \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\mathbf{a}_{j,t}) \right] \sqrt{b'_t}\,\hat{\psi}_{j,t-1} + \sqrt{b_t}\,\hat{\sigma}_{j,t-1}(\mathbf{a}_{j,t})\sqrt{b'_t}\,\hat{\phi}_{j,t-1} \right\} \\
&\leq \sum_{j=1}^{N} \left[v_{\max}\sqrt{b_t} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) + n_{\max}\sqrt{b'_t}\,\hat{\psi}_{j,t-1} + 3\sqrt{b_t b'_t}\,\hat{\psi}_{j,t-1} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) \right] \\
&\leq v_{\max}\sqrt{b_t} \sum_{j=1}^{N} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) + n_{\max}\sqrt{b_t} \sum_{j=1}^{N} \hat{\psi}_{j,t-1} + 3\sqrt{b_t b'_t} \sum_{j=1}^{N} \hat{\psi}_{j,t-1} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) \\
&\leq v_{\max}\sqrt{b_t} \sum_{j=1}^{N} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) + (n_{\max}\sqrt{b_t} + 3\sqrt{b_t b'_t}\sigma) \sum_{j=1}^{N} \hat{\psi}_{j,t-1},
\end{aligned}
$$

where $\mathbf{a}_{j,t}$ is the arm chosen for campaign C_j in the superarm S_t, $v_{\max} := \max_{j\in\{1,\dots,N\}} v_j$ is the maximum expected value per click, $n_{\max} := \max_{j,\mathbf{a}} n_j(\mathbf{a})$ is the maximum expected number of clicks for any campaign. In the above derivation we used that $\sigma^2_{j,t}(\boldsymbol{a}) \leq k(\boldsymbol{a},\boldsymbol{a}) =: \sigma^2$ for each j, t and $\boldsymbol{a}$.

Let us focus on the term $(r_{\boldsymbol{\mu}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t))$:

$$
\begin{aligned}
&r_{\boldsymbol{\mu}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t) = \sum_{j=1}^{N} [\hat{\nu}_{j,t-1}\hat{\mu}_{j,t-1}(\mathbf{a}_{j,t}) - v_j n_j(\mathbf{a}_{j,t})] \\
&= \sum_{j=1}^{N} [\hat{\nu}_{j,t-1}\hat{\mu}_{j,t-1}(\mathbf{a}_{j,t}) - \hat{\nu}_{j,t-1} n_j(\mathbf{a}_{j,t}) + \hat{\nu}_{j,t-1} n_j(\mathbf{a}_{j,t}) - v_j n_j(\mathbf{a}_{j,t})] \\
&\leq \sum_{j=1}^{N} \left[(v_j + \sqrt{b'_t}\,\hat{\psi}_{j,t-1})(\hat{\mu}_{j,t-1}(\mathbf{a}_{j,t}) - n_j(\mathbf{a}_{j,t})) + n_j(\mathbf{a}_{j,t})(\hat{\nu}_{j,t-1} - v_j) \right] \\
&\leq \sum_{j=1}^{N} (v_{\max} + \sqrt{b'_t}\hat{\psi}_{j,t-1})\sqrt{b_t} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) + n_{\max}\sqrt{b'_t} \sum_{j=1}^{N} \hat{\psi}_{j,t-1} \\
&\leq v_{\max}\sqrt{b_t} \sum_{j=1}^{N} \max_{\boldsymbol{a}\in\mathcal{D}} \hat{\sigma}_{j,t-1}(\boldsymbol{a}) + (n_{\max}\sqrt{b_t} + \sqrt{b_t b'_t}\sigma) \sum_{j=1}^{N} \hat{\psi}_{j,t-1},
\end{aligned}
$$

where we used arguments similar to the ones considered in the previous derivation.

Overall, summing up the two terms, we have:

$$\begin{aligned}
reg_t &\leq v_{\max}\sqrt{b_t}\sum_{j=1}^{N}\max_{\boldsymbol{a}\in\mathcal{D}}\hat{\sigma}_{j,t-1}(\boldsymbol{a}) + (n_{\max}\sqrt{b_t} + 3\sqrt{b_t b_t'}\sigma)\sum_{j=1}^{N}\hat{\psi}_{j,t-1} \\
&+ v_{\max}\sqrt{b_t}\sum_{j=1}^{N}\max_{\boldsymbol{a}\in\mathcal{D}}\hat{\sigma}_{j,t-1}(\boldsymbol{a}) + (n_{\max}\sqrt{b_t} + \sqrt{b_t b_t'}\sigma)\sum_{j=1}^{N}\hat{\psi}_{j,t-1} \\
&= 2\sqrt{b_t}\left[v_{\max}\sum_{j=1}^{N}\max_{\boldsymbol{a}\in\mathcal{D}}\hat{\sigma}_{j,t-1}(\boldsymbol{a}) + (n_{\max} + 2\sqrt{b_t'}\sigma)\sum_{j=1}^{N}\hat{\psi}_{j,t-1}\right].
\end{aligned}$$

We need now to upper bound $\hat{\sigma}_{i,t-1}(\boldsymbol{a})$ and $\hat{\psi}_{j,t-1}$. Recall that, thanks to Lemma 5.3 in [52], under the Gaussian assumption we can express the information gain provided by the observations $\boldsymbol{n}_{t-1} = (\tilde{n}_{j,1}, \dots, \tilde{n}_{j,t-1})$ corresponding to the sequence of arms $(\boldsymbol{a}_{j,1}, \dots, \boldsymbol{a}_{j,t-1})$ as:

$$IG(\boldsymbol{n}_{t-1} \,|\, n_j) = \frac{1}{2}\sum_{h=1}^{t-1}\log\left(1 + \frac{\hat{\sigma}^2_{j,h}(\boldsymbol{a}_{j,h})}{\lambda}\right).$$

Since b_h is non-decreasing in h, we can write:

$$\sigma^2_{j,h}(\boldsymbol{a}_{j,h}) = \lambda\left[\frac{\hat{\sigma}^2_{j,h}(\boldsymbol{a}_{j,h})}{\lambda}\right] \leq \frac{\log\left(1 + \frac{\hat{\sigma}^2_{j,h}(\boldsymbol{a}_{j,h})}{\lambda}\right)}{\log\left(1 + \frac{1}{\lambda}\right)}, \tag{A.4}$$

since $s^2 \leq \frac{\log(1+s^2)}{\lambda\log\left(1+\frac{1}{\lambda}\right)}$ for all $s \in [0, \frac{1}{\lambda}]$, and $\frac{\hat{\sigma}^2_{j,h}(\boldsymbol{a}_{j,h})}{\lambda} \leq \frac{k(\boldsymbol{a}_{j,h}, \boldsymbol{a}_{j,h})}{\lambda} \leq \frac{1}{\lambda}$.

Using the definition of $\hat{\psi}_{j,t-1}$ we have:

$$\sum_{h=1}^{t-1}\hat{\psi}^2_{j,t-1} = \sum_{h=1}^{t-1}\frac{\psi_j^2\xi}{\xi + (h-1)\psi_j^2} \leq \xi\log\left(\frac{\xi}{\psi_j^2} + t\right).$$

Since Equation (A.4) holds for any $\boldsymbol{a} \in \mathcal{D}$, then it also holds for the arm $\boldsymbol{a}_{\max}$ maximizing the variance $\sigma^2_{j,h}(\boldsymbol{a}_{j,h})$ in n_j defined over $\mathcal{D}$. Thus, using the Cauchy-Schwarz inequality, we obtain:

$$
\begin{aligned}
\mathcal{R}_T^2(\mathfrak{U}) &\leq T \sum_{t=1}^{T} reg_t^2 \\
&\leq 4Tb_T \sum_{t=1}^{T} \left[2v_{\max}^2 \left(\sum_{j=1}^{N} \max_{\boldsymbol{a} \in \mathcal{D}_j} \sigma_{j,t-1}(\boldsymbol{a}) \right)^2 + 2(n_{\max} + 2\sqrt{b_t'}\sigma)^2 \left(\sum_{j=1}^{N} \hat{\psi}_{j,t-1} \right)^2 \right] \\
&\leq 8Tb_T \left\{ \sum_{t=1}^{T} \left[v_{\max}^2 N \sum_{j=1}^{N} \max_{\boldsymbol{a} \in \mathcal{D}} \hat{\sigma}_{j,t-1}^2(\boldsymbol{a}) \right] + \sum_{t=1}^{T} \left[(n_{\max} + 2\sqrt{b_t'}\sigma)^2 N \sum_{j=1}^{N} \hat{\psi}_{j,t-1}^2 \right] \right\} \\
&\leq 8TNb_T \left\{ v_{\max}^2 \sum_{j=1}^{N} \sum_{t=1}^{T} \left[\max_{\boldsymbol{a} \in \mathcal{D}} \frac{\log\left(1 + \frac{\hat{\sigma}_{i,n-1}^2(\boldsymbol{a})}{\lambda}\right)}{\log\left(1 + \frac{1}{\lambda}\right)} \right] \right. \\
&\left. + (n_{\max} + 2\sqrt{b_t'}\sigma)^2 \sum_{j=1}^{N} \sum_{t=1}^{T} \hat{\psi}_{j,t-1}^2 \right\} \\
&\leq 8TNb_T \left\{ \frac{v_{\max}^2}{\log\left(1 + \frac{1}{\lambda}\right)} \sum_{j=1}^{N} \underbrace{\sum_{t=1}^{T} \max_{\boldsymbol{a} \in \mathcal{D}} \log\left(1 + \frac{\hat{\sigma}_{i,n-1}^2(\boldsymbol{a})}{\lambda}\right)}_{= \gamma_T(n_j)} \right. \\
&\left. + \xi(n_{\max} + 2\sqrt{b_t'}\sigma)^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\psi_j^2} + T\right) \right\} \\
&\leq 8TNb_T \left[\frac{v_{\max}^2}{\log\left(1 + \frac{1}{\lambda}\right)} \sum_{j=1}^{N} \gamma_T(n_j) + \xi(n_{\max} + 2\sqrt{b_t'}\sigma)^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\psi_j^2} + T\right) \right].
\end{aligned}
$$

We conclude the proof by taking the square root on both the r.h.s. and the l.h.s. of the last inequality.

□

Theorem 2. *Let us consider an ABBA problem over T rounds where the functions $n_j(x, y)$ is the realization of a GP. Using the AdComb-U-TS algorithm, for every $\delta \in (0, 1)$, the following holds with probability at least $1 - \delta$:*

$$
\begin{aligned}
\mathcal{R}_T(\mathfrak{U}) \leq \Bigg\{ 8TN \Bigg[& \frac{v_{\max}^2}{\log\left(1 + \frac{1}{\lambda}\right)} b_T \sum_{j=1}^{N} \gamma_T(n_j) \\
& + \xi b_T' (n_{\max} + \sqrt{b_T}\sigma)^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\psi_j^2} + T\right) \Bigg] \Bigg\}^{1/2},
\end{aligned}
$$

where $b_t := 8 \log\left(\frac{2NMt^2}{3\delta}\right)$, $b_t' := 8 \log\left(\frac{2Nt^2}{3\delta}\right)$, λ and ξ are variances of the measurement noise of the click functions $n_j(\cdot)$ and of the value per click v_j, respectively, $v_{\max} := \max_{j \in \{1,\dots,N\}} v_j$ is the maximum expected value per click, $n_{\max} := \max_{x \in X, y \in Y, j \in \{1,\dots,N\}} n_j(x, y)$ is the maximum expected number of click we might obtain on average over all the campaigns C_j, and $\sigma^2 := k(\boldsymbol{a}, \boldsymbol{a}) \geq \hat{\sigma}_{j,t}^2(\boldsymbol{a})$ for each j, t and $\boldsymbol{a}$.

Equivalently, with probability at least $1 - \delta$, it holds:

$$
\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN \sum_{j=1}^{N} \gamma_T(n_j)} \right).
$$

Proof. Recall that in AdComb-2D-TS we assume the number of clicks $n_j(x, y) = n_j(\boldsymbol{a})$ of a campaign C_j is the realization of a GP over the space $\mathcal{D}$ of the bid/budget pairs $\boldsymbol{a} = (x, y)$. Using the selected input $\boldsymbol{a}_{j,h}$ and corresponding observations $\tilde{n}_{j,h} = \tilde{n}_{j,h}(\boldsymbol{a}_{j,h})$ for each $h \in \{1, \ldots, t-1\}$ the GP provides us with the estimates of the mean $\hat{\mu}_{j,t-1}(\boldsymbol{a})$ and variance $\hat{\sigma}^2_{j,t-1}(\boldsymbol{a})$ for each $\boldsymbol{a} \in \mathcal{D}$. The sampling phase generates the following two values for the number of clicks and on the value per click, formally, for each campaign C_j, a sample $\theta^{(n)}_{j,t-1}(\mathbf{a})$ is extracted from $\mathcal{N}(\hat{\mu}_{j,t-1}(\mathbf{a}), \hat{\sigma}^2_{j,t-1}(\mathbf{a}))$ for the number of clicks, and a sample $\theta^{(v)}_{j,t-1}$ is extracted from $\mathcal{N}(\hat{\nu}_{j,t-1}, \hat{\psi}_{j,t-1})$.

Let us focus on $\theta^{(n)}_{j,t-1}(\mathbf{a})$. Since Lemma A.1 also applies to univariate Gaussian distributions, it holds for $\theta^{(n)}_{j,t-1}(\mathbf{a})$, for a generic arm $\boldsymbol{a}$, and, formally, we have:

$$\mathbb{P}\left[|\theta^{(n)}_{j,t-1}(\boldsymbol{a}) - \hat{\mu}_{j,t-1}(\boldsymbol{a})| > \sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a})\right] \le e^{-\frac{b_t}{2}},$$

for each $b_t > 0$. By relying on the triangle inequality, fora each $\boldsymbol{a} \in \mathcal{D}$ we have:

$$\begin{aligned}
&\mathbb{P}\left[|\theta^{(n)}_{j,t-1}(\boldsymbol{a}) - n_j(\boldsymbol{a})| > \sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a})\right] \\
&\le \mathbb{P}\left[|\theta^{(n)}_{j,t-1}(\boldsymbol{a}) - \hat{\mu}_{j,t-1}(\boldsymbol{a})| + |\hat{\mu}_{j,t-1}(\boldsymbol{a}) - n_j(\boldsymbol{a})| > \sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a})\right] \\
&\le \mathbb{P}\left[|\theta^{(n)}_{j,t-1}(\boldsymbol{a}) - \hat{\mu}_{j,t-1}(\boldsymbol{a})| > \frac{1}{2}\sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a})\right] \\
&+ \mathbb{P}\left[|\hat{\mu}_{j,t-1}(\boldsymbol{a}) - n_j(\boldsymbol{a})| > \frac{1}{2}\sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a})\right] \\
&\le 2e^{-\frac{b_t}{8}}.
\end{aligned}$$

Similarly to what done in the proof of Theorem 1, setting $b_t := 8 \log\left(\frac{2NMt^2}{3\delta}\right)$, applying the union bound over the rounds, the subsets $\mathcal{D}$, the number of times the arms are chosen in $\mathcal{D}$, and the available arms, we have that the following holds with probability at least $1 - \frac{\delta}{2}$:

$$|\theta^{(n)}_{j,h-1}(\boldsymbol{a}) - n_j(\boldsymbol{a})| < \sqrt{b_h}\hat{\sigma}_{j,h-1}(\boldsymbol{a}),$$

for all $\boldsymbol{a} \in \mathcal{D}_j$, $j \in \{1, \ldots N\}$ and $h \in \{1, \ldots, t\}$.

The same reasoning can be carried out with $\theta^{(v)}_{j,t-1}$ setting $b'_t := 8 \log\left(\frac{2Nt^2}{3\delta}\right)$, so that the following bound:

$$|\theta^{(v)}_{j,t-1} - v_j| < \sqrt{b'_t}\hat{\psi}_{j,t-1},$$

holds for each $j \in \{1, \ldots, N\}$ and $h \in \{1, \ldots, t\}$ with probability at least $1 - \frac{\delta}{2}$. Therefore, jointly, the bounds over the number of clicks and on the value per click hold with probability at least $1 - \delta$.

Let us assume that all previous bounds hold. Consider the instantaneous pseudo-regret reg_t at round t:

$$\begin{aligned}
reg_t &= r^*_{\boldsymbol{\mu}} - r_{\boldsymbol{\mu}}(S_t) \\
&= r^*_{\boldsymbol{\mu}} - r_{\boldsymbol{\theta_t}}(S^*_{\boldsymbol{\mu}}) + r_{\boldsymbol{\theta_t}}(S^*_{\boldsymbol{\mu}}) - r_{\boldsymbol{\theta_t}}(S_t) + r_{\boldsymbol{\theta_t}}(S_t) - r_{\boldsymbol{\mu}}(S_t) \\
&\le |r_{\boldsymbol{\mu}}(S^*_{\boldsymbol{\mu}}) - r_{\boldsymbol{\theta_t}}(S^*_{\boldsymbol{\mu}})| + |r_{\boldsymbol{\theta_t}}(S_t) - r_{\boldsymbol{\mu}}(S_t)|,
\end{aligned}$$

where $\boldsymbol{\theta_t} := (\theta^{(v)}_{1,t-1}\theta^{(n)}_{1,t-1}(\boldsymbol{a}_1), \ldots, \theta^{(v)}_{N,t-1}\theta^{(n)}_{N,t-1}(\boldsymbol{a}_M))$ is the vector of the drawn payoffs for the turn t and $r_{\boldsymbol{\theta_t}}(S^*_{\boldsymbol{\mu}}) - r_{\boldsymbol{\theta_t}}(S_t) \le 0$ for the fact that the chosen arm S_t maximize the reward assuming an expected reward over the arms of $\boldsymbol{\theta_t}$.

Let us focus on the term $|r_{\boldsymbol{\mu}}(S) - r_{\boldsymbol{\theta_t}}(S)|$ on a generic superarm $S = (\boldsymbol{a}_1, \dots \boldsymbol{a}_N)$:

$$\begin{aligned}
&|r_{\boldsymbol{\mu}}(S) - r_{\boldsymbol{\theta_t}}(S)| = \sum_{j=1}^{N} |v_j n_j(\mathbf{a}_j) - \theta^{(v)}_{j,t-1}\theta^{(n)}_{j,t-1}(\mathbf{a}_j)| \\
&= \sum_{j=1}^{N} |v_j n_j(\mathbf{a}_j) - \theta^{(v)}_{j,t-1} n_j(\mathbf{a}_j)| + |\theta^{(v)}_{j,t-1} n_j(\mathbf{a}_j) - \theta^{(v)}_{j,t-1}\theta^{(n)}_{j,t-1}(\mathbf{a}_j)| \\
&= \sum_{j=1}^{N} \left[n_{\max}\sqrt{b'_t}\hat{\psi}_{j,t-1} + \theta^{(v)}_{j,t-1}\sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a}_j) \right] \\
&= n_{\max}\sqrt{b'_t}\sum_{j=1}^{N}\hat{\psi}_{j,t-1} + \sum_{j=1}^{N}\left(v_j + \sqrt{b'_t}\hat{\psi}_{j,t-1}\right)\sqrt{b_t}\hat{\sigma}_{j,t-1}(\boldsymbol{a}_j) \\
&= n_{\max}\sqrt{b'_t}\sum_{j=1}^{N}\hat{\psi}_{j,t-1} + v_{\max}\sqrt{b_t}\sum_{j=1}^{N}\hat{\sigma}_{j,t-1}(\boldsymbol{a}_j) + \sqrt{b_t b'_t}\sigma\sum_{j=1}^{N}\hat{\psi}_{j,t-1} \\
&\leq v_{\max}\sqrt{b_t}\sum_{j=1}^{N}\max_{\boldsymbol{a}\in\mathcal{D}}\hat{\sigma}_{j,t-1}(\boldsymbol{a}) + \sqrt{b'_t}(n_{\max} + \sqrt{b_t}\sigma)\sum_{j=1}^{N}\hat{\psi}_{j,t-1},
\end{aligned}$$

and, therefore, the instantaneous regret reg_t is bounded by twice the quantity we derived.

The cumulative regret becomes:

$$\begin{aligned}
&\mathcal{R}^2_T(\mathfrak{U}) \leq T\sum_{t=1}^{T} reg_t^2 \\
&\leq 4T\sum_{t=1}^{T}\left[2v^2_{\max}b_t\left(\sum_{j=1}^{N}\max_{\boldsymbol{a}\in\mathcal{D}_j}\sigma_{j,t-1}(\boldsymbol{a})\right)^2 + 2b'_t(n_{\max} + \sqrt{b_t}\sigma)^2\left(\sum_{j=1}^{N}\hat{\psi}_{j,t-1}\right)^2\right] \\
&\leq 8T\left\{b_T\sum_{t=1}^{T}\left[v^2_{\max}N\sum_{j=1}^{N}\max_{\boldsymbol{a}\in\mathcal{D}}\hat{\sigma}^2_{j,t-1}(\boldsymbol{a})\right] + b'_T\sum_{t=1}^{T}\left[(n_{\max} + \sqrt{b_t}\sigma)^2 N\sum_{j=1}^{N}\hat{\psi}^2_{j,t-1}\right]\right\} \\
&\leq 8TN\left[\frac{v^2_{\max}}{\log\left(1+\frac{1}{\lambda}\right)}b_T\sum_{j=1}^{N}\gamma_T(n_j) + \xi b'_T(n_{\max} + \sqrt{b_T}\sigma)^2\sum_{j=1}^{N}\log\left(\frac{\xi}{\psi_j^2} + T\right)\right].
\end{aligned}$$

We conclude the proof by taking the square root on both the r.h.s. and the l.h.s. of the last inequality.

□

Theorem 3. *Let us consider an ABBA problem over T rounds where the functions $n^{\mathsf{sat}}_j(x)$ and $e^{\mathsf{sat}}_j(x)$ are the realization of GPs. Using the* AdComb-F-UCB *algorithm with the following upper bounds for the number of clicks, the number of clicks per unit of budget, and the value per click, respectively:*

$$\begin{aligned}
u^{(n)}_{j,t-1}(x) &:= \hat{\mu}_{j,t-1}(x) + \sqrt{b_t}\hat{\sigma}_{j,t-1}(x), \\
u^{(e)}_{j,t-1}(x) &:= \hat{\eta}_{j,t-1}(x) + \sqrt{b_t}\hat{s}_{j,t-1}(x), \\
u^{(v)}_{j,t-1} &:= \nu_{j,\hat{t}-1} + \sqrt{b'_t}\hat{\psi}_{j,t-1},
\end{aligned}$$

with $b_t = 2\log\left(\frac{\pi^2 N M t^2}{2\delta}\right)$ *and* $b'_t := 2\log\left(\frac{\pi^2 N t^2}{2\delta}\right)$*. For every $\delta \in (0,1)$, the following holds* *with probability at least* $1 - \delta$*:*

$$\mathcal{R}_T(\mathfrak{U}) \leq \left\{ TN \left[\bar{c}_1 b_T \sum_{j=1}^{N} \gamma_T(n_j) + \bar{c}_2 b_T \sum_{j=1}^{N} \gamma_T(e_j) \right.\right.$$
$$\left.\left. + \bar{c}_3 b'_T \left(2 s y_{\max} \sqrt{b_T} + 2\sigma \sqrt{b_T} + n^{\mathsf{sat}}_{\max} \right)^2 \sum_{j=1}^{N} \log \left(\frac{\xi}{\psi_j^2} + T \right) \right] \right\}^{1/2},$$

where $\bar{c}_1 := \frac{12 v_{\max}^2}{\log\left(1+\frac{1}{\lambda}\right)}$, $\bar{c}_2 := \frac{12 v_{\max}^2 y_{\max}^2}{\log\left(1+\frac{1}{\lambda'}\right)}$, *and* $\bar{c}_3 := 12\xi$, ξ, λ *and* λ' *are the variance of the value per click, measurement noise on the maximum number of clicks and number of clicks per unit of daily budget, respectively,* $v_{\max} := \max_{j \in \{1,\dots,N\}} v_j$ *is the maximum expected value per click,* $n_{\max} := \max_{x \in X, y \in Y, j \in \{1,\dots,N\}} n_j(x,y)$ *is the maximum expected number of click we might obtain on average over all the campaigns* C_j, $y_{\max} := \max_{y \in Y} y$ *is the maximum budget one can allocate on a campaign, and* $\sigma^2 := k(x,x) \geq \hat{\sigma}^2_{j,t}(x)$, $s^2 := k'(x,x) \geq \hat{s}^2_{j,t}(x)$ *for each* j, t *and* x.

Equivalently, with probability at least $1 - \delta$*, it holds:*

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN \sum_{j=1}^{N} [\gamma_T(n_j) + \gamma_T(e_j)]} \right).$$

Proof. At first notice that Lemma A.1 can be applied to the quantities of maximum number of clicks, maximum cost and value per click. This allows, setting $b_t := 2 \log \left(\frac{\pi^2 N M t^2}{2\delta} \right)$ and $b'_t := 2 \log \left(\frac{\pi^2 N t^2}{2\delta} \right)$, that the following bounds for each arm x and each round t hold at the same time:

$$|n^{\mathsf{sat}}_j(x) - \hat{\mu}_{j,t-1}(x)| \leq \sqrt{b_t} \hat{\sigma}_{j,t-1}(x),$$
$$|e^{\mathsf{sat}}_j(x) - \hat{\eta}_{j,t-1}(x)| \leq \sqrt{b_t} \hat{s}_{j,t-1}(x),$$
$$|v_j - \hat{\nu}_{j,t-1}| \leq \sqrt{b'_t} \hat{\psi}_{j,t-1},$$

with probability at least $1 - \delta$, since each one of the above events holds with probability at least $1 - \frac{\delta}{3}$.

Assume that the previous bounds hold and consider the following quantity:

$$\left| \min \left\{ n^{\mathsf{sat}}_j(x), y e^{\mathsf{sat}}_j(x) \right\} - \min \left\{ \hat{\mu}_{j,t-1}(x), y \hat{\eta}_{j,t-1}(x) \right\} \right|. \tag{A.5}$$

If we are able to provide a bound for this quantity, then following the proof of Theorem 1 it is possible to provide a bound on the regret of the AdComb-F-UCB algorithm.

Depending on the values of y, $e^{\mathsf{sat}}_j(x)$ and $\hat{\eta}_{j,t-1}(x)$ we can distinguish the following 4 cases:

Case 1: if $y c^{\mathsf{sat}}_j(x) > n^{\mathsf{sat}}_j(x) \wedge y \hat{\eta}_{j,t-1}(x) > \hat{\mu}_{j,t-1}(x)$ the quantity in Equation A.5 becomes:

$$\left| \min \left\{ n^{\mathsf{sat}}_j(x), y e^{\mathsf{sat}}_j(x) \right\} - \min \left\{ \hat{\mu}_{j,t-1}(x), y \hat{\eta}_{j,t-1}(x) \right\} \right| \tag{A.6}$$
$$\leq |n^{\mathsf{sat}}_j(x) - \hat{\mu}_{j,t-1}(x)| \leq \sqrt{b_t} \hat{\sigma}_{j,t-1}(x). \tag{A.7}$$

Case 2 $y c^{\mathsf{sat}}_j(x) < n^{\mathsf{sat}}_j(x) \wedge y \hat{\eta}_{j,t-1}(x) < \hat{\mu}_{j,t-1}(x)$ the quantity in Equation A.5 becomes:

$$\left| \min \left\{ n^{\mathsf{sat}}_j(x), y e^{\mathsf{sat}}_j(x) \right\} - \min \left\{ \hat{\mu}_{j,t-1}(x), y \hat{\eta}_{j,t-1}(x) \right\} \right| \tag{A.8}$$
$$= y \left| e^{\mathsf{sat}}_j(x) - \hat{\eta}_{j,t-1}(x) \right| \leq y_{\max} \sqrt{b_t} \hat{s}_{j,t-1}(x). \tag{A.9}$$

Case 3: $\frac{n_j^{\mathsf{sat}}(x)}{e_j^{\mathsf{sat}}(x)} < y < \frac{\hat{\mu}_{j,t-1}(x)}{\hat{\eta}_{j,t-1}(x)}$ the quantity in Equation A.5 becomes:

$$\left|\min\left\{n_j^{\mathsf{sat}}(x), ye_j^{\mathsf{sat}}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{j,t-1}(x)\right\}\right| \tag{A.10}$$
$$= \left|n_j^{\mathsf{sat}}(x) - y\hat{\eta}_{j,t-1}(x)\right| \tag{A.11}$$
$$\leq y\left|e_j^{\mathsf{sat}}(x) - \hat{\eta}_{j,t-1}(x)\right| \leq y_{\max}\sqrt{b_t}\hat{s}_{j,t-1}(x), \tag{A.12}$$

where we used that $n_j^{\mathsf{sat}}(x) \leq ye_j^{\mathsf{sat}}(x)$.
Case 4: $\frac{\hat{\mu}_{j,t-1}(x)}{\hat{\eta}_{j,t-1}(x)} < y < \frac{n_j^{\mathsf{sat}}(x)}{e_j^{\mathsf{sat}}(x)}$ the quantity in Equation A.5 becomes:

$$\left|\min\left\{n_j^{\mathsf{sat}}(x), ye_j^{\mathsf{sat}}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{j,t-1}(x)\right\}\right| \tag{A.13}$$
$$= \left|ye_j^{\mathsf{sat}}(x) - \hat{\mu}_{j,t-1}(x)\right| \tag{A.14}$$
$$\leq \left|n_j^{\mathsf{sat}}(x) - \hat{\mu}_{j,t-1}(x)\right| \leq \sqrt{b_t}\hat{\sigma}_{j,t-1}(x), \tag{A.15}$$

where we used that $n_j^{\mathsf{sat}}(x) \geq ye_j^{\mathsf{sat}}(x)$.
Overall we have that:

$$\left|\min\left\{n_j^{\mathsf{sat}}(x), ye_j^{\mathsf{sat}}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{j,t-1}(x)\right\}\right| \tag{A.16}$$
$$\leq \sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x)\right). \tag{A.17}$$

Similarly to what has been provided above, let us focus on the quantity:

$$\left|\min\left\{u_{j,t-1}^{(n)}(x), yu_{j,t-1}^{(e)}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\}\right|, \tag{A.18}$$

which can be bounded by looking at the same 4 different cases.
Case 1: if $yu_{j,t-1}^{(e)}(x) > u_{j,t-1}^{(n)}(x) \wedge y\hat{\eta}_{j,t-1}(x) > \hat{\mu}_{j,t-1}(x)$ the quantity in Equation A.18 becomes by definition:

$$\left|\min\left\{u_{j,t-1}^{(n)}(x), yu_{j,t-1}^{(e)}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\}\right| \tag{A.19}$$
$$= |u_{j,t-1}^{(n)}(x) - \hat{\mu}_{j,t-1}(x)| = \sqrt{b_t}\hat{\sigma}_{j,t-1}(x). \tag{A.20}$$

Case 2 $yu_{j,t-1}^{(e)}(x) < u_{j,t-1}^{(n)}(x) \wedge y\hat{\eta}_{j,t-1}(x) < \hat{\mu}_{j,t-1}(x)$ the quantity in Equation A.18 becomes:

$$\left|\min\left\{u_{j,t-1}^{(n)}(x), yu_{j,t-1}^{(e)}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{j,t-1}(x)\right\}\right| \tag{A.21}$$
$$= y\left|u_{j,t-1}^{(e)}(x) - \hat{\eta}_{1,t-1}(x)\right| \tag{A.22}$$
$$\leq y_{\max}\sqrt{b_t}\hat{s}_{j,t-1}(x). \tag{A.23}$$

Case 3: $\frac{u_{j,t-1}^{(n)}(x)}{u_{j,t-1}^{(e)}(x)} < y < \frac{\hat{\mu}_{j,t-1}(x)}{\hat{\eta}_{j,t-1}(x)}$ the quantity in Equation A.18 becomes:

$$\left|\min\left\{u_{j,t-1}^{(n)}(x), yu_{j,t-1}^{(e)}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\}\right| \tag{A.24}$$
$$= \left|u_{j,t-1}^{(n)}(x) - y\hat{\eta}_{j,t-1}(x)\right| \tag{A.25}$$
$$\leq y\left|u_{j,t-1}^{(e)}(x) - \hat{\eta}_{j,t-1}(x)\right| \tag{A.26}$$
$$\leq y_{\max}\sqrt{b_t}\hat{s}_{j,t-1}(x). \tag{A.27}$$

Case 4: $\frac{\hat{\mu}_{j,t-1}(x)}{\hat{\eta}_{j,t-1}(x)} < y < \frac{u^{(n)}_{j,t-1}(x)}{u^{(e)}_{j,t-1}(x)}$ the quantity in Equation A.18 becomes:

$$\left|\min\left\{u^{(n)}_{j,t-1}(x), yu^{(e)}_{j,t-1}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\}\right| \tag{A.28}$$
$$= \left|yu^{(e)}_{j,t-1}(x) - \hat{\mu}_{j,t-1}(x)\right| \tag{A.29}$$
$$\leq \left|u^{(n)}_{j,t-1}(x) - \hat{\mu}_{j,t-1}(x)\right| = \sqrt{b_t}\hat{\sigma}_{j,t-1}(x). \tag{A.30}$$

Overall we have the bound:

$$\left|\min\left\{u^{(n)}_{j,t-1}(x), yu^{(e)}_{j,t-1}(x)\right\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\}\right| \tag{A.31}$$
$$\leq \sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x)\right). \tag{A.32}$$

Since the definition of per round regret reg_t is the same as the one in Theorem 1, defining:

$$\bar{\boldsymbol{\mu}}_t := \Bigg(u^{(v)}_{1,t-1}\min\left\{u^{(n)}_{1,t-1}(x_1), y_1 u^{(e)}_{1,t-1}(x_1)\right\}, \ldots,$$
$$u^{(v)}_{N,t-1}\min\left\{u^{(n)}_{N,t-1}(x_M), y_M u^{(e)}_{N,t-1}(x_M)\right\}\Bigg),$$
$$\boldsymbol{\mu}_t := \Bigg(\hat{\nu}_{1,t-1}\min\left\{\hat{\mu}_{1,t-1}(x_1), y_1\hat{\eta}_{1,t-1}(x_1)\right\}, \ldots,$$
$$\hat{\nu}_{N,t-1}\min\left\{\hat{\mu}_{N,t-1}(x_M), y_M\hat{\eta}_{N,t-1}(x_M)\right\}\Bigg)$$
$$\boldsymbol{\mu} := \Bigg(v_1\ \min\left\{n_1^{\mathsf{sat}}(x_1), y_1 c_1^{\mathsf{sat}}(x_1)\right\}, \ldots,$$
$$v_N\ \min\left\{n_N^{\mathsf{sat}}(x_M), y_M c_N^{\mathsf{sat}}(x_M)\right\}\Bigg),$$

we have:

$$\begin{aligned} reg_t &\leq r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t) \\ &\leq r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}_t}(S_t) + r_{\boldsymbol{\mu}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t), \end{aligned}$$

A bound the terms $(r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}_t}(S_t))$ is provided by the definition of the upper confidence bounds:

$$
\begin{aligned}
& r_{\bar{\boldsymbol{\mu}}_t}(S_t) - r_{\boldsymbol{\mu}_t}(S_t) \\
&= \sum_{j=1}^{N} \left[u^{(v)}_{j,t-1} \min\left\{ u^{(n)}_{j,t-1}(x_{j,t}), y_{j,t} u^{(e)}_{j,t-1}(x_{j,t}) \right\} - \hat{\nu}_{j,t-1} \min\left\{ \hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t}) \right\} \right] \\
&= \sum_{j=1}^{N} \left[u^{(v)}_{j,t-1} \min\left\{ u^{(n)}_{j,t-1}(x_{j,t}), y_{j,t} u^{(e)}_{j,t-1}(x_{j,t}) \right\} - u^{(v)}_{j,t-1} \min\left\{ \hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t}) \right\} \right. \\
& \left. + u^{(v)}_{j,t-1} \min\left\{ \hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t}) \right\} - \hat{\nu}_{j,t-1} \min\left\{ \hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t}) \right\} \right] \\
&\leq \sum_{j=1}^{N} \left\{ \left[v_{\max} + 2\sqrt{b'_t}\hat{\psi}_{j,t-1} \right] \left(\min\left\{ u^{(n)}_{j,t-1}(x_{j,t}), y_{j,t} u^{(e)}_{j,t-1}(x_{j,t}) \right\} \right. \right. \\
& \left. - \min\left\{ \hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t}) \right\} \right) \\
& \left. + \left(u^{(v)}_{j,t-1} - \hat{\nu}_{j,t-1} \right) \left[n^{\mathsf{sat}}_{\max} + \sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x) \right) \right] \right\} \\
&\leq \sum_{j=1}^{N} \left\{ \left[v_{\max} + 2\sqrt{b'_t}\hat{\psi}_{j,t-1} \right] \sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x) \right) \right. \\
& \left. + \left(n^{\mathsf{sat}}_{\max} + \sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x) \right) \right) \sqrt{b'_t}\hat{\psi}_{j,t-1} \right\} \\
&= v_{\max}\, y_{\max}\sqrt{b_t} \sum_{j=1}^{N} \hat{s}_{j,t-1}(x) + v_{\max}\sqrt{b_t} \sum_{j=1}^{N} \hat{\sigma}_{j,t-1}(x) + 2\sqrt{b_t b'_t}(s y_{\max} + \sigma) \sum_{j=1}^{N} \hat{\psi}_{j,t-1} \\
& + n^{\mathsf{sat}}_{\max}\sqrt{b'_t} \sum_{j=1}^{N} \hat{\psi}_{j,t-1} + s y_{\max}\sqrt{b_t b'_t} \sum_{j=1}^{N} \hat{\psi}_{j,t-1} + \sigma\sqrt{b_t b'_t} \sum_{j=1}^{N} \hat{\psi}_{j,t-1} \\
&= v_{\max}\, y_{\max}\sqrt{b_t} \sum_{j=1}^{N} \hat{s}_{j,t-1}(x) + v_{\max}\sqrt{b_t} \sum_{j=1}^{N} \hat{\sigma}_{j,t-1}(x) + \left(3 s y_{\max}\sqrt{b_t b'_t} \right. \\
& \left. + n^{\mathsf{sat}}_{\max}\sqrt{b'_t} + 3\sigma\sqrt{b_t b'_t} \right) \sum_{j=1}^{N} \hat{\psi}_{j,t-1},
\end{aligned}
$$

where $v_{\max} := \max_{j=1}^{N} v_j$ is the maximum expected value per click, $n_{\max} := max_{j,x}\, n_j(x)$ is the maximum number of clicks for any campaign, and we have $\hat{\sigma}_{j,t}(x) \leq \sigma$ and $\hat{s}_{j,t}(x) \leq s$ for each j, t, and x.

Let us focus on the term $(r_{\boldsymbol{\mu}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t))$:

$$
\begin{aligned}
&r_{\boldsymbol{\mu}_t}(S_t) - r_{\boldsymbol{\mu}}(S_t) \\
&= \sum_{j=1}^{N} \left[\hat{\nu}_{j,t-1} \min\left\{\hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t})\right\} - v_j \min\left\{n_j^{\mathsf{sat}}(x_{j,t}), y_{j,t}e_j^{\mathsf{sat}}(x_{j,t})\right\}\right] \\
&= \sum_{j=1}^{N} \left[\hat{\nu}_{j,t-1} \min\left\{\hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t})\right\} - \hat{\nu}_{j,t-1} \min\left\{n_j^{\mathsf{sat}}(x_{j,t}), y_{j,t}e_j^{\mathsf{sat}}(x_{j,t})\right\}\right. \\
&\left.+\hat{\nu}_{j,t-1} \min\left\{n_j^{\mathsf{sat}}(x_{j,t}), y_{j,t}e_j^{\mathsf{sat}}(x_{j,t})\right\} - v_j \min\left\{n_j^{\mathsf{sat}}(x_{j,t}), y_{j,t}e_j^{\mathsf{sat}}(x_{j,t})\right\}\right] \\
&\leq \sum_{j=1}^{N} \left\{\left[v_{\max} + \sqrt{b_t}\hat{\psi}_{j,t-1}\right] \left(\min\left\{\hat{\mu}_{j,t-1}(x_{j,t}), y_{j,t}\hat{\eta}_{j,t-1}(x_{j,t})\right\}\right.\right. \\
&\left.\left.- \min\left\{n_j^{\mathsf{sat}}(x_{j,t}), y_{j,t}e_j^{\mathsf{sat}}(x_{j,t})\right\}\right) + (\hat{\nu}_{j,t-1} - v_{j,t-1})n_{\max}^{\mathsf{sat}}\right\} \\
&\leq \sum_{j=1}^{N} \left\{\left[v_{\max} + \sqrt{b'_t}\hat{\psi}_{j,t-1}\right] \sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x)\right) + n_{\max}^{\mathsf{sat}}\sqrt{b'_t}\hat{\psi}_{j,t-1}\right\} \\
&= v_{\max}\, y_{\max}\sqrt{b_t}\sum_{j=1}^{N}\hat{s}_{j,t-1}(x) + v_{\max}\sqrt{b_t}\sum_{j=1}^{N}\hat{\sigma}_{j,t-1}(x) \\
&+ \left(\sqrt{b_t b'_t}\sigma + s y_{\max}\sqrt{b_t b'_t} + n_{\max}^{\mathsf{sat}}\sqrt{b'_t}\right)\sum_{j=1}^{N}\hat{\psi}_{j,t-1}
\end{aligned}
$$

where we used arguments similar to the ones we considered in the previous derivation.

Summing up we have:

$$
\begin{aligned}
reg_t \leq 2v_{\max}\sqrt{b_t}\sum_{j=1}^{N}\hat{\sigma}_{j,t-1}(x) + 2v_{\max}\, y_{\max}\sqrt{b_t}\sum_{j=1}^{N}\hat{s}_{j,t-1}(x) \\
+ \sqrt{b'_t}\left(4s y_{\max}\sqrt{b_t} + 4\sigma\sqrt{b_t} + 2n_{\max}^{\mathsf{sat}}\right)\sum_{j=1}^{N}\hat{\psi}_{j,t-1},
\end{aligned}
$$

Using arguments similar to what has been used in Theorem 1 we have:

$$\begin{aligned}
\mathcal{R}_T^2(\mathfrak{U}) &\leq T \sum_{t=1}^{T} reg_t^2 \\
&= T \sum_{t=1}^{T} \Bigg[2 v_{\max} \sqrt{b_t} \sum_{j=1}^{N} \hat{\sigma}_{j,t-1}(x) + 2 v_{\max}\, y_{\max} \sqrt{b_t} \sum_{j=1}^{N} \hat{s}_{j,t-1}(x) \\
&+ \sqrt{b_t'} \left(4 s y_{\max} \sqrt{b_t} + 4 \sigma \sqrt{B_t} + 2 n_{\max}^{\mathsf{sat}} \right) \sum_{j=1}^{N} \hat{\psi}_{j,t-1} \Bigg]^2 \\
&\leq T \Bigg[12 v_{\max}^2 b_T N \sum_{j=1}^{N} \sum_{t=1}^{T} \max_x \hat{\sigma}_{j,t-1}^2(x) + 12 v_{\max}^2\, y_{\max}^2 b_T N \sum_{j=1}^{N} \sum_{t=1}^{T} \max_x \hat{s}_{j,t-1}^2(x) \\
&+ 3 b_T' N \left(4 s y_{\max} \sqrt{b_T} + 4 \sigma \sqrt{b_T} + 2 n_{\max}^{\mathsf{sat}} \right)^2 \sum_{j=1}^{N} \sum_{t=1}^{T} \hat{\psi}_{j,t-1}^2 \Bigg]^2 \\
& TN \Bigg[\bar{c}_1 b_T \sum_{j=1}^{N} \gamma_T(n_j) + \bar{c}_2 b_T \sum_{j=1}^{N} \gamma_T(e_j) \\
&+ \bar{c}_3 b_T' \left(2 s y_{\max} \sqrt{b_T} + 2 \sigma \sqrt{b_T} + n_{\max}^{\mathsf{sat}} \right)^2 \sum_{j=1}^{N} \log \left(\frac{\xi}{\psi_j^2} + T \right) \Bigg],
\end{aligned}$$

where we defined $\bar{c}_1 := \frac{12 v_{\max}^2}{\log\left(1+\frac{1}{\lambda}\right)}$, $\bar{c}_2 := \frac{12 v_{\max}^2 y_{\max}^2}{\log\left(1+\frac{1}{\lambda'}\right)}$, and $\bar{c}_3 := 12\xi$, where λ and λ' are the variance of the measurement noise on the maximum number of clicks and number of clicks per unit of daily budget. Taking the square root of both right and left hand side of this inequality concludes the proof.

□

Theorem 4. *Let us consider an ABBA problem over T rounds where the functions $n_j^{\mathsf{sat}}(x)$ and $e_j^{\mathsf{sat}}(x)$ are the realization of GPs. Using the* AdComb-F-TS *algorithm, for every $\delta \in (0,1)$, the following holds with probability at least $1-\delta$:*

$$\mathcal{R}_T(\mathfrak{U}) \leq \Bigg\{ TN \Bigg[\bar{c}_1 b_T \sum_{j=1}^{N} \gamma_T(n_j) + \bar{c}_2 b_T \sum_{j=1}^{N} \gamma_T(e_j) + \bar{c}_3 b_T' \left(2 s y_{\max} \sqrt{b_T} + 2 \sigma \sqrt{b_T} + n_{\max}^{\mathsf{sat}} \right)^2 \sum_{j=1}^{N} \log \left(\frac{\xi}{\psi_j^2} + T \right) \Bigg] \Bigg\}^{1/2},$$

where $b_t = 2\log\left(\frac{\pi^2 N M t^2}{2\delta}\right)$, $b_t' := 2\log\left(\frac{\pi^2 N t^2}{2\delta}\right)$, $\bar{c}_1 := \frac{48 v_{\max}^2}{\log\left(1+\frac{1}{\lambda}\right)}$, $\bar{c}_2 := \frac{48 v_{\max}^2 y_{\max}^2}{\log\left(1+\frac{1}{\lambda'}\right)}$, and $\bar{c}_3 := 12\xi$, ξ, λ and λ' are the variance of the value per click, measurement noise on the maximum number of clicks and number of clicks per unit of daily budget, respectively, $v_{\max} := \max_{j \in \{1,\dots,N\}} v_j$ is the maximum expected value per click, $n_{\max} := \max_{x \in X, y \in Y, j \in \{1,\dots,N\}} n_j(x,y)$ is the maximum expected number of click we might obtain on average over all the campaigns C_j, $y_{\max} := \max_{y \in Y} y$ is the maximum budget one can allocate on a campaign, and $\sigma^2 := k(x,x) \geq \hat{\sigma}_{j,t}^2(x)$, $s^2 := k'(x,x) \geq \hat{s}_{j,t}^2(x)$ for each j, t and x.

Equivalently, with probability at least $1-\delta$, it holds:

$$\mathcal{R}_T(\mathfrak{U}) = \tilde{O}\left(\sqrt{TN \sum_{j=1}^{N} [\gamma_T(n_j) + \gamma_T(e_j)]} \right).$$

Proof. We recall that the decision we take are based on samples drawn from these distributions:

$$\begin{aligned}
&\theta_{j,t-1}^{(n)}(x) \sim \mathcal{N}(\hat{\mu}_{j,t-1}(x), \hat{\sigma}_{j,t-1}^2(x)),\\
&\theta_{j,t-1}^{(e)}(x) \sim \mathcal{N}(\hat{\eta}_{j,t-1}(x), \hat{s}_{j,t-1}^2(x)),\\
&\theta_{j,t-1}^{(v)} \sim \mathcal{N}(\hat{\nu}_{j,t-1}, \hat{\psi}_{j,t-1}^2).
\end{aligned}$$

Ad we derived in the proof of Theorem 2, by extracting samples from the predictive distributions of the GP we have that:

$$\begin{aligned}
&\mathbb{P}\left[|\theta_{j,t}^{(n)}(x) - n_j^{\mathsf{sat}}(x)| > \sqrt{b_t}\hat{\sigma}_{j,t-1}(x)\right] \leq 2e^{-\frac{b_t}{8}},\\
&\mathbb{P}\left[|\theta_{j,t}^{(e)}(x) - e_j^{\mathsf{sat}}(x)| > \sqrt{b_t}\hat{s}_{j,t-1}(x)\right] \leq 2e^{-\frac{b_t}{8}},\\
&\mathbb{P}\left[|\theta_{j,t}^{(v)} - v_j| > \sqrt{b'_t}\hat{\psi}_{j,t-1}\right] \leq 2e^{-\frac{b'_t}{8}},
\end{aligned}$$

and, therefore, setting $b_t := 8\log\left(\frac{2NMt^2}{2\delta}\right)$ and $b'_t := 8\log\left(\frac{2Nt^2}{2\delta}\right)$ we have that the bounds over the rounds t, the different GPs j and the different arms x holds together with probability greater than $1-\delta$.

Using arguments similar to what has been used in Theorem 3, we obtain the following bound for a generic bid/budget pair (x, y):

$$\begin{aligned}
&|\min\{n_j^{\mathsf{sat}}(x), ye_j^{\mathsf{sat}}(x)\} - \min\{\theta_{j,t}^{(n)}(x_j), y_j\theta_{j,t}^{(e)}(x_j)\}|\\
&\leq |\min\{n_j^{\mathsf{sat}}(x), ye_j^{\mathsf{sat}}(x)\} - \min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\}|\\
&+ |\min\left\{\hat{\mu}_{j,t-1}(x), y\hat{\eta}_{1,t-1}(x)\right\} - \min\{\theta_{j,t}^{(n)}(x_j), y_j\theta_{j,t}^{(e)}(x_j)\}|\\
&\leq 2\sqrt{b_t}\left(y_{\max}\hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x)\right).
\end{aligned}$$

As in Theorem 2, the instantaneous pseudo-regret reg_t at round t is bounded as follows:

$$reg_t \leq |r_{\boldsymbol{\mu}}(S^*_{\boldsymbol{\mu}}) - r_{\boldsymbol{\theta_t}}(S^*_{\boldsymbol{\mu}})| + |r_{\boldsymbol{\theta_t}}(S_t) - r_{\boldsymbol{\mu}}(S_t)|,$$

where $\boldsymbol{\mu}$ us defined as in Theorem 3 and

$$\begin{aligned}
\boldsymbol{\theta_t} := (&\theta_{1,t}^{(v)}\min\{\theta_{1,t}^{(n)}(x_1), y_1\theta_{1,t}^{(e)}(x_1)\}, \ldots,\\
&\qquad\qquad \theta_{N,t}^{(v)}\min\{\theta_{N,t}^{(n)}(x_M), y_M\theta_{1,t}^{(e)}(x_M)\}
\end{aligned}$$

is the vector of the drawn payoffs for the turn t and $r_{\boldsymbol{\theta_t}}(S^*_{\boldsymbol{\mu}}) - r_{\boldsymbol{\theta_t}}(S_t) \leq 0$ for the fact that the chosen arm S_t maximize the reward assuming an expected reward over the arms of $\boldsymbol{\theta_t}$.

Let us focus on bounding the quantity $|r_{\boldsymbol{\mu}}(S) - r_{\boldsymbol{\theta_t}}(S)|$ on a generic superarm S:

$$
\begin{aligned}
&|r_{\boldsymbol{\mu}}(S) - r_{\boldsymbol{\theta_t}}(S)| = \\
&= \sum_{j=1}^{N} \Big[v_j \min\{n_j^{\mathsf{sat}}(x_j), y_j e_j^{\mathsf{sat}}(x_j)\} - \theta_{j,t}^{(v)} \min\{\theta_{j,t}^{(n)}(x_j), y_j \theta_{j,t}^{(e)}(x_j)\} \Big] \\
&= \sum_{j=1}^{N} \Big[v_j \min\{n_j^{\mathsf{sat}}(x_j), y_j e_j^{\mathsf{sat}}(x_j)\} - v_j \min\{\theta_{j,t}^{(n)}(x_j), y_j \theta_{j,t}^{(e)}(x_j)\} \\
&+ v_j \min\{\theta_{j,t}^{(n)}(x_j), y_j \theta_{j,t}^{(e)}(x_j)\} - \theta_{j,t}^{(v)} \min\{\theta_{j,t}^{(n)}(x_j), y_j \theta_{j,t}^{(e)}(x_j)\} \Big] \\
&\leq \sum_{j=1}^{N} \Big[2 v_{\max} \sqrt{b_t} \left(y_{\max} \hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x) \right) \\
&+ \sqrt{b'_t} \hat{\psi}_{j,t-1} \left(n_{\max}^{\mathsf{sat}} + 2\sqrt{b_t} \left(y_{\max} \hat{s}_{j,t-1}(x) + \hat{\sigma}_{j,t-1}(x) \right) \right) \Big] \\
&\leq 2 v_{\max} y_{\max} \sqrt{b_t} \sum_{j=1}^{N} \hat{s}_{j,t-1}(x) + 2 v_{\max} \sqrt{b_t} \sum_{j=1}^{N} \hat{\sigma}_{j,t-1}(x) + \\
&+ \sqrt{b'_t} (n_{\max}^{\mathsf{sat}} + 2 s y_{\max} \sqrt{b_t} + 2\sigma \sqrt{b_t}) \sum_{j=1}^{N} \hat{\psi}_{j,t-1})
\end{aligned}
$$

Thus, the the instantaneous pseudo-regret reg_t at round t is bounded as follows:

$$
\begin{aligned}
reg_t \leq 4 v_{\max} y_{\max} \sqrt{b_t} \sum_{j=1}^{N} \hat{s}_{j,t-1}(x) + 4 v_{\max} \sqrt{b_t} \sum_{j=1}^{N} \hat{\sigma}_{j,t-1}(x) + \\
+ 2\sqrt{b'_t} (n_{\max}^{\mathsf{sat}} + 2 s y_{\max} \sqrt{b_t} + 2\sigma \sqrt{b_t}) \sum_{j=1}^{N} \hat{\psi}_{j,t-1})
\end{aligned}
$$

Defining the constants $\bar{c}_1 := \frac{48 v_{\max}^2}{\log\left(1+\frac{1}{\lambda}\right)}$, $\bar{c}_2 := \frac{48 v_{\max}^2 y_{\max}^2}{\log\left(1+\frac{1}{\lambda'}\right)}$, and $\bar{c}_3 := 12\xi$ we have a cumulative regret of:

$$
\begin{aligned}
&\mathcal{R}_T^2(\mathfrak{U}) \leq T \sum_{t=1}^{T} reg_t^2 \\
&TN \Bigg[\bar{c}_1 b_T \sum_{j=1}^{N} \gamma_T(n_j) + \bar{c}_2 b_T \sum_{j=1}^{N} \gamma_T(e_j) \\
&+ \bar{c}_3 b'_T \left(2 s y_{\max} \sqrt{b_T} + 2\sigma \sqrt{b_T} + n_{\max}^{\mathsf{sat}} \right)^2 \sum_{j=1}^{N} \log\left(\frac{\xi}{\psi_j^2} + T \right) \Bigg],
\end{aligned}
$$

and taking the square root of both the left and right hand side of the inequality concludes the proof. □

APPENDIX $\mathcal{B}$

Safe Online Bid Optimization

Proofs

Theorem 5 (Inapproximability of the optimization problem)**.** *For any $\rho \in (0, 1]$, there is no polynomial-time algorithm returning a ρ-approximation to the problem in Equations* (6.1a)–(6.1c)*, unless* $\mathsf{P} = \mathsf{NP}$.

Proof. We restrict to the instances of SUBSET-SUM such that $z \leq \sum_{i \in S} u_i$. Solving these instances is trivially NP-hard, as any instance with $z > \sum_{i \in S} u_i$ is not satisfiable and we can decide it in polynomial time. Given an instance of SUBSET-SUM, let $l = \frac{\sum_{i \in S} u_i + 1}{\rho}$. Assume that for every sub-campaign C_j, the available bids are $\{0, 1\}$. Furthermore, there is a sub-campaign C_0 with $v_0 = 1$ and such that: $c_0(x) = 2l + z$ and $n_0(x) = l$ if $x = 1$, and $c_0(x) = 0$ and $n_0(x) = 0$ if $x = 0$. For every $i \in S$, we have a sub-campaign C_i with $v_i = 1$ and such that: $c_i(x) = u_i$ and $n_i(x) = u_i$ if $x = 1$, and $c_i(x) = 0$ and $n_i(x) = 0$ if $x = 0$. Set the daily budget $y_t = 2(z + l)$ and the ROI limit $\lambda = \frac{1}{2}$.

We show that, if SUBSET-SUM is satisfiable, then the corresponding instance of our problem admits a solution with a revenue larger than l, while, if SUBSET-SUM is not satisfiable, the maximum revenue is at most $\rho\, l - 1$. Thus, any approximation algorithm guaranteeing a ρ approximation factor would decide whether an instance of SUBSET-SUM is satisfiable.

If. Suppose SUBSET-SUM is satisfied by the set $S^* \subseteq S$ and that the solution assigns $x_i = 1$ if $i \in S^*$ and $x_i = 0$ otherwise, and it assigns $x_0 = 1$. The total revenue is $l + z \geq l$ and the constraints are satisfied. In particular, the sum of the costs is $2l + z + z = 2(l + z)$, while $\text{ROI} = \frac{l+z}{2l+2z} = \frac{1}{2}$.

Only if. Assume by contradiction that the instance of our problem admits a solution with a revenue strictly larger than $\rho\, l - 1$ and that SUBSET-SUM is not satisfiable. Then, it is easy to see that we need $x_0 = 1$ for campaign C_0 as the maximum achievable revenue is $\sum_{i \in S} u_i = \rho\, l - 1$ when $x_0 = 0$. Thus, since $x_0 = 1$, the budget constraint forces $\sum_{i \in S : x_i = 1} c_i(x_i) \leq z$, thus implying $\sum_{i \in S : x_i = 1} u_i \leq z$. By the satisfaction of the ROI constraint, *i.e.*, $\frac{\sum_{i \in S : x_i = 1} u_i + l}{\sum_{i \in S : x_i = 1} u_i + 2l + z} \geq \frac{1}{2}$, it must hold $\sum_{i \in S : x_i = 1} u_i \geq z$. Therefore, the set $S^* = \{i \in S : x_i = 1\}$ is a solution to SUBSET-SUM, thus reaching a contradiction. □

Theorem 6 (Optimality). *Subroutine* $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ *using* $\overline{w}_j(x) = \underline{w}_j(x) = v_j\, n_j(x)$ *and* $\overline{c}_j(x) = c_j(x)$ *for each* $j \in \{1, \ldots, N\}$ *returns the optimal solution to the problem in Equations* (6.1a)–(6.1c) *when the values of revenues and costs are in* R *and* Y*, respectively.*

Proof. Since all the possible values for the revenues and costs are accounted in the subroutine, the elements in $S(y, r)$ satisfy the two inequalities in Equation (6.2) with the equal sign. Therefore, all the elements in $S(y, r)$ would contribute to the computation of the final value of the ROI and budget constraints, *i.e.,* the ones after evaluating all the N sub-campaings, with the same values for revenue and costs, being their overall revenue equal to r and their overall cost equal to y. Notice that we do not have to discuss the violation of Constraint (6.1c) as long as we choose $\max(Y) = y_t$. The maximum operator in Line 12 excludes only solutions with the same pairs of revenue and costs and a lower revenue, therefore, the subroutine excludes only dominated solutions. The same reasoning holds also for the sub-campaign C_1 analysed by the algorithm. Finally, after all the dominated allocations have been discarded, the solution is selected by Equation (6.3), *i.e.*, among all the solutions satisfying the ROI constraints the one with the largest revenue is selected. □

Theorem 7 (Pseudo-regret/safety tradeoff). *For every* $\epsilon > 0$ *and time horizon* T*, there is no algorithm with pseudo-regret smaller than* $T(1/2 - \epsilon)$ *that violates the constraint on the budget less than* $T(1/2 - \epsilon)$ *times in expectation.*

Proof. As a first step, we show that an algorithm that satisfies the two conditions of the theorem can be used to distinguish between $\mathcal{N}(1, 1)$ and $\mathcal{N}(1 + \delta, 1)$ with arbitrary good probability using a number of samples independent from δ. Consider two instances of the bid optimization problem. Both instances have a single sub-campaign with $x \in \{0, 1\}$, $c(0) = 0$, $r(0) = 0$, $r(1) = 1$, $B = 1$. The first instance has cost $c^1(1) = \mathcal{N}(1, 1)$, while the second has $c^2(1) = \mathcal{N}(1 + \delta, 1)$. In the first setting, the algorithm must choose $x = 1$ at least $T(1/2 + \epsilon)$ times in expectation otherwise the pseudo-regret would be strictly greater than $T(1/2 - \epsilon)$, while in the second setting the algorithm must choose $x = 1$ at most than $T(1/2 - \epsilon)$ times in expectation otherwise the constraint on the budget would be violated strictly more than $T(1/2 - \epsilon)$ times. Standard concentration inequalities implies that for each $\gamma > 0$, there exists a $n(\epsilon, \gamma)$ such that, given $n(\epsilon, \gamma)$ executions of the learning algorithm, in the first setting $x = 1$ is played strictly more that $Tn(\epsilon, \gamma)/2$ times, while in the second setting strictly less than $Tn(\epsilon, \gamma)/2$ times. This implies that the learning algorithm can distinguish with arbitrary good success probability (independent of δ) between the two settings using (at most) $n(\epsilon, \gamma)T$ samples from one of the normal distributions. However, the Kullback-Leibler divergence between the two normal distributions is $KL(\mathcal{N}(1, 1), \mathcal{N}(1 + \delta, 1)) = \delta^2/2$ and each algorithm needs at least $\Omega(1/\delta^2)$ samples to distinguish between the two distributions with arbitrary good probability. Since δ can be arbitrary small, we have a contradiction. Thus, such an algorithm is not possible. □

Theorem 8 (GCB pesudo-regret). *Given* $\delta \in (0,\ 1)$*, the* GCB *algorithm applied to the problem in Equations* (6.1a)–(6.1c)*, with probability at least* $1 - \delta$*, suffers from a pseudo-regret of:*

$$R_T(\mathsf{GCB}) \leq \sqrt{\frac{16 T N^3 b_t}{\ln(1 + \sigma^2)} \sum_{j=1}^{N} \gamma_{j,T}},$$

where $\sigma \in \mathbb{R}^+$ and $\gamma_{j,T} \in \mathbb{R}^+$ are the noise standard deviation and the maximum information gain of a generic set of T samples for the GP modeling the number of clicks of sub-campaign C_j, respectively.

Proof. The bounds in Equations (6.4) and (6.5) guarantee that the probability that there is at least a triple (j, x, t) with $j \in N$, $x \in X_j$, $t \in \{1, \dots, T\}$ such that the actual value of $v_j n_j(x)$ is larger than the upper bound $\overline{w}_{j,t-1}(x) = \underline{w}_{j,t-1}(x)$ or the actual value of $c_j(x)$ is smaller than the lower bound $\overline{c}_{j,t-1}(x)$ is less than $\delta/2$ (see [3] for details). This implies, using a union bound, that the values in $\boldsymbol{\mu}$ used in the oracle $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ are statistical (optimistical) bounds for the true values with probability at least $1 - \delta$, as required by GCB. Then, the proof follows by applying Theorem 1 by [3] to our setting, using that $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ subroutine is an (α, β)-approximation algorithm with $\alpha = 1$ and $\beta = 1$. □

Theorem 9 (GCB safety)**.** *Given $\delta \in (0,\ 1)$, the probability that for at least a $t \in \{1, \dots, T\}$ the allocation returned by the GCB algorithm applied to the problem in Equations* (6.1a)–(6.1c) *violates at least one of the constraints is at least $1 - \frac{\delta}{2NT}$.*

Proof. Let us focus on a specific day t. Consider the case in which Constraints (6.1b) and (6.1c) are active, and, therefore, the left side equals the right side: $\sum_{j=1}^N \underline{w}_j(x_{j,t}) - \lambda^* \sum_{j=1}^N \overline{co}_j(x_{j,t}) = 0$ and $\sum_{j=1}^N \overline{co}_j(x_{j,t}) = y_t$. For the sake of simplicity we focus on the costs $\overline{co}_j(x_{j,t})$, but similar arguments also applies to the revenues $\underline{w}_j(x_{j,t})$. A necessary condition for which the two constraints are valid also for the real revenue and costs is that for at least one of the cost it holds $c_j(x_{j,t}) \leq \overline{c}_j(x_{j,t})$. Indeed, if the opposite holds, *i.e.*, $\overline{c}_j(x_{j,t}) > c_j(x_{j,t})$ for each $j \in \{1, \dots, N\}$ and $x_{j,t} \in X_j$, the budget constraint we would be violated by the allocation since $\sum_{j=1}^N co_j(x_{j,t}) > \sum_{j=1}^N \overline{co}_j(x_{j,t}) = y_t$. Since the event $c_j(x_{j,t}) \leq \overline{c}_j(x_{j,t})$ occurs with probability at most $\frac{\delta \pi^2}{12NTt^2}$, over the $t \in \mathbb{N}$ the probability that the constraints are not violated is at most $\frac{\delta}{2NT}$. □

Theorem 10 (GCB$_{\mathtt{safe}}$ safety)**.** *Given $\delta \in (0,\ 1)$, the GCB$_{\mathtt{safe}}$ algorithm applied to the problem in Equations* (6.1a)–(6.1c) *is δ-safe.*

Proof. Let us focus on a specific day t. Constraints (6.1b) and (6.1c) are satisfied by the solution of $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ for the properties of the optimization procedure. Thanks to the specific construction of the upper bounds we have that $c_j(x_{j,t}) \leq \overline{c}_j(x_{j,t})$ and $n_j(x_{j,t}) \geq \underline{n}_j(x_{j,t})$, each holding with probability at least $1 - \frac{\delta \pi^2}{12NTt^2}$. As a consequence, we have:

$$\frac{\sum_{j=1}^N v_j \; n_j(x_{j,t})}{\sum_{j=1}^N \; co_j(x_{j,t})} > \frac{\sum_{j=1}^N v_j \; \underline{n}_j(x_{j,t})}{\sum_{j=1}^N \; \overline{co}_j(x_{j,t})} \geq \lambda^*$$

and

$$\sum_{j=1}^N c_j(x_{j,t}) < \sum_{j=1}^N \overline{c}_j(x_{j,t}) \leq y_t.$$

Using a union bound on the probability that at least one of these bounds is violated, and summing up over the time horizon T and over the number of samples of each GP over the days provides that the probability that the constraints are violated is at most δ. This concludes the proof. □

Theorem 11 (GCB$_{\mathtt{safe}}$ pseudo-regret)**.** *Given $\delta \in (0,\ 1)$, the GCB$_{\mathtt{safe}}$ algorithm applied to the problem in Equations* (6.1a)–(6.1c) *problem suffers from a pseudo-regret $R_t(\text{GCB}_{\mathtt{safe}}) = \mathcal{O}(T)$.*

Proof. The optimal solution has at least one of the constraints which is active, *i.e.*, it has the left hand side equal to the right hand side. Assume that the optimal clairvoyant solution $\left\{x_j^*\right\}_{j=1}^N$ to the optimization problem has a value of the ROI equal to λ^*. We showed in the proof of Theorem 10 that for any allocation, with probability at least $1-\frac{\delta\pi^2}{6NTt^2}$, it holds that $\frac{\sum_{j=1}^N v_j\ n_j(x_{j,t})}{\sum_{j=1}^N\ co_j(x_{j,t})} > \frac{\sum_{j=1}^N v_j\ \underline{n}_j(x_{j,t})}{\sum_{j=1}^N\ \overline{co}_j(x_{j,t})}$. This is true also for the optimal clairvoyant solution $\left\{x_j^*\right\}_{j=1}^N$, for which $\lambda^* = \frac{\sum_{j=1}^N v_j\ n_j(x^*)}{\sum_{j=1}^N\ co_j(x^*)} > \frac{\sum_{j=1}^N v_j\ \underline{n}_j(x^*)}{\sum_{j=1}^N\ \overline{co}_j(x^*)}$, implying that the values used in the ROI constraint make this allocation not feasible for the $\mathsf{Opt}(\boldsymbol{\mu}, \lambda^*)$ procedure. As shown before, this happens with probability at least $1 - \frac{\delta\pi^2}{6NTt^2}$ at day t, and $1-\delta$ over the time horizon T. To conclude, with probability $1-\delta$, not depending on the time horizon T, we will not choose the optimal arm during the time horizon and, therefore, the regret of the algorithm cannot be sublinear. Notice that the same line of proof is also holding in the case the budget constraint is active, therefore, the previous result holds for each instance of the problem in Equations (6.1a)–(6.1c). $\square$

Additional Experiments

Parameters from Experiment #1

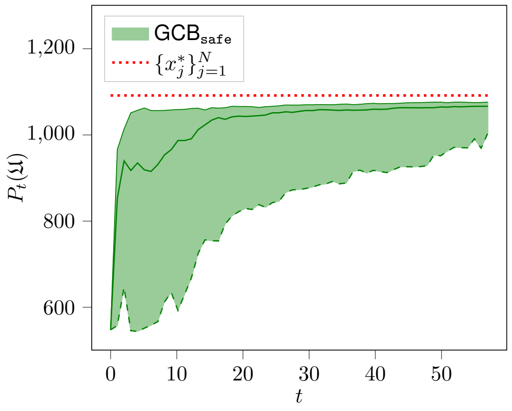
Table B.1 specifies the values of the parameters of cost and number-of-click functions of the sub-campaigns simulated in Experiment #1 and Experiment #2.

Table B.1: *Parameters of the synthetic settings used in Experiment #1 and Experiment #2.*

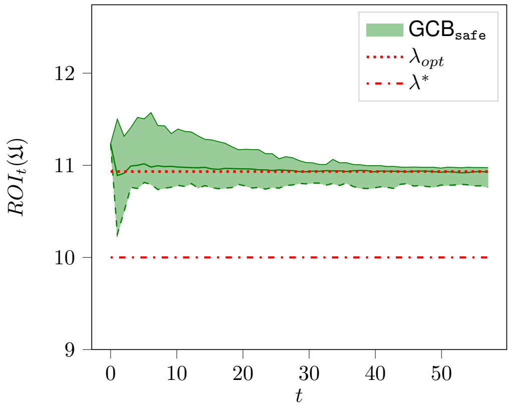
	C_1	C_2	C_3	C_4	C_5
β_j	60	77	75	65	70
δ_j	0.41	0.48	0.43	0.47	0.40
α_j	497	565	573	503	536
γ_j	0.65	0.62	0.67	0.68	0.69
σ_f GP revenue	0.669	0.499	0.761	0.619	0.582
l GP revenue	0.425	0.469	0.471	0.483	0.386
σ_f GP cost	0.311	0.443	0.316	0.349	0.418
l GP cost	0.76	0.719	0.562	0.722	0.727

Additional Figures Experiment #2

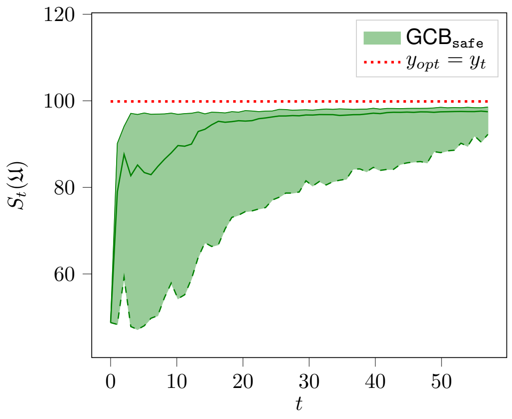
We report here the figures showing the 90% and 10% of the quantities analysed in the experimental section for Experiment #2.



(a)

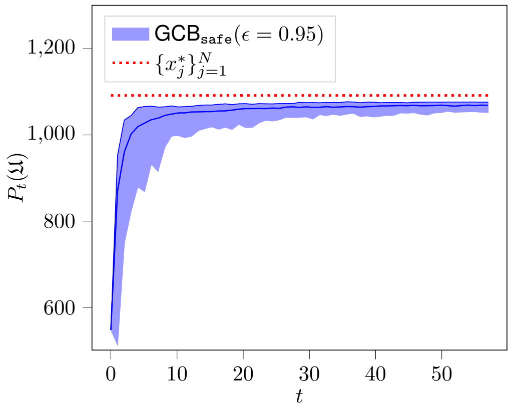


(b)

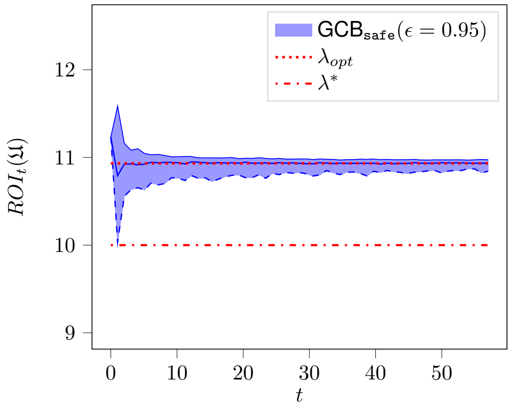


(c)

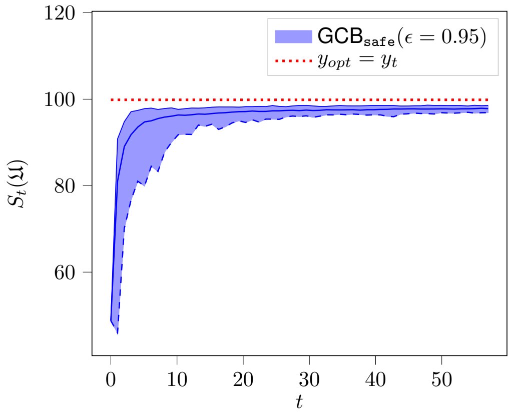
Figure B.1: *Results of Experiment #2: daily revenue (a), ROI (b), and spend (c) obtained by GCB$_{\mathtt{safe}}$. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*



(a)

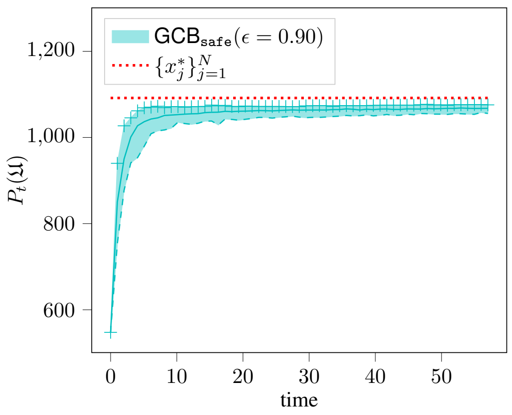


(b)

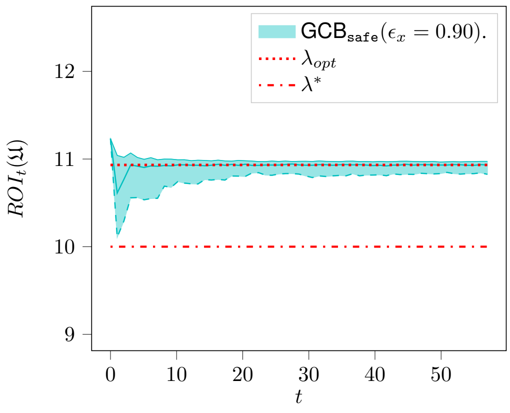


(c)

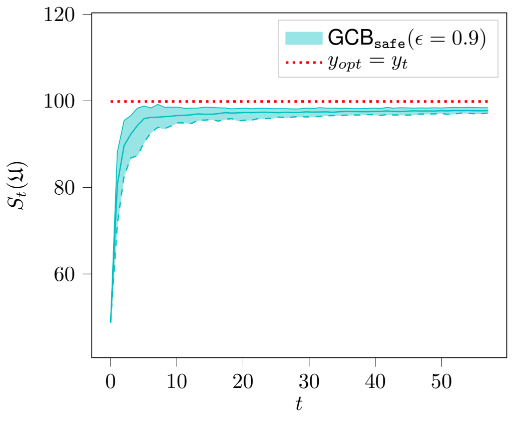
Figure B.2: *Results of Experiment #2: daily revenue (a), ROI (b), and spend (c) obtained by and* GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$*. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*



(a)

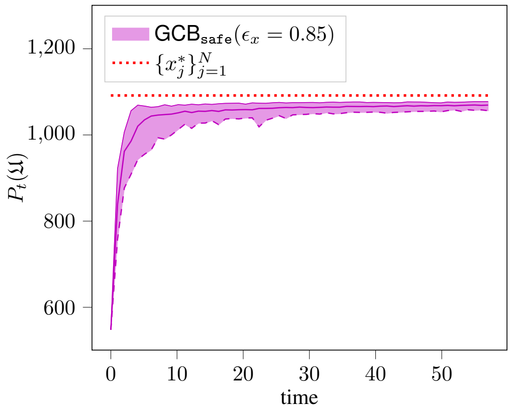


(b)

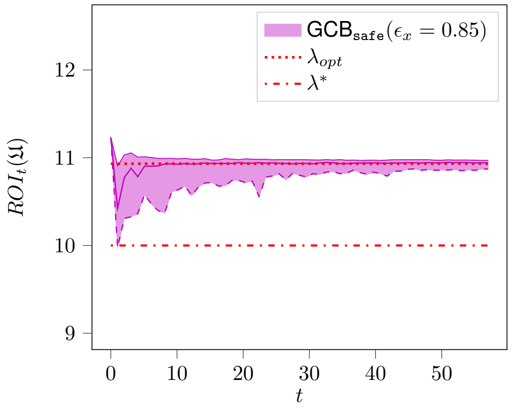


(c)

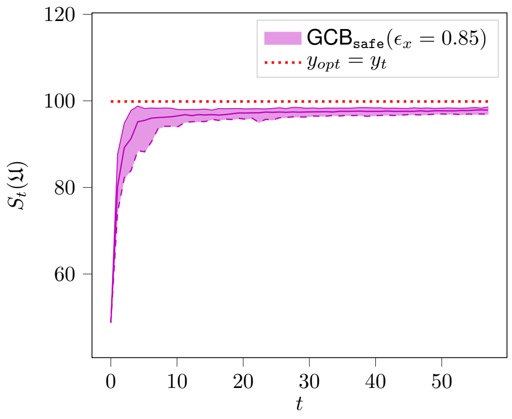
Figure B.3: *Results of Experiment #2: daily revenue (a), ROI (b), and spend (c) obtained by and GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*



(a)



(b)

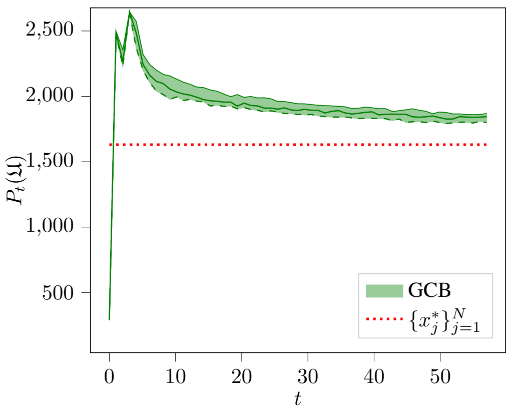


(c)

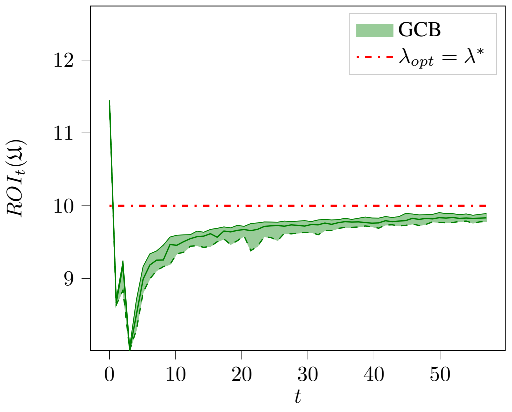
Figure B.4: *Results of Experiment #2: daily revenue (a), ROI (b), and spend (c) obtained by and* $\mathsf{GCB}_{\mathtt{safe}}(\epsilon_x = 0.85)$*. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*

Additional Figures Experiment #3

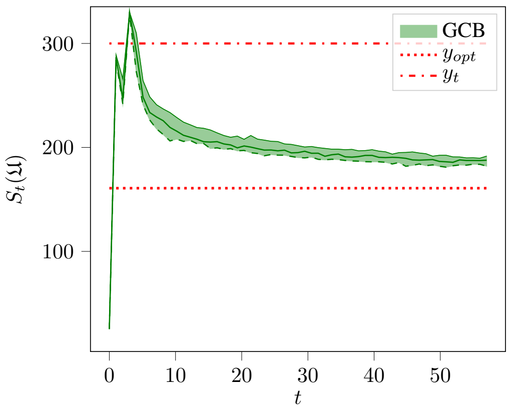
We report here the figures showing the 90% and 10% of the quantities analysed in the experimental section for Experiment #3.



(a)

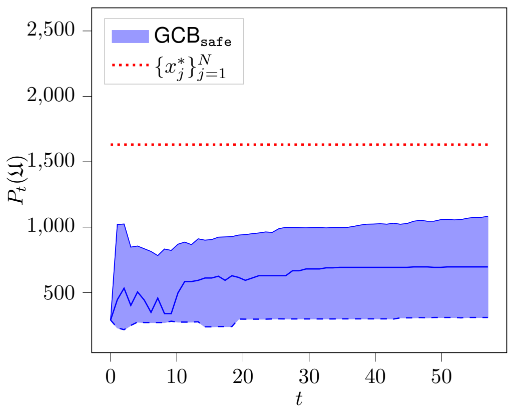


(b)

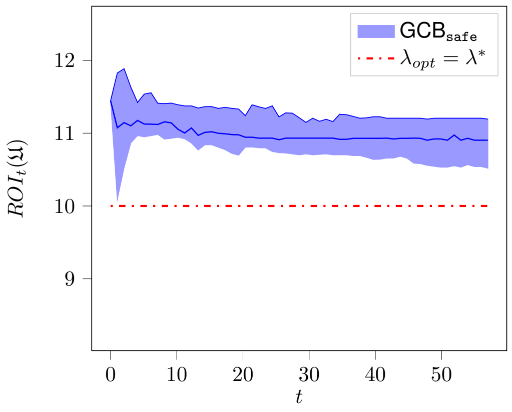


(c)

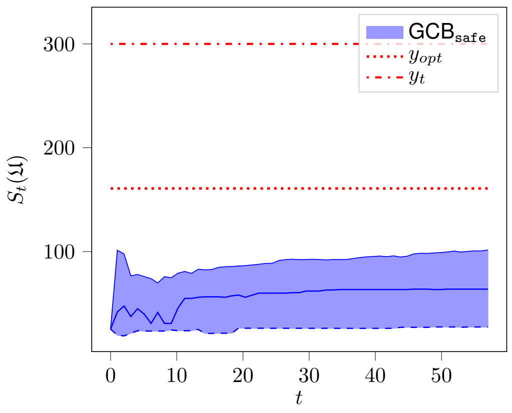
Figure B.5: *Results of Experiment #3: daily revenue (a), ROI (b), and spend (c) obtained by GCB in Experiment 3. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*



(a)

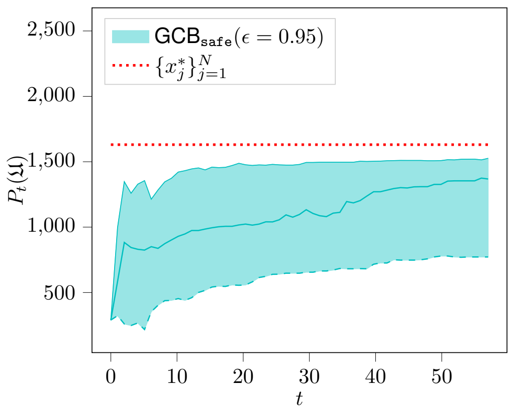


(b)

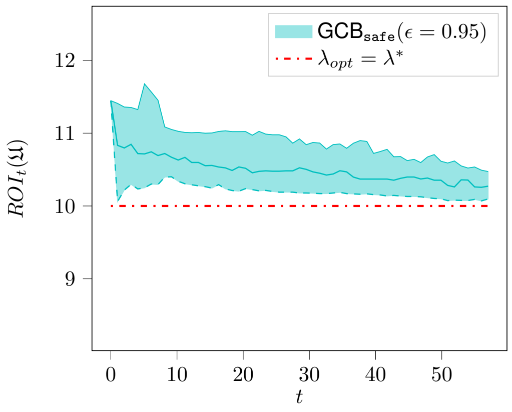


(c)

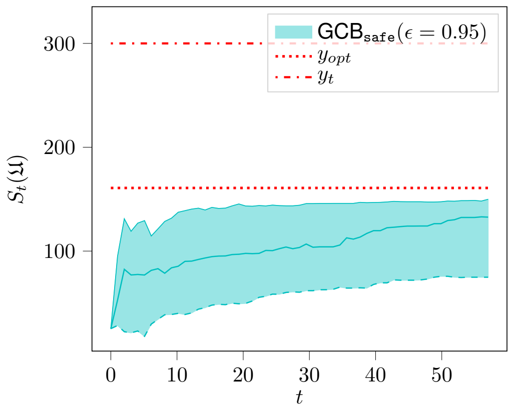
Figure B.6: *Results of Experiment #3: daily revenue (a), ROI (b), and spend (c) obtained by* GCB$_{\mathtt{safe}}$ *in Experiment 3. Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*



(a)



(b)



(c)

Figure B.7: *Results of Experiment #3: daily revenue (a), ROI (b), and spend (c) obtained by GCB($\epsilon_x = 0.95$). Dash-dotted lines correspond to the optimum values for the revenue and ROI, while dashed lines correspond to the values of the ROI and budget constraints.*

Experiment #4

In this experiment we extend the results of Experiment #1 and Experiment #2 to other settings. We simulate $N = 5$ sub-campaigns, with $|X| = 201$ bid values evenly spaced in $[0,\ 2]$, $|Y| = 101$ cost values evenly spaced in $[0,\ 100]$, and $|R|$ evenly spaced revenue values depending on the setting. We set a constant daily budget $y_t = 100$ for every t.

We build 10 scenarios that differ in the parameters defining the cost and revenue functions, and in the ROI parameter λ^*. Recall that the number-of-click functions coincides with the revenue functions since $v_j = 1$ for each $j \in \{1, \dots, N\}$. Parameters $\alpha_j \in \mathbb{N}^+$ and $\beta_j \in \mathbb{N}^+$ are sampled from discrete uniform distributions $\mathcal{U}\{50, 100\}$ and $\mathcal{U}\{400, 700\}$, respectively. Parameters γ_j and δ_j are sampled from the continuous uniform distributions $\mathcal{U}[0.2, 1.1)$. Finally, parameters λ^* are chosen so that the ROI constraint would be an active constraint for the original problem. Table B.2 summarize the values of such parameters.

Results Table B.3 reports the performances of algorithms GCB, GCB$_{\texttt{safe}}$, GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$ and GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$. In particular, $\mathbb{E}[\text{CR}_{t=\hat{t}}]$ is the cumulative revenue until day $\hat{t}$ averaged on the number of simulations, while $\sigma_{\text{CR}_{t=\hat{t}}}$ and $i_{th}^{t=\hat{t}}$p. are the corresponding standard deviation and i_{th} percentile, respectively. These results are reported w.r.t. two different time instant: $t = \lfloor \frac{T}{2} \rfloor = 28$, *i.e.,* at half of the period, and $t = T = 57$, *i.e.,* at the end of the time horizon. Finally, S_{ROI} and S_{budget} denotes the total number of days in which the ROI and the budget constraints were violated, respectively. In the last two columns we report the percentage of days on which the ROI and the budget constraint were violated, *i.e.,* $\frac{S_{ROI}}{T}$ and $\frac{S_{budget}}{T}$, respectively, averaged by the number of simulations. We performed 100 independent runs for each setting and each algorithm.

The results are in line with what have been observed in the main paper, showing that the GCB$_{\texttt{safe}}$ algorithm and its $\epsilon_x = 0.95$ variant are able not to violate the constraints with high probability, while GCB shows the worst performance in terms of constraints violations. In terms of cumulative revenue, the algorithms providing the largest values are the ones violating the constraint, while the algorithm showing the largest revenue while satisfying the problem constraints is GCB$_{\texttt{safe}}$ with $\epsilon_x = 0.95$. These results corroborates the idea that the relaxing the constraints for a small percentage (*e.g.,* 5%) provides a good tradeoff between revenue maximization and constraint satisfaction in most of the cases.

Table B.2: *Parameters characterizing the* 10 *different settings in Experiment #4.*

		C_1	C_2	C_3	C_4	C_5	λ^*
Setting 1	β_j	530	417	548	571	550	10.0
	δ_j	0.356	0.689	0.299	0.570	0.245	
	α_j	83	97	72	100	96	
	γ_j	0.939	0.856	0.484	0.661	0.246	
Setting 2	β_j	597	682	698	456	444	14.0
	δ_j	0.202	0.520	0.367	0.393	0.689	
	α_j	83	98	56	60	51	
	γ_j	0.224	0.849	0.726	0.559	0.783	
Setting 3	β_j	570	514	426	469	548	10.5
	δ_j	0.217	0.638	0.694	0.391	0.345	
	α_j	97	78	53	80	82	
	γ_j	0.225	0.680	1.051	0.412	0.918	
Setting 4	β_j	487	494	467	684	494	12.0
	δ_j	0.348	0.424	0.326	0.722	0.265	
	α_j	62	79	76	69	99	
	γ_j	0.460	1.021	0.515	0.894	1.056	
Setting 5	β_j	525	643	455	440	600	14.0
	δ_j	0.258	0.607	0.390	0.740	0.388	
	α_j	52	87	68	99	94	
	γ_j	0.723	0.834	1.054	1.071	0.943	
Setting 6	β_j	617	518	547	567	576	11.0
	δ_j	0.844	0.677	0.866	0.252	0.247	
	α_j	71	53	87	98	59	
	γ_j	0.875	0.841	1.070	0.631	0.288	
Setting 7	β_j	409	592	628	613	513	11.5
	δ_j	0.507	0.230	0.571	0.359	0.307	
	α_j	77	78	91	50	71	
	γ_j	0.810	0.246	0.774	0.516	0.379	
Setting 8	β_j	602	605	618	505	588	13.0
	δ_j	0.326	0.265	0.201	0.219	0.291	
	α_j	67	80	99	77	99	
	γ_j	0.671	0.775	0.440	0.310	0.405	
Setting 9	β_j	486	684	547	419	453	13.0
	δ_j	0.418	0.330	0.529	0.729	0.679	
	α_j	53	82	58	96	100	
	γ_j	0.618	0.863	0.669	0.866	0.831	
Setting 10	β_j	617	520	422	559	457	14.0
	δ_j	0.205	0.539	0.217	0.490	0.224	
	α_j	51	86	93	61	84	
	γ_j	1.0493	0.779	0.233	0.578	0.562	

Table B.3: *Performances of the GCB, GCB$_{\texttt{safe}}$, GCB$_{\texttt{safe}}$ $(\epsilon_x = 0.95)$, algsafe $(\epsilon_x = 0.90)$ algorithms in the 10 different settings in Experiment #4.*

		$\mathbb{E}\left[\mathrm{CR}_{t=\lfloor \frac{T}{2} \rfloor}\right]$	$\mathbb{E}[\mathrm{CR}_{t=T}]$	$\sigma_{\mathrm{CR}_{t=T}}$	$\sigma_{\mathrm{CR}_{t=\lfloor \frac{T}{2} \rfloor}}$	$50_{th}^{t=T}$p.	$50_{th}^{t=\lfloor \frac{T}{2} \rfloor}$p.	$90_{th}^{t=T}$p.	$90_{th}^{t=\lfloor \frac{T}{2} \rfloor}$p.	$10_{th}^{t=T}$p.	$10_{th}^{t=\lfloor \frac{T}{2} \rfloor}$ p.	$\mathbb{E}[\frac{S_{ROI}}{T}]$	$\mathbb{E}[\frac{S_{budget}}{T}]$
Setting 1	GCB	30767.336	57481.762	556.485	376.091	57497.185	30811.6042	58081.570	31239.227	56758.890	30288.910	1.000	0.617
	GCB$_{\texttt{safe}}$	21549.919	44419.090	4766.152	2474.262	45348.994	21972.152	46783.628	23163.449	42287.807	20324.750	0.019	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	23524.149	48028.035	4902.964	2487.586	48626.046	23831.680	50388.713	24827.723	46307.675	22506.989	0.208	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	25859.354	52327.738	829.526	611.281	52338.567	25887.959	53324.250	26605.853	51316.486	25104.946	0.938	0.002
Setting 2	GCB	35566.326	63664.204	1049.276	679.520	63701.943	35573.450	64984.421	36524.615	62249.768	34675.640	1.000	0.136
	GCB$_{\texttt{safe}}$	16290.759	34675.746	8541.501	4448.184	37028.783	17647.169	39594.699	19473.840	27748.425	11141.368	0.030	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	19564.274	40962.919	6013.044	3122.532	41823.542	20152.608	44468.575	21698.836	38640.207	17645.532	0.042	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	22099.617	46694.548	6382.710	3112.150	47749.883	22433.984	51564.023	24776.729	44099.097	19929.975	0.715	0.000
Setting 3	GCB	30213.282	54845.400	757.500	478.611	54816.940	30177.713	55734.991	30885.009	54006.505	29638.342	1.000	0.246
	GCB$_{\texttt{safe}}$	16577.752	35726.325	8239.174	4361.902	38302.025	18114.824	40746.921	19882.713	27279.764	8791.751	0.030	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	18370.540	38757.228	8492.878	4594.693	41422.689	19808.168	43337.069	21092.243	30413.268	12678.440	0.067	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	19993.574	42184.689	9652.822	5056.867	44820.418	21574.935	47659.265	23118.783	36570.721	14450.134	0.747	0.000
Setting 4	GCB	37383.516	71404.431	351.167	262.972	71399.582	37387.776	71877.052	37732.239	70930.123	37021.387	0.982	0.982
	GCB$_{\texttt{safe}}$	13817.172	29101.003	7052.947	3646.223	30992.413	14680.303	35602.210	17256.390	20509.269	9562.233	0.002	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	18270.458	39802.784	10232.989	4955.693	38296.818	17994.578	53375.436	24962.574	25197.830	11341.536	0.008	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	24095.853	51515.405	11094.352	5639.939	56621.600	24902.495	61992.866	30020.554	35642.967	16198.675	0.564	0.000
Setting 5	GCB	39523.308	74638.549	642.852	392.228	74693.882	39529.172	75405.263	40049.761	73756.512	39063.081	0.982	0.313
	GCB$_{\texttt{safe}}$	23230.524	48956.750	6715.177	3486.395	50021.031	23838.642	53644.831	26266.692	42946.297	19287.622	0.000	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	27003.316	56205.696	2578.603	1742.997	56554.329	27211.923	58839.509	28802.539	53278.885	24987.319	0.000	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	30207.289	63411.706	5636.641	2916.673	64364.864	30665.923	66764.410	32212.999	60519.219	28260.128	0.592	0.000
Setting 6	GCB	35775.780	67118.895	327.601	260.730	67130.819	35795.169	67536.732	36111.540	66726.744	35424.307	0.982	0.982
	GCB$_{\texttt{safe}}$	7707.891	14448.084	6006.165	3065.0127	15019.737	8075.216	18581.109	9800.874	6781.934	3926.408	0.022	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	7710.433	14968.966	6174.049	2974.836	15161.566	8157.195	20548.672	10351.782	7954.149	3860.675	0.024	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	15507.708	34716.555	16133.168	7280.176	37409.763	16601.218	55236.860	25366.308	9895.308	5188.600	0.189	0.000
Setting 7	GCB	35330.492	63038.578	873.627	401.328	63088.710	35367.147	64226.964	35793.889	61754.491	34823.374	1.000	0.408
	GCB$_{\texttt{safe}}$	14806.541	31662.450	5651.204	3090.871	33009.540	15570.468	35004.135	16922.489	28296.715	11338.293	0.037	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	17606.829	37744.715	4173.089	2619.873	38321.168	18161.518	41184.415	19805.806	33914.967	15276.547	0.031	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	20046.597	42528.584	7497.485	3624.829	43765.141	20683.409	47187.695	22301.568	38988.321	18314.219	0.696	0.000
Setting 8	GCB	42322.124	79571.510	476.880	375.810	79581.190	42317.922	80073.259	42743.107	78969.521	41913.760	1.000	0.982
	GCB$_{\texttt{safe}}$	22478.215	48046.987	11779.079	6000.067	52094.653	24180.975	57321.725	28024.487	30655.291	13338.036	0.020	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	27477.419	58450.174	10296.288	5605.913	61404.554	28845.740	66902.032	32883.702	41196.805	18222.303	0.021	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	33255.310	68252.406	3436.395	2417.770	68886.202	33857.704	70758.511	35377.143	65394.939	30696.023	0.069	0.000
Setting 9	GCB	37363.123	70280.744	672.557	347.550	70275.810	37352.860	71123.791	37811.648	69379.646	36942.376	1.000	0.339
	GCB$_{\texttt{safe}}$	18895.696	40116.370	5522.553	3047.034	40673.320	19357.076	43850.251	21161.005	37310.507	17222.155	0.028	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	23683.390	51138.961	3110.432	2036.069	50984.031	23375.652	54545.527	26174.860	47465.626	21385.349	0.031	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	29675.845	63574.004	3810.347	3323.916	64011.449	30112.566	66658.298	32559.645	60970.369	27280.237	0.795	0.000
Setting 10	GCB	41973.160	80570.847	435.533	344.547	80568.068	42019.380	81127.535	42388.800	80023.986	41496.669	1.000	0.982
	GCB$_{\texttt{safe}}$	28785.949	58965.259	3097.576	1465.835	60033.228	28917.795	62353.466	30535.428	54590.515	26931.753	0.018	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.95)$	31004.511	63685.460	3787.176	1876.417	65273.877	31550.496	67364.597	33105.511	57860.165	28349.753	0.019	0.000
	GCB$_{\texttt{safe}}(\epsilon_x = 0.90)$	33358.275	68480.403	4224.024	2181.878	70388.508	33998.870	72730.750	35838.227	61971.163	30317.769	0.652	0.000