A Practical Guide to Budget Pacing Algorithms in Digital Advertising

For Engineers

Yuanlong Chen

PREFACE

A typical real-time ad-serving funnel comprises ad targeting, conversion modeling (e.g., click-through rate prediction), budget pacing (bidding), and auction processes. While there is a wealth of research and articles on ad targeting and conversion modeling, budget pacing—a crucial component—lacks a systematic treatment specifically tailored for engineers in existing literature. This book aims to provide engineers with a practical yet relatively comprehensive introduction to budget pacing algorithms within the digital advertising domain. The book is structured as follows:

In Part I, we introduce foundational concepts in the digital advertising business, along with preliminary knowledge essential for understanding the subsequent chapters. We begin with a brief introduction to the history of digital advertising. Next, we cover some basics of programmatic ads, including the concepts of CPM, CPV, CPC, CPL and CPA ads. The entire ad-serving funnel is then briefly discussed to illustrate how ads are served in real time. The pipeline presented focuses on first-party ads (e.g., ads on Instagram or LinkedIn), though the serving pipeline for DSPs is similar. Additionally, we address basic optimization techniques, auction mechanism design, and other related preliminary topics that will be referenced throughout the book. Readers already familiar with these subjects may choose to skip this section and proceed directly to Part II.

In Part II, we discuss various pacing methods under standard second price auction. Two main bidding products, max delivery and cost cap, are used as examples to demonstrate the concepts of these pacing methods. Nevertheless, the underlying principles introduced here are applicable to other problems as well. We first provide a rigorous mathematical formulation of both the max delivery and cost cap problems. In the subsequent sections, we discuss various pacing algorithms commonly adopted in the industry, including throttling, PID controllers, MPC controllers and online adaptive optimal control. For each approach, we explain the motivation, introduce the basic background, and describe how it can be applied to bidding/pacing problems such as max delivery and cost cap. Additionally, we discuss the pros and cons of each approach, enabling readers to select the most suitable method for real-world applications based on their specific business needs. For some algorithms, pseudo-code and simple implementations are also provided to give readers a practical understanding of how to implement them in their daily work.

In Part III, we demonstrate how the pacing frameworks introduced in Part II can be applied to various other business scenarios. Topics include the initialization of campaign bids, bidding under different auction mechanisms (e.g., first-price auctions, where bid shading is required), bid optimization for multi-constraint problems (e.g., campaigns delivered across different placements or channels such as first-party and third-party platforms, or campaign groups where multiple campaigns share the same budget), deep funnel conversion problems (e.g., bid optimization for post-conversion events such as retention), common brand advertisements with reach and frequency requirements, and the over-delivery problem. Hopefully, these topics cover most of the tasks that a budget pacing engineer might encounter in their daily work.

This book is well-suited for engineers working on or interested in budget optimization and bidding algorithms in digital advertising. It is also valuable for engineers specializing in other aspects of ad serving, such as targeting and ranking, by providing insights into how downstream services in the serving funnel operate.While a basic understanding of mathematical optimization and control theory can be beneficial, it is not a prerequisite for reading this book. We also point out that the methodology discussed in this book primarily focuses on traditional control theory. For alternative approaches, such as those based on General Artificial Intelligence (GAI) methods (e.g., diffusion model-based bidding strategies), readers are encouraged to refer to the relevant research papers.

Budget optimization in digital advertising is a broad and complex topic. This little book primarily aims to provide engineers in the field with a comprehensive overview of the landscape of budget pacing algorithms. It does not attempt to cover every detail of budget pacing. For better readability, we omit some theoretical aspects, such as regret analysis and equilibrium analysis. Readers interested in these topics are encouraged to refer to the academic papers mentioned throughout the book for more in-depth information.

Y. Chen
Berkeley, California

CONTENTS

LIST OF ALGORITHMS

LIST OF FIGURES

LIST OF TABLES

DISCLAIMER

This book is intended solely for **educational, academic, and informational purposes**. The algorithms, methods, and technologies discussed herein, including those covered by active patents, are presented to provide a theoretical understanding and promote scholarly communication.

All content in this book has been derived from publicly available sources, including research papers, articles, and patents. The inclusion of these materials is for the purpose of **summarization, comparison, and educational discussion**. No proprietary or confidential information has been used in the creation of this book.

This book does not provide any authorization to reproduce, implement, or commercialize patented inventions or other proprietary technologies. Readers are advised that the use or implementation of patented algorithms or inventions may require prior permission from the respective patent holders.

The author(s) do not claim any ownership of third-party intellectual property discussed in this book. Every effort has been made to attribute work appropriately and respect intellectual property rights. If there are any inaccuracies or concerns regarding attribution, please contact the author(s) for correction.

Part I

Preliminaries

CHAPTER 1

BASICS OF DIGITAL ADVERTISING

In this chapter, we discuss the fundamentals of digital advertising. We begin with a brief history of digital advertising and how real-time bidding has transformed the landscape of this industry. Next, we explore the ad-serving pipeline, illustrating how ads are delivered in real time. Finally, we introduce two key campaign configurations—objective/optimization goal and charging model—and explain how different bidding products are designed based on these factors.

1 Brief History

Digital advertising has undergone a remarkable transformation since its inception, evolving from simple banner ads to sophisticated programmatic systems powered by real-time data. At the heart of this evolution is Real-Time Bidding (RTB), an innovation that has revolutionized the way advertisers and publishers interact. RTB operates alongside key players such as Demand-Side Platforms (DSPs), Supply-Side Platforms (SSPs), and major in-house bidding systems, such as Google's and Facebook's ad-serving platforms. In this chapter, we introduce the fundamentals of digital advertising to help readers familiarize themselves with key concepts in this domain.

The Early Days of Digital Advertising The first era of digital advertising began in the mid-1990s with the advent of the internet. Banner ads, such as the iconic AT&T ad on HotWired in 1994, marked the beginning of online monetization. During this period, advertisers purchased ad space directly from publishers, with limited data available to inform decisions.

As internet adoption grew, ad networks emerged to connect advertisers with publishers more efficiently. These networks aggregated inventory but lacked sophisticated targeting capabilities, leading to inefficiencies and limited personalization.

The Rise of Programmatic Advertising The introduction of programmatic advertising in the early 2000s addressed many of the shortcomings of traditional models. Automated systems replaced manual negotiations, enabling advertisers to target audiences based on demographic, geographic, and behavioral data. This innovation paved the way for the creation of DSPs and SSPs.

- **Demand-Side Platforms (DSPs):** DSPs provide advertisers with a centralized platform to manage and optimize ad campaigns across multiple channels. By leveraging advanced algorithms and real-time data, DSPs empower advertisers to bid on impressions that align with their target audience and campaign objectives.
- **Supply-Side Platforms (SSPs):** On the publisher side, SSPs enable the efficient management of ad inventory. SSPs connect publishers to multiple ad exchanges and DSPs, ensuring maximum revenue through competitive bidding. Together, DSPs and SSPs form the backbone of the programmatic advertising ecosystem.

The Emergence of Real-Time Bidding (RTB) RTB emerged in 2009 as a game-changer in programmatic advertising. Unlike earlier methods that involved bulk purchasing of ad space, RTB allows advertisers to bid on individual impressions in real time. Key milestones in RTB history include:

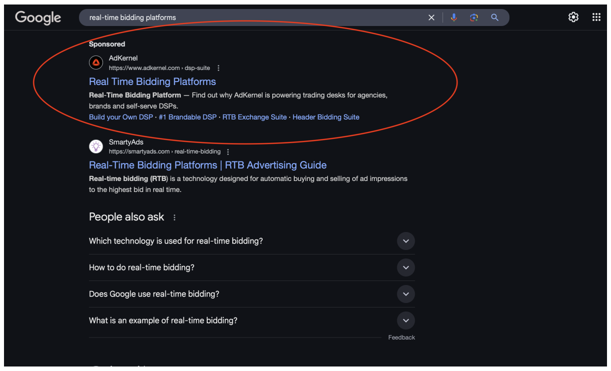
- **2009:** Google launched DoubleClick Ad Exchange, introducing the first large-scale RTB platform.
- **2011:** Facebook introduced Facebook Exchange (FBX), extending RTB capabilities to social media advertising.

- **2013:** Mobile RTB gained prominence, reflecting the rapid growth of mobile internet usage.

- **2015:** Header bidding strategies allowed publishers to maximize revenue by offering inventory to multiple exchanges simultaneously.

In-House Real-Time Bidding Systems Today, many large internet companies, such as Google, Meta(Facebook), Amazon, and LinkedIn, have developed their own in-house ad-serving systems to leverage unique data resources and monetize their vast user traffic. Advertisers can set up campaigns directly through these companies' ad management tools, and their ads are served to users across the companies' apps and websites.

In this scenario, these companies effectively function as both a DSP (Demand-Side Platform) and an SSP (Supply-Side Platform) simultaneously. When a user engages with a platform, ad campaigns can be displayed in various placements, such as news feeds or search result pages. By leveraging extensive in-house user data, these platforms can optimize ad delivery more efficiently, improving targeting accuracy and maximizing engagement.



Sponsored

AdKernel
https://www.adkernel.com › dsp-suite

Real Time Bidding Platforms

Real-Time Bidding Platform — Find out why AdKernel is powering trading desks for agencies, brands and self-serve DSPs.

Build your Own DSP · #1 Brandable DSP · RTB Exchange Suite · Header Bidding Suite

SmartyAds
https://smartyads.com › real-time-bidding

Real-Time Bidding Platforms | RTB Advertising Guide

Real-time bidding (RTB) is a technology designed for automatic buying and selling of ad impressions to the highest bid in real time.

People also ask

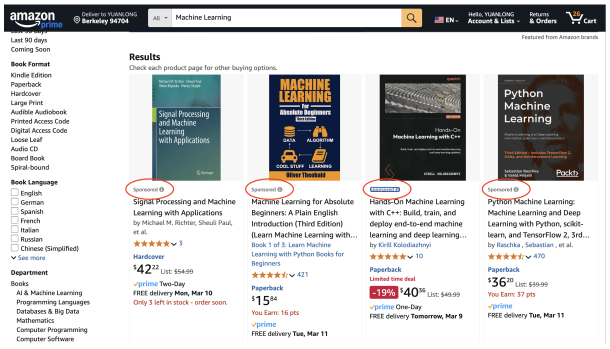
Which technology is used for real-time bidding?

How to do real-time bidding?

Does Google use real-time bidding?

What is an example of real-time bidding?

Figure 1.1: Google Search Ads



Last 90 days
Coming Soon

Book Format
Kindle Edition
Paperback
Hardcover
Large Print
Audible Audiobook
Printed Access Code
Digital Access Code
Loose Leaf
Audio CD
Board Book
Spiral-bound

Book Language
English
German
Spanish
French
Italian
Russian
Chinese (Simplified)
See more

Department
Books
AI & Machine Learning
Programming Languages
Databases & Big Data
Mathematics
Computer Programming
Computer Software

Results
Check each product page for other buying options.

Sponsored
Signal Processing and Machine Learning with Applications
by Michael M. Richter, Sheuli Paul, et al.
3
Hardcover
\$42.22 List: ~~\$54.99~~
prime Two-Day
FREE delivery Mon, Mar 10
Only 3 left in stock - order soon.

Sponsored
Machine Learning for Absolute Beginners: A Plain English Introduction (Third Edition) (Learn Machine Learning with…
Book 1 of 3: Learn Machine Learning with Python Books for Beginners
421
Paperback
\$15.84
You Earn: 16 pts
prime
FREE delivery Tue, Mar 11

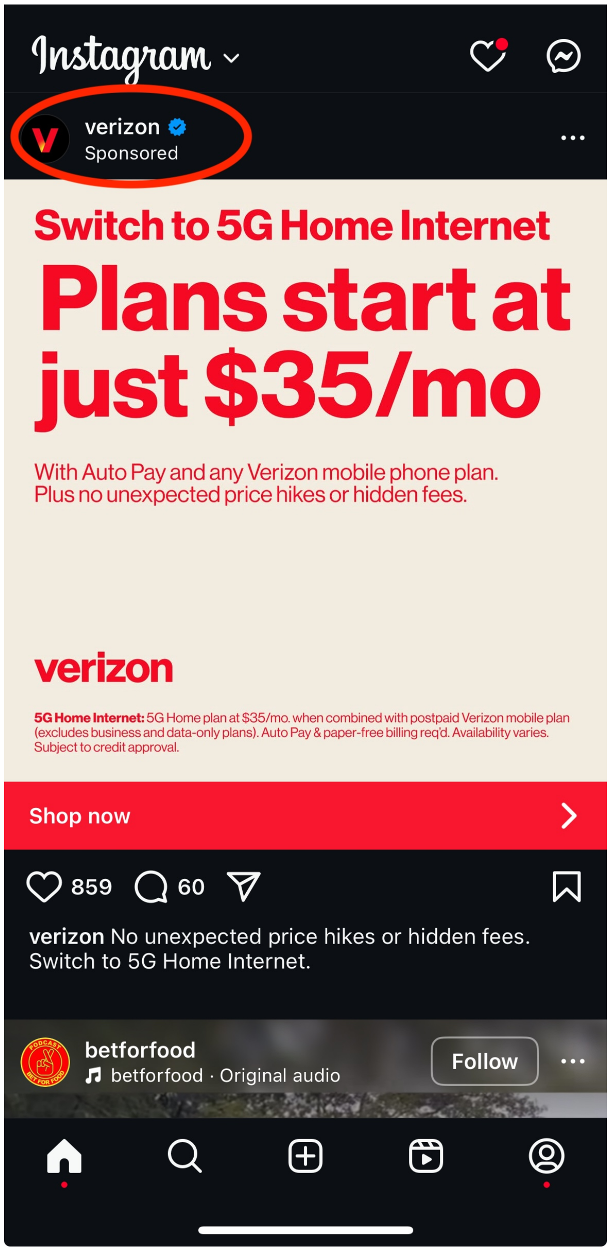
Sponsored
Hands-On Machine Learning with C++: Build, train, and deploy end-to-end machine learning and deep learning…
by Kirill Kolodiazhnyi
10
Paperback
Limited time deal
-19% \$40.36 List: ~~\$49.99~~
prime One-Day
FREE delivery Tomorrow, Mar 9

Sponsored
Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd…
by Raschka , Sebastian , et al.
470
Paperback
\$36.20 List: ~~\$39.99~~
You Earn: 37 pts
prime
FREE delivery Tue, Mar 11

Figure 1.2: Amazon Ads

The two snapshots above show typical search ads displayed when users perform searches on Google and Amazon. When a user searches for "real-time bidding platform" on Google, relevant ad campaigns related to the search keywords appear at the top of the results page. Similarly, on Amazon, when "Machine Learning" is searched, several ads for books titled "Machine Learning" are displayed among the top search results.

These keywords indicate strong user intent, and aligning ads with such search queries can significantly enhance campaign efficiency. This targeting strategy increases the likelihood of users clicking on an RTB platform's website or purchasing machine learning textbooks. Notably, in both cases, these ads are labeled as "Sponsored," signifying that they are paid advertisements.



Switch to 5G Home Internet

Plans start at just $35/mo

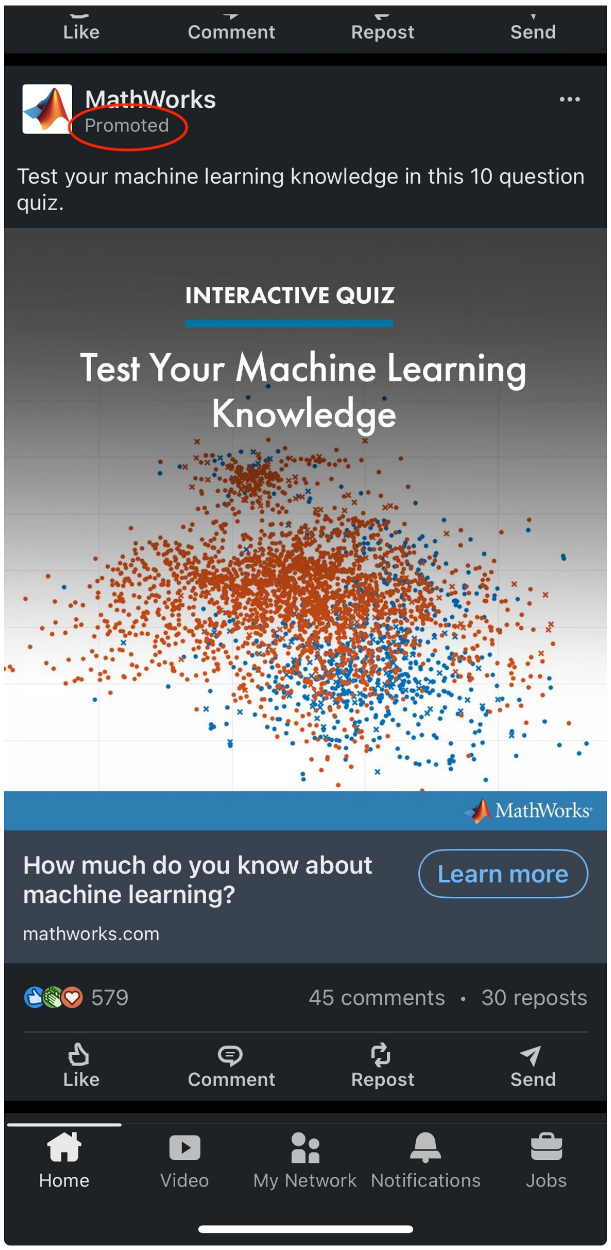
With Auto Pay and any Verizon mobile phone plan. Plus no unexpected price hikes or hidden fees.

verizon

5G Home Internet: 5G Home plan at $35/mo. when combined with postpaid Verizon mobile plan (excludes business and data-only plans). Auto Pay & paper-free billing req'd. Availability varies. Subject to credit approval.

verizon No unexpected price hikes or hidden fees. Switch to 5G Home Internet.

Figure 1.3: Instagram Ads



Test your machine learning knowledge in this 10 question quiz.

Test Your Machine Learning Knowledge

How much do you know about machine learning?

mathworks.com

Figure 1.4: LinkedIn Ads

These two snapshots illustrate a different type of ad placement. The left image shows an advertisement from Meta's *Instagram*, while the right image displays an ad from *LinkedIn*. Unlike search ads, which are triggered by user search queries, these ads are generated based on a user's profile and past behavior on the platform. They are automatically inserted between organic content in the feed.

For example, in LinkedIn's ad recommendation, an advertisement from *MathWorks* appears with the message: *"How much do you know about machine learning?"* This ad was likely shown because the user had recently searched for topics related to machine learning, and LinkedIn's ad delivery algorithm detected this intent.

2 Ads Serving Pipeline

We use LinkedIn's in-house ad delivery system as an example to illustrate how an ad campaign is served to users. Suppose an advertiser creates a campaign C through the ad management tool to maximize landing page clicks by targeting "Machine Learning Engineers." The campaign is set with a fixed budget (e.g., \$1000) and a defined start and end date. Once created, this campaign is added to LinkedIn's internal ad inventory within the ad-serving system.

Whenever a user logs into LinkedIn (via mobile or desktop), an ad request u is sent to the ad-serving system to allocate available ad slots in the user's feed. The serving algorithm is then triggered to check whether the user belongs to the target audience of campaign C. If the user qualifies (e.g., their profile contains the job title "Machine Learning Engineer"), the campaign is retrieved along with other eligible campaigns to participate an auction to compete for the available ad slots. The auction process for these slots depends primarily on two factors:

- **Ad Quality**: This is typically measured by the relevance between the campaign and the target user. For example, if a campaign is optimized for clicks, the quality score can be defined as the click-through rate (CTR)—the probability that the user will click on the ad. This score is usually predicted by a machine learning model, such as a deep neural network.

- **Bid Level**: The bid level is determined by factors such as the remaining budget, size of the target audience, and cost constraints set by the advertiser. For example, an advertiser may specify that the cost per click (CPC) must not exceed \$2. The bidding algorithm takes this constraint, along with other factors, into account to determine an optimal bid for the ad request. (This bidding process is a central topic discussed in this book.)

The auction algorithm then computes a ranking score for campaign C (along with other competing campaigns) based on both ad quality and bid level. One of the most widely used ranking scores is effective cost-per-mille (eCPM), which represents the cost per one thousand impressions. It is computed as:

$$\text{eCPM} = (\text{Bid per Click}) \times \text{CTR} \times 1000.$$

Since the advertiser is bidding per click, multiplying by CTR converts the expected cost into an impression-based metric.

The ad slots are then assigned to the campaigns with the highest ranking scores and displayed to the user accordingly. The advertiser is charged based on the auction's pricing/charging model—either per impression (CPM), per click (CPC), or per conversion (CPA). We illustrate this procedure in the following diagram:

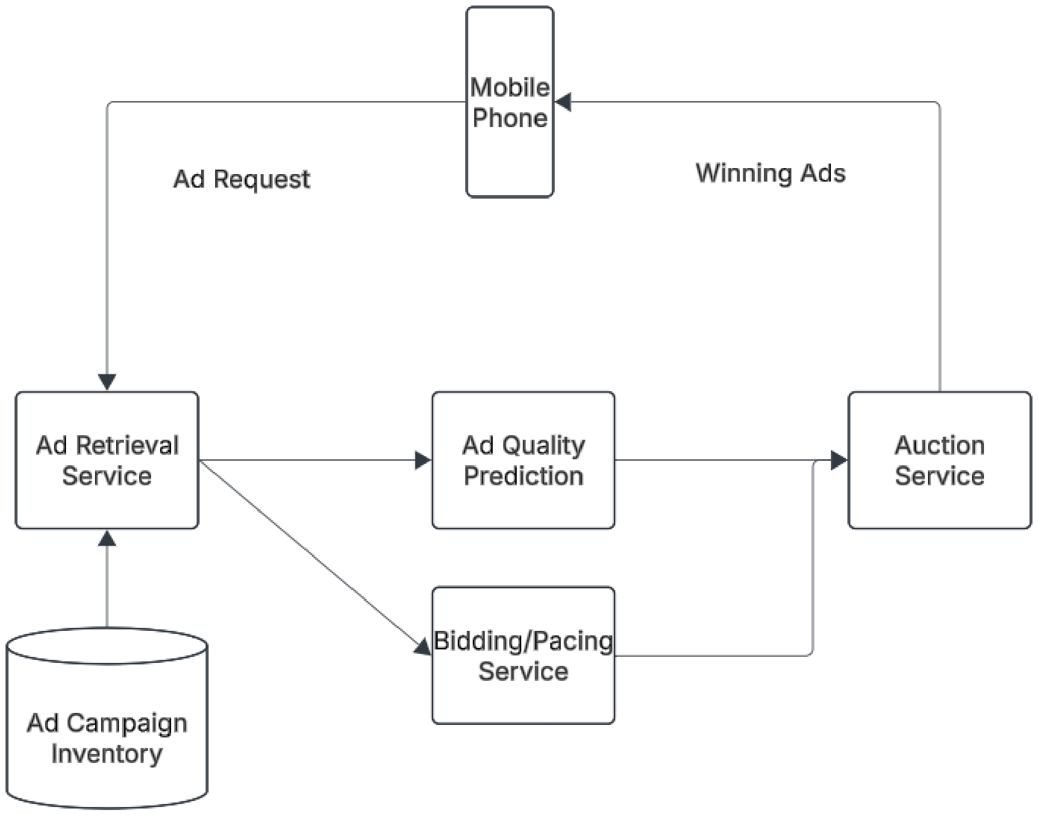


Figure 1.5: Illustration of the Ad Serving Pipeline

The serving pipeline described here is an over-simplified version; a real-world production system is significantly more complex. For example, the enormous volume of traffic and the vast number of campaigns make it infeasible to compute ad quality scores for all campaigns simultaneously using computationally intensive machine learning models, such as deep neural networks. To address this challenge, it is common to structure the process as a multi-stage funnel, incorporating multiple stages such as candidate generation, preliminary ranking, fine-grained ranking, and pacing/auction. Each stage involves trade-offs in computational cost, latency, and accuracy to ensure efficient and scalable ad delivery. For more details on ad-serving architectures, one may refer to works such as [22], [46], or [65].

3 Ad Campaign Configurations

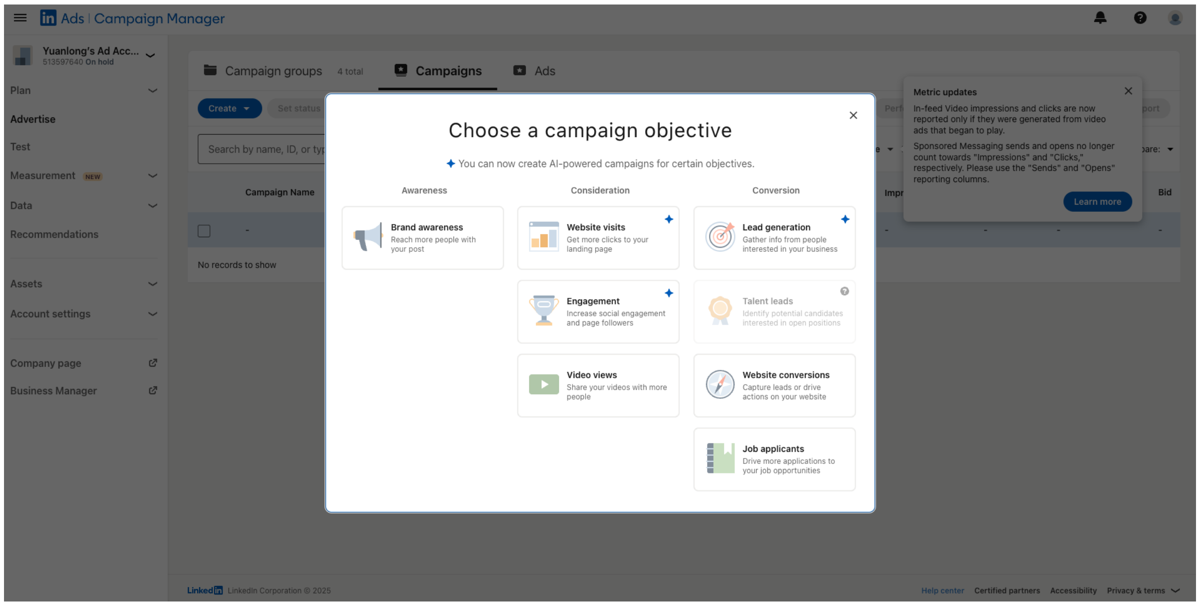
There are two important configurations of a campaign that help in designing the bidding and pacing strategy:

- **Objective and Optimization Goal**
- **Charging Model**

Objective and Optimization Goal The objective represents the primary outcome that advertisers seek—such as increasing brand awareness or driving conversions for a campaign. Within that objective, the optimization goal defines the specific metric that the ad platform will maximize. For example, if the objective is "Conversion", the optimization goal might be "Website Conversion" or "Lead Generation."

When advertisers create campaigns using the ad management tool, they are required to specify both the objective and the optimization goal. The platform's ad delivery algorithm then focuses on maximizing those specific actions to improve delivery efficiency. Different advertising platforms may define objectives and optimization goals differently.

Figure 1.6 illustrates the objectives and optimization goals available in LinkedIn Ads when advertisers create a new campaign using LinkedIn's Ads Campaign Manager. As shown, there are three main objectives: Awareness, Consideration, and Conversion. Within each objective, advertisers can select from multiple optimization goals.



Choose a campaign objective

You can now create AI-powered campaigns for certain objectives.

Awareness	Consideration	Conversion
Brand awareness Reach more people with your post	**Website visits** Get more clicks to your landing page	**Lead generation** Gather info from people interested in your business
	Engagement Increase social engagement and page followers	**Talent leads** Identify potential candidates interested in open positions
	Video views Share your videos with more people	**Website conversions** Capture leads or drive actions on your website
		Job applicants Drive more applications to your job opportunities

Metric updates

In-feed Video impressions and clicks are now reported only if they were generated from video ads that began to play.

Sponsored Messaging sends and opens no longer count towards "Impressions" and "Clicks," respectively. Please use the "Sends" and "Opens" reporting columns.

Figure 1.6: Objectives and Optimization Goals of LinkedIn Ads

Charging Model The charging model refers to how an advertiser is billed for ad placements and interactions. It determines the basis on which costs are calculated—such as per click, per thousand impressions, or per conversion.

For example, under a CPC (Cost Per Click) model, advertisers are charged each time a user clicks on the ad. Under a CPM (Cost Per Mille) model, advertisers pay for every 1,000 impressions (views), regardless of how many users engage with the ad. In an oCPM (Optimized Cost Per Mille) model, advertisers still pay for every 1,000 impressions, but the ad delivery is optimized towards achieving specific objectives, such as conversions.

Different platforms may offer various charging models to advertisers. For example, Google explicitly provides pure CPC bidding (especially for Search) and a well-defined CPV (Cost Per View) model for YouTube ads. In contrast, Facebook (Meta) frequently defaults to an impression-based billing system (CPM) while leveraging advanced algorithms (oCPM) to optimize for clicks, conversions, or video views.

Comparison of Different Bidding Products A campaign's objective, optimization goal, and charging model determine how the bidding and pacing optimization strategy is designed. For example, if the optimization goal of a campaign is to maximize total web conversions and the charging model is per impression, the corresponding bidding approach is oCPM. If the optimization goal is conversion and the charging model remains the same, CPA bidding may be used. Both oCPM and CPA bidding models compute an eCPM value to participate in the auction. The eCPM calculation is as follows:

$$eCPM = bid \times CVR \times CTR \times 1000.$$

where CVR represents the post-click conversion rate.

If the estimates for CTR and CVR are unbiased, both oCPM and CPA bidding strategies should theoretically yield similar results. However, when prediction models contain noise, oCPM advertisers risk overpaying for conversions due to inaccurate CTR or CVR estimates, leading to higher costs per conversion. In contrast, CPA bidding mitigates this risk by ensuring that advertisers are only charged for successful conversions, making it more resilient to prediction errors.

We compare some popular bidding products in Table 1.1 in terms of objective, optimization goal, and charging model.

Bidding Product	**Objective**	**Optimization Goal**	**Charging Model**
CPC (Cost Per Click)	Drive traffic to a website or landing page.	Maximizes clicks (link clicks, website visits).	Pay per click.
ECPC (Enhanced CPC)	Increase conversions while maintaining control over CPC.	Adjusts bids dynamically to acquire clicks more likely to convert.	Pay per click (bids adjust dynamically).
oCPC (Optimized CPC)	Drive more conversions by optimizing for higher-value clicks.	Optimizes delivery for clicks likely to result in conversions.	Pay per click (optimized for conversions).
CPM (Cost Per Mille)	Maximize brand awareness and visibility.	Maximizes ad impressions (reach as many users as possible).	Pay per 1,000 impressions.
oCPM (Optimized CPM)	Drive conversions with impression-based delivery.	Optimizes impressions for users most likely to take action.	Pay per 1,000 impressions (optimized for conversions).
CPA (Cost Per Action)	Acquire conversions efficiently.	Maximizes conversions at a predefined cost per action (purchase, sign-up, etc.).	Pay per completed action.
CPV (Cost Per View)	Maximize video engagement.	Optimizes for video views (e.g., 30-second watch or full view).	Pay per completed view.
CPL (Cost Per Lead [1])	Generate high-quality leads.	Maximizes lead form submissions (e.g., email sign-ups, inquiries).	Pay per lead captured.

Table 1.1: Comparison of Different Bidding Products

CHAPTER 2

OPTIMIZATION BASICS

We introduce fundamental optimization techniques here, focusing on two main topics: the primal-dual method and the gradient descent method. Both techniques will be utilized in later chapters to derive the optimal bidding formula and design the online bid update algorithm.

Optimization techniques will be used throughout the remainder of this book. To ensure the book is self-contained, we introduce some fundamental concepts and useful techniques of mathematical optimization in this chapter. The topics discussed here provide the minimal necessary foundation for deriving pacing algorithms in the subsequent chapters. We do not aim to cover all aspects of optimization, as the field is vast and beyond the scope of this book. For a more comprehensive understanding, readers may refer to classical optimization textbooks such as [12], [14] and [67].

In this book, we consider the following optimization problem:

$$\begin{aligned} \max_{x} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

The objective of (2.1) is to maximize the function $f(x)$ subject to the constraints $g_i(x) \leq 0$. Many real-world problems can be formulated in this way. For example, in a pacing algorithm, one may seek to maximize the total conversions of a campaign while adhering to constraints such as budget limits, cost controls, and other operational restrictions.

1 Primal-Dual Method

A fundamental approach for solving constrained optimization problems such as (2.1) is the primal-dual method, which leverages the concept of Lagrange duality. We provide a high-level overview of this method here, as it will be used frequently throughout this book. For a more comprehensive treatment of the primal-dual method and its applications, readers may refer to [26].

A typical procedure of the primal-dual method is as follows:

- **Construct the Lagrangian Function**: The Lagrangian function $\mathcal{L}(x, \lambda_i)$[1] transforms the constrained optimization problem (2.1) into an unconstrained optimization problem by incorporating penalty terms via Lagrange multipliers (dual variables) $\{\lambda_i\}_{i=1}^n$. It is defined as:

$$\mathcal{L}(x, \lambda_i) = f(x) - \sum_{i=1}^{n} \lambda_i \cdot g_i(x),$$

  where we require $\lambda_i \geq 0$. Notably, $\lambda_i \cdot g_i(x) \geq 0$ when $g_i(x)$ violates the constraints. Thus, we can interpret $-\sum_{i=1}^n \lambda_i \cdot g_i(x)$ as a penalty term in the maximization of $f(x)$. The Lagrangian function $\mathcal{L}(x, \lambda_i)$ effectively captures the trade-off between optimality (the objective function f) and feasibility (the constraint functions g_i).

- **Derive the Dual Problem**: Based on the Lagrangian $\mathcal{L}(x, \lambda_i)$ defined above, the dual function of (2.1) is obtained by taking the supremum of $\mathcal{L}(x, \lambda_i)$ with respect to x:

$$\mathcal{L}^*(\lambda_i) = \sup_{x} \mathcal{L}(x, \lambda_i).$$

[1]In this book, we use $\mathcal{L}(x, \lambda_i)$ to denote $\mathcal{L}(x, \lambda_1, \ldots, \lambda_i, \ldots, \lambda_n)$ when there is no risk of ambiguity.

The corresponding dual problem is to minimize the dual function $\mathcal{L}^*(\lambda_i)$ with respect to λ_i:

$$\min_{\lambda_i \geq 0} \mathcal{L}^*(\lambda_i). \tag{2.2}$$

- **Leverage the Primal-Dual Relationship**: Let f^* denote the optimal value of the primal problem (2.1), and let q^* be the optimal value of the dual problem (2.2). By definition, the dual problem provides an upper bound on the primal objective, meaning that:

  $$f^* \leq q^*.$$

  Solving the dual problem thus provides an approximation of the primal solution. Under certain conditions[2], strong duality holds, implying that the primal and dual solutions are equal:

  $$f^* = q^*.$$

  Furthermore, the optimal solution must satisfy the following **Karush-Kuhn-Tucker (KKT) conditions**:

  - **Primal Feasibility**:

    $$g_i(x^*) \leq 0.$$

  - **Dual Feasibility**:

    $$\lambda_i^* \geq 0.$$

  - **Complementary Slackness**:

    $$\lambda_i^* g_i(x^*) = 0.$$

  - **Stationarity**:

    $$\nabla_x \mathcal{L}(x^*, \lambda_i^*) = 0.$$

  Solving the KKT conditions yields an optimal solution for both the primal and dual problems when strong duality holds.

We use a concrete example to demonstrate how the primal-dual method can be applied to solve an optimization problem. Specifically, we aim to maximize a quadratic objective function subject to a linear constraint:

$$\begin{aligned} \max_x \quad & -\frac{1}{2}x^2 + b \cdot x \\ \text{s.t.} \quad & a \cdot x - c \leq 0, \end{aligned}$$

where a, b, c are constants, and $a \neq 0$.

Following the procedure outlined earlier, we first construct the Lagrangian function:

$$\mathcal{L}(x, \lambda) = -\frac{1}{2}x^2 + b \cdot x - \lambda \cdot (a \cdot x - c) .$$

[2]E.g., Slater's condition.

The corresponding dual function is given by:

$$\mathcal{L}^*(\lambda) = \sup_x \mathcal{L}(x, \lambda) = \sup_x \left(-\frac{1}{2}x^2 + b \cdot x - \lambda \cdot (a \cdot x - c) \right).$$

Since this is a quadratic function in x, it is straightforward to show that:

$$\mathcal{L}^*(\lambda) = \frac{1}{2}a^2\lambda^2 + (c - ab)\lambda + \frac{1}{2}b^2.$$

This is a quadratic function in λ. Since $\lambda \geq 0$, it attains its maximum at:

$$\lambda^* = \max\left(0, \frac{ab - c}{a^2}\right).$$

KKT Conditions The optimal solution must satisfy the KKT conditions:

- **Primal Feasibility**:
$$a \cdot x^* - c \leq 0.$$
- **Dual Feasibility**:
$$\lambda^* \geq 0.$$
- **Complementary Slackness**:
$$\lambda^* \cdot (a \cdot x^* - c) = 0.$$
- **Stationarity**:
$$\frac{\partial}{\partial x}\mathcal{L}(x^*, \lambda^*) = 0 \quad \Rightarrow \quad x^* = b - \lambda^* \cdot a.$$

Case Analysis There are two possible cases:

- **Case 1:** $\lambda^* = 0$

  This occurs when $ab - c < 0$. From the stationarity condition, we obtain:
$$x^* = b - \lambda^* \cdot a = b.$$
  In this case, the constraint is inactive.
- **Case 2:** $\lambda^* = \frac{ab-c}{a^2}$

  This occurs when $ab - c \geq 0$. From the stationarity condition, we obtain:
$$x^* = b - \lambda^* \cdot a = \frac{c}{a}.$$
  In this case, the constraint is active.

2 Gradient Descent Method

Gradient descent is a fundamental method for solving unconstrained mathematical optimization problems:

$$\min_x f(x) \tag{2.3}$$

The idea is straightforward: to find the minimum value of a function $f(x)$, we take steps in the opposite direction of the gradient at the current point, as this is the direction of steepest descent. This procedure is repeated iteratively until convergence. In this section, we provide a brief introduction to gradient descent method. For a more in-depth discussion of advanced techniques, readers may refer to [60].

Mathematically, let $f(x)$ be a differentiable function, and suppose we aim to minimize $f(x)$, where $x \in \mathbb{R}^n$ represents the parameter vector. The gradient descent method adaptively updates the parameter vector based on the following update rule until $f(x)$ converges:

$$x^{k+1} \leftarrow x^k - \epsilon \cdot \nabla_x f(x^k),$$

where:

- x^k is the parameter vector at the k-th iteration.
- $\epsilon > 0$ is the learning rate, which controls the step size.
- $\nabla_x f(x^k)$ is the gradient of the function evaluated at x^k.

It can be shown that, under certain regularity conditions and assuming $f(x)$ is a convex function, gradient descent converges to a global minimum with a properly chosen learning rate.

An Example Figure 2.1 illustrates how gradient descent works in practice to solve the following unconstrained optimization problem:

$$\min_{x_1, x_2} f(x_1, x_2) = x_1^2 + 4x_2^2.$$

Each elliptical curve in the figure represents a level set of the objective function $f(x)$, where $f(x) = c$ for some constant c. The function $f(x)$ is a convex quadratic function with its global minimum located at the origin $(0, 0)$. The level sets of this function are elliptical contours due to the different scaling of x_1 and x_2. Given a current iterate (x_1^k, x_2^k), the update rule for gradient descent is given by:

$$x^{k+1} \leftarrow x^k - \epsilon \cdot \nabla_x f(x^k),$$

where ϵ is the learning rate and $\nabla f(x^k)$ is the gradient of the function at the current point. For the given function, the gradient is computed as:

$$\nabla_x f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 8x_2 \end{bmatrix}.$$

Thus, the gradient descent update rule becomes:

$$\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \end{bmatrix} \leftarrow \begin{bmatrix} x_1^{k} \\ x_2^{k} \end{bmatrix} - \epsilon \cdot \begin{bmatrix} 2x_1^{k} \\ 8x_2^{k} \end{bmatrix} = \begin{bmatrix} x_1^{k} - 2\epsilon x_1^{k} \\ x_2^{k} - 8\epsilon x_2^{k} \end{bmatrix}.$$

This update rule shows that the values of x_1 and x_2 are reduced proportionally to their gradients, with different scaling due to the factors 2 and 8. As seen in Figure 2.1, starting from the initial point labeled "Start," gradient descent moves along the blue dotted path, progressively decreasing the function value at each step. The trajectory aligns with the steepest descent direction at each iteration. Eventually, the iterates converge to the global minimum at the point labeled "End," where $x_1 = 0$ and $x_2 = 0$, achieving the optimal solution.

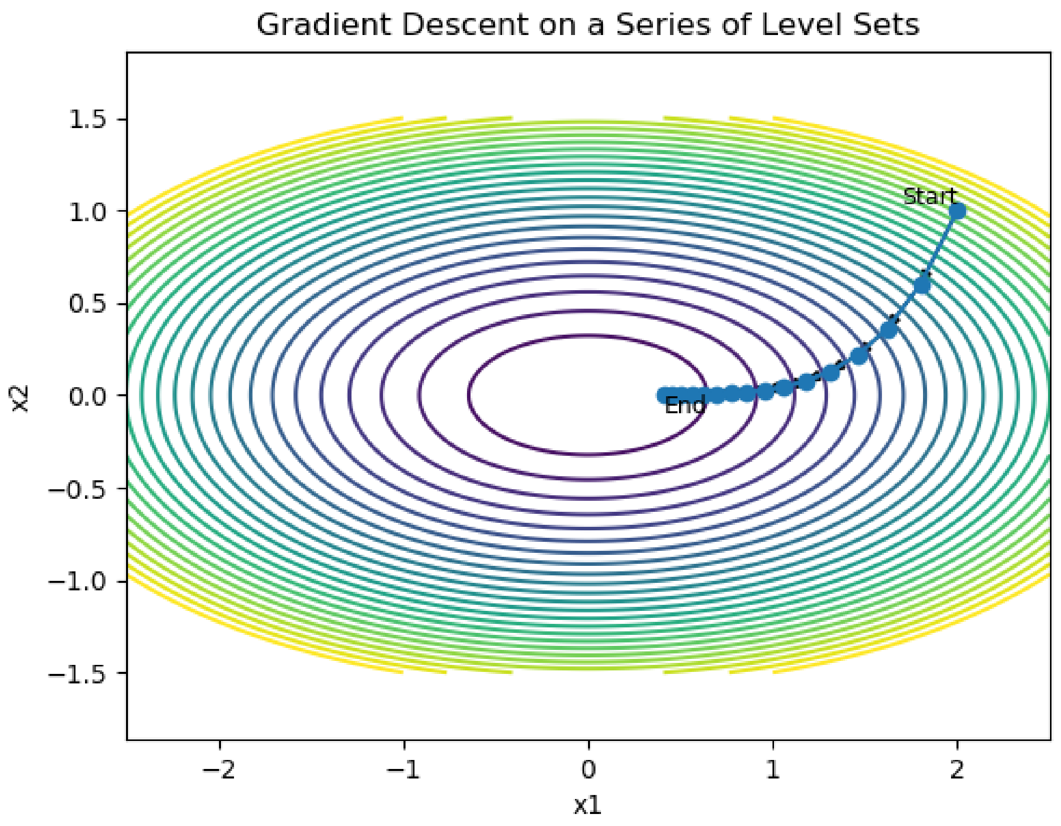


Figure 2.1: Illustration of the Gradient Descent Algorithm

2.1 Stochastic Gradient Descent

A useful variant of gradient descent is the so-called Stochastic Gradient Descent (SGD) method. In many optimization problems, the objective function is not a simple scalar function but rather an aggregate sum over multiple individual functions. A common form of such an objective function is:

$$f(x) = \sum_{i=1}^{n} g(y_i; x),$$

where each $g(y_i; x)$ represents the loss function associated with a single data point y_i, parameterized by the vector x. The data points $\{y_i\}_{i=1}^{n}$ may be drawn from an unknown i.i.d. distribution. Our goal is to find an optimal parameter x^* that minimizes the objective function $f(x)$. In this case, the optimization problem is formulated as:

$$\min_{x} \sum_{i=1}^{n} g(y_i; x). \tag{2.4}$$

The standard gradient descent method computes the full gradient of $f(x)$ at each iteration:

$$\nabla_x f(x) = \sum_{i=1}^{n} \nabla_x g(y_i; x).$$

The corresponding update rule is:

$$x^{k+1} \leftarrow x^k - \epsilon \cdot \sum_{i=1}^{n} \nabla_x g(y_i; x^k). \tag{2.5}$$

However, in many online optimization scenarios, data arrives in a streaming manner. In production systems, waiting for all samples to be available before updating the model is impractical. Thus, batch-based gradient descent is not a suitable approach in such cases.

Stochastic Gradient Descent (SGD) Stochastic Gradient Descent (SGD) approximates the full gradient by using only a single function $g(y_i; x)$ at each iteration. Instead of computing the sum over all n gradients, SGD updates the parameter using only one term at each step, allowing updates to be made in real-time:

$$x^{k+1} \leftarrow x^k - \epsilon \cdot \nabla_x g(y_k; x^k).$$

It can be shown that, under proper conditions (e.g., i.i.d. distribution of data samples, well-chosen learning rates), the SGD algorithm exhibits guaranteed convergence properties(e.g., see [11]). We will frequently use SGD throughout this book. A typical application is in real-time bidding systems, where auction requests arrive in a continuous stream, and we need to adaptively update parameters in real-time to determine the optimal bid level.

3 Remarks

3.1 Standard Form of an Optimization Problem

The formulation of the optimization problem in (2.1) is tailored to align with our applications in bidding problems in subsequent chapters. However, it is worth noting that in many optimization textbooks (e.g., [14]), the standard form of an optimization problem is typically presented as:

$$\begin{aligned} \min_{x} \quad & f(x) \\ \text{s.t.} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, n, \\ & h_j(x) = 0, \quad j = 1, \ldots, q. \end{aligned} \tag{2.6}$$

All these formulations are mathematically equivalent, and one can transform from one to another. For instance, to convert the maximization objective in (2.1) into the minimization form in (2.6), it suffices to introduce a negative sign:

$$\max_x f(x) \Leftrightarrow \min_x \left(-f(x)\right).$$

3.2 Stochastic Approximation Algorithm

In this section, we discuss the Stochastic Approximation Algorithm, specifically the Robbins-Monro Algorithm, which has a close connection with the stochastic gradient descent (SGD) method introduced earlier in this chapter.

The Robbins-Monro Algorithm is a fundamental stochastic approximation method introduced by Herbert Robbins and Sutton Monro in 1951. It provides an iterative procedure for finding the root of a function when only noisy observations are available.

Given a function $h(\theta)$, the goal is to determine θ^* such that:

$$h(\theta^*) = \mathbb{E}[X|\theta^*] = 0,$$

where X represents a noisy observation. Since X is not observed directly, we approximate it using a sample X_t drawn at each iteration. The Robbins-Monro algorithm updates the estimate of θ iteratively:

$$\theta_{t+1} = \theta_t - \epsilon_t h(\theta_t), \tag{2.7}$$

where:

- θ_t is the estimate at iteration t,
- ϵ_t is a sequence of step sizes (learning rates),
- $h(\theta_t)$ is an unbiased estimator of the true function value.

For the algorithm to converge to θ^* with probability 1, the step size sequence $\{\epsilon_t\}$ must satisfy the following conditions:

$$\sum_{t=1}^{\infty} \epsilon_t = \infty, \quad \sum_{t=1}^{\infty} \epsilon_t^2 < \infty. \tag{2.8}$$

These conditions ensure that the step size is initially large enough to allow exploration but diminishes over time to facilitate convergence. In practice, a common choice is:

$$\epsilon_t = \frac{1}{t}.$$

For further technical details on the Robbins-Monro algorithm, refer to [58].

Connection to Stochastic Gradient Descent Stochastic gradient descent (SGD) can be regarded as a special case of the Robbins-Monro algorithm. Consider the optimization problem:

$$\min_x \frac{1}{n} \sum_{i=1}^{n} g(y_i; x).$$

When n is large, we approximate:

$$\frac{1}{n} \sum_{i=1}^{n} g(y_i; x) \approx \mathbb{E}_{y \sim \mathcal{D}} \left[g(y; x) \right],$$

where $\mathcal{D}$ represents the distribution from which the random variable y is drawn.

We aim to find x^* such that:

$$\min_x \mathbb{E}_{y \sim \mathcal{D}} \left[g(y; x) \right] \quad \Leftrightarrow \quad \text{find } x^* \text{ such that } \frac{\partial}{\partial x} \mathbb{E}_{y \sim \mathcal{D}} \left[g(y; x^*) \right] = 0.$$

Under mild regularity conditions, such as the dominated convergence condition, we can interchange expectation and differentiation:

$$\frac{\partial}{\partial x} \mathbb{E}_{y \sim \mathcal{D}} \left[g(y; x^*) \right] = \mathbb{E}_{y \sim \mathcal{D}} \left[\frac{\partial}{\partial x} g(y; x^*) \right].$$

Setting:

$$h(x^*) = \mathbb{E}_{y \sim \mathcal{D}} \left[\frac{\partial}{\partial x} g(y; x^*) \right],$$

it follows that:

$$\frac{\partial}{\partial x} g(y_t; x_t)$$

is an unbiased estimator of $h(x^*)$. Applying the Robbins-Monro update rule (2.7), we recover the stochastic gradient descent algorithm:

$$x_{t+1} = x_t - \epsilon_t \frac{\partial}{\partial x} g(y_t; x_t). \tag{2.9}$$

This connection also suggests that when selecting learning rates for SGD, we should follow the conditions in (2.8) to ensure convergence.

3.3 Online Convex Optimization

Another topic closely related to our earlier discussion is *Online Convex Optimization* (OCO). In OCO, the goal is to design an efficient algorithm that makes decisions iteratively in scenarios where the outcome is revealed only after the decision has been made. More specifically, at each iteration, the algorithm chooses an action x_t, and after this action is committed, a convex loss function f_t is revealed. The objective is to minimize the regret, defined as

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(x_t) - \min_x \sum_{t=1}^{T} f_t(x).$$

This formulation has a deep connection to the pacing optimization problem discussed later in the book. One can think of f_t as the loss incurred after a bidding agent submits its bid to an auction (e.g., the loss from an auction opportunity). The pacing algorithm can then be viewed as an online decision-making process that aims to minimize its cumulative regret compared to the optimal fixed decision in hindsight.

An efficient first-order algorithm for solving the OCO problem is *Online Gradient Descent* (OGD), which updates the decision solely based on the gradient of the loss function f_t:

$$x_{t+1} = x_t - \epsilon_t \nabla f_t(x_t),$$

where ϵ_t is the step size at iteration t.

It can be shown that regret bounds for OCO directly translate into convergence rates for the online stochastic gradient descent algorithm. Readers interested in a more comprehensive treatment of this topic are referred to [38] and [79].

CHAPTER 3

BASICS OF AUCTION MECHANISMS

We introduce several of the most widely used auction mechanisms in the industry, including First Price Auction, Second Price Auction, VCG Auction, Myerson's Optimal Auction, and Generalized Second Price Auction. These auction mechanisms are widely adopted in real-time bidding (RTB) systems and have a profound impact on the design of bidding algorithms, which we will explore later in this book.

We provide a brief introduction to auction mechanisms in this chapter. We start by introducing single-item auctions, specifically the First Price Auction (FPA) and the Second Price Auction (SPA), with a discussion on incentive compatibility using these two auctions as examples. We then proceed to multiple-item incentive compatiple auction mechanism, the VCG auction. We also briefly discuss Myerson's classical optimal auction mechanism, which aims to maximize the total revenue of the auctioneer and is widely adopted by many companies in the industry. In the last section, we introduce one the most popular auctions in the industry: the Generalized Second Price Auction (GSP).

The discussion in this chapter serves as a primer and provides a working understanding of how auctions function in the online ad-serving funnel. For readers interested in exploring this topic further, a good introductory course on auction mechanisms is [50]. For general auction mechanism design, see [59], [54] and [37]. The classical paper on Myerson's optimal auction design is [53]. For the GSP auction, refer to [27] and [61]. For the VCG auction, see [21], [36], [63] and [62].

1 First Price Auction

An auction mechanism essentially determines which bidders win the auctioned items (allocation rules) and what prices they should pay (payment rules). Suppose there is an available ad slot and N ad campaigns are bidding for this slot with per click bid prices $\{b_i\}_{i=1}^N$, and their corresponding CTRs are $\{\alpha_i\}_{i=1}^N$. The per impression bids are $\{b_i \cdot \alpha_i\}_{i=1}^N$. A First Price Auction assigns the ad slot to the bidder with the highest bid per impression and charges them $b_k \cdot \alpha_k$ per impression, assuming k is the index of the winning bidder (campaign). At first glance, such a mechanism appears simple and intuitive, but bidders in a First Price Auction tend to bid lower than their true values.

To illustrate this, consider an example. Suppose there are two bidders (campaigns) competing for a single ad slot: bidder B_1 privately values each click at \$3 and bidder B_2 privately values each click at \$2. For simplicity, assume both bidders have a CTR of 1. If they bid truthfully (i.e., B_1 bids \$3 and B_2 bids \$2), B_1 wins the ad slot and pays \$3 per click.

If we define utility u as the difference between the bidder's true value v and their payment p, i.e.,

$$u = v - p,$$

then in this case, the utility for B_1 is:

$$u_1 = 3 - 3 = 0.$$

Clearly, this is not an optimal strategy for B_1. Instead of bidding truthfully, B_1 could bid slightly higher than B_2, say \$2.01. In this case, B_1 still wins the auction but pays a much lower price, significantly increasing their utility. This demonstrates that in a First Price Auction, bidders have an incentive to bid below their true value, otherwise, the utility will be zero. Additionally, bidders must consider the behavior of other bidders to determine how much to lower their bid in order to maximize their utility.

The example above also motivates the definition of incentive compatibility in auction mechanisms. In auction theory, incentive compatibility refers to a property of a mechanism or auction in which it is in every participant's best interest to act truthfully, i.e., to reveal their true valuation of the items or outcomes. This ensures that participants maximize their utility by being truthful, regardless of what others do.

Mathematically, let:

- N: the set of participants (bidders).
- $\mathcal{X}$: the set of possible outcomes of the mechanism.
- $v_i : \mathcal{X} \to \mathbb{R}$: the valuation function of participant i, representing their true value for each outcome.
- p_i: the payment made by participant i.
- u_i: the utility of participant i, defined as:

$$u_i(x, p_i) = v_i(x) - p_i,$$

  where $x \in \mathcal{X}$ is the chosen outcome.

A mechanism $(x(\cdot), p(\cdot))$ is **incentive compatible (IC)** if, for every participant i, reporting their true valuation v_i maximizes their utility:

$$v_i(x(v_i, v_{-i})) - p_i(v_i, v_{-i}) \geq v_i(x(v_i', v_{-i})) - p_i(v_i', v_{-i}),$$

for all $v_i' \neq v_i$, where v_{-i} represents the valuations of all other participants.

From the analysis above, we can conclude that the First Price Auction is not incentive compatible. We will discuss how to bid optimally under a First Price Auction later in this book.

2 Second Price Auction

Second Price Auction(or *Vickrey* auction) is different auction format in which the highest bidder wins the auction but, different from the First Price Auction, pays the second highest bid. Second price auction is incentive compatible. Use the example above, under second price auction, if bidder B_1 bids her true valuation of the ad slot, i.e., \$3, she only needs to pay \$2, the utility is \$3 - \$2 = \$1, there is no need for her to bid different than her private valuation. Indeed, we may prove that, under the second price auction, every bidder has a dominant strategy, that is to set her bid equal to her private valuation. More specifically, we have the following claim

A single-item second price auction is incentive compatible. If there are N bidders with private evaluations of the item $\{v_i\}_{i=1}^N$, the dominant strategy of each bidder is to set the bid b_i equal to v_i and this truthful bidding strategy maximizes the utility u_i of bidder i and the utilities are non-negative if bidder bids truthfully.

Fix a bidder i, let $B = \max_{j \neq i} b_j$ denote the highest bid among other bidders. The utility function u_i of bidder i is defined as

$$u_i = \begin{cases} v_i - B & \text{if bidder } i \text{ wins,} \\ 0, & \text{otherwise.} \end{cases}$$

To prove this claim, we just need to show that the truthful bidding strategy(i.e., $b_i = v_i$) maximizes the utility of bidder i. We analyze the utility for bidder i when they report truthfully ($b_i = v_i$) versus when they misreport ($b'_i \neq v_i$):

- **Case 1: Bidder i wins by bidding $b_i = v_i$:**
  - The bidder wins if $v_i >= B$, and their utility is:

    $$u_i = v_i - B.$$

- **Case 2: Bidder i wins by bidding $b'_i > v_i$:**
  - The bidder might win, but their utility remains the same:

    $$u_i = v_i - B.$$

    Overbidding does not increase their utility beyond truthfully reporting.

- **Case 3: Bidder i loses by bidding $b'_i < v_i$:**
  - If bidder i underbids and loses (i.e., $b'_i < B$), their utility becomes:

    $$u_i = 0,$$

    even though they could have won the item and obtained a positive utility by bidding truthfully.

In all cases, bidding truthfully maximizes the bidder's utility u_i. It's easy to see in all cases, the utility is non-negative.

The incentive compatibility property discussed above implies that Second Price Auctions are particularly convenient for bidders to participate in, as truthful bidding is always the dominant strategy. This ensures that no other strategy yields higher utility, regardless of the bids of others. Additionally, it avoids both the risk of losing the auction unnecessarily (as in underbidding) and the risk of overpaying (as in overbidding), making it highly appealing for real-world applications.

3 VCG Auction

The Second Price Auction works well for a single item (in the sense that it is incentive compatible). However, in real-world scenarios (e.g., in real-time bidding), it is very common to auction multiple items simultaneously. This motivates the need for a generalization that preserves desirable properties such as incentive compatibility and efficiency in such environments. The *Vickrey-Clarke-Groves (VCG)* mechanism is one such generalization that achieves the following:

- **Efficient Allocation:** The resources are allocated to maximize total social welfare, defined as the sum of the valuations of all participants.
- **Incentive Compatibility:** Each participant's dominant strategy is to truthfully reveal their private valuation, even in multi-item or combinatorial settings.

The formal definition of the ***VCG*** mechanism is as follows: Consider N participants (bidders) and a set of possible allocations A. Each participant i has a private valuation function $v_i(a)$ for an allocation $a \in A$. The goal of the auction is to select the allocation a^* that maximizes total social welfare:

$$a^* = \arg\max_{a \in A} \sum_{i=1}^{N} v_i(a).$$

Once the optimal allocation a^* is determined, the payment p_i for each participant i is computed as:

$$p_i = h_i - \sum_{j \neq i} v_j(a^*),$$

where h_i is the hypothetical social welfare if participant i were excluded:

$$h_i = \max_{a \in A} \sum_{j \neq i} v_j(a).$$

This payment structure ensures that each participant pays an amount equal to the "externality" they impose on others by participating in the auction. It can be shown that under the ***VCG*** auction, the optimal strategy for each bidder is to bid truthfully, a rigirous proof could be found in [54].

Let's explore two concrete examples to demonstrate how the VCG mechanism works.

Example 1: Single Item Auction

We apply VCG to a single-item auction. Suppose there are N bidders competing for a single ad slot with private valuations $\{v_i\}_{i=1}^{N}$. The allocation rule a^* is given by:

$$a^* = \arg\max_{a \in A} \sum_{i=1}^{N} v_i(a).$$

Since there is only one slot available, only one campaign can win the auction. The winner (e.g., bidder k) values the outcome at v_k, while the rest of the bidders value the outcome at 0 as they do not win the auction. To maximize $\sum_{i=1}^{N} v_i(a)$, the optimal allocation rule assigns the ad slot to the bidder with the highest valuation. Thus, bidder k with the highest valuation wins the auction.

For payments, the payment for bidder k is:

$$p_k(a^*) = \max_{a \in A} \sum_{j \neq k} v_j(a) - \sum_{j \neq k} v_j(a^*),$$

where:

- $\max_{a \in A} \sum_{j \neq k} v_j(a)$: The maximum value achievable by all other bidders if bidder k does not participate, which is simply the highest valuation among all other bidders, i.e., $\max_{j \neq k} v_j$.
- $\sum_{j \neq k} v_j(a^*) = 0$: In the optimal allocation, only bidder k is assigned the ad slot, and all other bidders lose the auction.

Thus:

$$p_k(a^*) = \max_{j \neq k} v_j.$$

It is clear that all other bidders pay \$0. Hence, in a single-item auction, the VCG mechanism reduces to the regular Second Price Auction.

Example 2: Two Ad Slots with Three Bidders

Consider two ad slots with the following properties:

- Ad slot X: CTR = 0.2
- Ad slot Y: CTR = 0.1

The bidders have the following valuation per click:

- Bidder 1: \$100/click
- Bidder 2: \$40/click
- Bidder 3: \$20/click

The effective per-impression bids (CTR multiplied by the bid per click) for each slot are:

- Bidder 1: \$20 for slot X, \$10 for slot Y
- Bidder 2: \$8 for slot X, \$4 for slot Y
- Bidder 3: \$4 for slot X, \$2 for slot Y

The total valuations for different allocations of the two slots are shown in the following table:

Allocation	$X1Y2$	$X1Y3$	$X2Y1$	$X2Y3$	$X3Y1$	$X3Y2$
Bidder 1	20	20	10	0	10	0
Bidder 2	4	0	8	8	0	4
Bidder 3	0	2	0	2	4	4

Table 3.1: Valuations for different allocations of ad slots.

Note: In this table, X1Y2 means bidder 1 *wins slot* X *and bidder* 2 *wins slot* Y*, and so on.*

VCG Payment for Bidder 1

To compute the payment for Bidder 1, we calculate the externality they impose on others by winning their assigned slot. The payment for Bidder 1, $p_1(a^*)$, is given by:

$$p_1(a^*) = \max_{a \in A} \sum_{j \neq 1} v_j(a) - \sum_{j \neq 1} v_j(a^*),$$

where:

- $\max_{a \in A} \sum_{j \neq 1} v_j(a)$: The maximum valuation achievable by all other bidders if Bidder 1 does not participate.
- $\sum_{j \neq 1} v_j(a^*)$: The total valuation of all other bidders under the current allocation a^*.

For this example:

- $\max_{a \in A} \sum_{j \neq 1} v_j(a) = 10$: allocation $X2Y3$: Bidder 2 wins slot X and Bidder 3 wins slot Y;
- $\sum_{j \neq 1} v_j(a^*) = 4$: current allocation $X1Y2$: Bidder 2 wins slot Y with valuation 4, Bidder 3 gets no slot.

Thus:

$$p_1(a^*) = 10 - 4 = 6.$$

VCG Payment for Bidder 2

Next, we compute the payment for Bidder 2, $p_2(a^*)$, who wins slot Y. The payment is given by:

$$p_2(a^*) = \max_{a \in A} \sum_{j \neq 2} v_j(a) - \sum_{j \neq 2} v_j(a^*),$$

where:

- $\max_{a \in A} \sum_{j \neq 2} v_j(a) = 22$: Allocation $X1Y3$: Bidder 1 wins slot X with valuation 20 and Bidder 3 wins slot Y with valuation 2.
- $\sum_{j \neq 2} v_j(a^*) = 20$: Current allocation $X1Y2$: Bidder 1 wins slot X with valuation 20, and Bidder 3 gets no slot.

Thus:

$$p_2(a^*) = 22 - 20 = 2.$$

Conclusion:

- Bidder 1 pays \$6 for slot X.
- Bidder 2 pays \$2 for slot Y.

Actually, we can prove that if there are k ad slots with CTRs $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$, and N bidders such that the j-th bidder's value v_j satisfies $v_1 \geq v_2 \geq \cdots \geq v_N$ (re-indexed if necessary), then: If we want to design a DSIC (Dominant Strategy Incentive Compatible) welfare-maximization auction mechanism, the only possible allocation is to assign the j-th highest bidder to the j-th slot for $j = 1, 2, \cdots, k$ and the payment for the i-th highest bidder is given by:

$$p_i = \sum_{j=1}^{k} b_{j+1}(\alpha_j - \alpha_{j+1}), \tag{3.1}$$

where $b_j = v_j$ is the truthful bid price for bidder j, and we set $\alpha_{k+1} = 0$. Readers can verify that this allocation and payment mechanism produces the same result as the one derived from the VCG auction definition in the example above. We will provide a sketch of the proof in the Remarks section, while a detailed proof can be found in [59].

A quick summary of the VCG mechanism

- **Efficiency:** The allocation maximizes social welfare.
- **Incentive Compatibility:** Truthful bidding is the dominant strategy, as payments depend only on the valuations of others, not on the bidder's own valuation.
- **Fairness:** Participants pay only for the externality they impose, ensuring a fair payment structure.

4 Myerson's Optimal Auction

The VCG mechanism is designed to maximize total welfare (i.e., the sum of participants' valuations). However, from the auctioneer's perspective, the goal may be to maximize profit instead of welfare. This raises the question:

What does a profit-maximizing mechanism look like?

To answer this, we consider Myerson's Optimal Auction, which provides a framework for maximizing the auctioneer's profit while maintaining truthfulness (incentive compatibility).

Key Concepts

If the mechanism is truthful, and fixing the bids of all other participants, the expected payment of bidder i is given by:

$$p_i(b_i) = b_i x_i(b_i) - \int_0^{b_i} x_i(z)dz,$$

where $x_i(b_i)$ is the probability that bidder i wins the item given their bid b_i.

If we assume that bidder i's valuation follows a known cumulative distribution function (CDF) F_i with probability density function (PDF) f_i, we define the *virtual value* for a given valuation v as:

$$\phi_i(v) = v - \frac{1 - F_i(v)}{f_i(v)}.$$

Myerson's Theorem

Theorem (Myerson, 1981): The expected profit of any truthful mechanism is equal to its expected total virtual valuations.

Proof:

$$
\begin{aligned}
\mathbb{E}[p(b)] = \int_0^h p(b)f(b)db &= \int_0^h bx(b)f(b)db - \int_0^h \int_0^b x(z)f(b)dzdb, \\
&= \int_0^h bx(b)f(b)db - \int_0^h x(z) \int_z^h f(b)dbdz, \\
&= \int_0^h bx(b)f(b)db - \int_0^h x(z)\left[1 - F(z)\right] dz, \\
&= \int_0^h \left[b - \frac{1 - F(b)}{f(b)}\right] x(b)f(b)db, \\
&= \mathbb{E}\left[\phi(b)x(b)\right].
\end{aligned}
$$

This demonstrates that maximizing expected profit is equivalent to maximizing the total virtual valuations.

Myerson's Optimal Auction

From Myerson's Theorem, we conclude:

$$\text{Profit maximization} \iff \text{Maximizing total virtual valuation.}$$

This insight allows us to leverage the VCG mechanism. To design a profit-maximizing auction, we translate bids into virtual bids and run a VCG auction on these virtual bids. The steps for Myerson's Optimal Auction are as follows:

1. Given bids $\mathbf{b}$ and value distributions $\mathbf{F}$, compute the "virtual bids" $\mathbf{b}'_i = \phi_i(\mathbf{b}_i)$.
2. Run a VCG auction on the virtual bids to obtain the allocation $\mathbf{x}'$ and payments $\mathbf{p}'$.
3. Output $\mathbf{x} = \mathbf{x}'$ and $\mathbf{p}_i = \phi^{-1}(\mathbf{p}'_i)$.

Note: In VCG, it makes no sense to accept negative bid prices. Therefore, Myerson's Optimal Auction introduces a reservation price to ensure that virtual bids are non-negative.

Example: Single Item Auction

Consider a single ad slot auction with two bidders, where the bid prices are b_1 and b_2. The allocation and payment rules are as follows:

- Bidder 1 wins if $\phi_1(b_1) \geq \max\{\phi_2(b_2), 0\}$, and their payment is:

$$p_1 = \inf\{b : \phi_1(b) \geq \phi_2(b_2) \text{ and } \phi_1(b) \geq 0\}.$$

- Bidder 2 wins if $\phi_2(b_2) \geq \max\{\phi_1(b_1), 0\}$, and their payment is:

$$p_2 = \inf\{b : \phi_2(b) \geq \phi_1(b_1) \text{ and } \phi_2(b) \geq 0\}.$$

If the value distributions are identical ($F_1 = F_2 = F$), then:

- Bidder 1 wins if $b_1 \geq \max\{b_2, \phi^{-1}(0)\}$, and their payment is:

$$p_1 = \max\{b_2, \phi^{-1}(0)\}.$$

- Bidder 2 wins if $b_2 \geq \max\{b_1, \phi^{-1}(0)\}$, and their payment is:

$$p_2 = \max\{b_1, \phi^{-1}(0)\}.$$

Optimal Auction for i.i.d. Distributions

For a single item auction with i.i.d. value distributions F, Myerson's Optimal Auction reduces to a Vickrey Second Price Auction with a reservation price $\phi^{-1}(0)$.

Example: Suppose F is uniform on $[0, 1]$. Then:

$$F(z) = z, \quad f(z) = 1 \implies \phi(z) = 2z - 1 \implies \phi^{-1}(0) = \frac{1}{2}.$$

Profit Comparison:

- Profit without a reservation price: $\frac{1}{3}$.
- Profit with a reservation price of $\frac{1}{2}$: $\frac{5}{12}$.

Remarks

One limitation of Myerson's optimal auction design is that it only applies to the so-called "single-parameter" auction environment, where each bidder can be characterized by a single number (the private value each bidder has for winning the item). However, in real-time bidding, there are usually multiple ad slots auctioned simultaneously. In this scenario, the "single-parameter" assumption no longer holds, and it becomes complex to extend Myerson's optimal auction mechanism design. There is active ongoing research on this topic, and readers who are interested may refer to, e.g., [15] and [47].

5 GSP Auction

Due to historical reasons, the most popular auction mechanism in the industry for multi-item auctions is the **Generalized Second Price Auction (GSP)**. It has been widely adopted by many IT companies, especially for sponsored search ads auctions and social network feed ads auctions.

The GSP auction is simple and intuitive: the highest bidder gets the top slot, the second-highest bidder gets the second slot, and so on. However, each bidder pays a price equal to the effective bid of the next-highest bidder for the slot they win.

Example: How GSP Works

To illustrate how the GSP auction works, we revisit the example discussed in section 3:

- There are two ad slots, X and Y, with the following Click-Through Rates (CTR):
  - Ad slot X: $\text{CTR} = 0.2$
  - Ad slot Y: $\text{CTR} = 0.1$
- There are three bidders with valuations per click:
  - Bidder 1: \$100/click
  - Bidder 2: \$40/click
  - Bidder 3: \$20/click
- The effective per-impression bids, computed as CTR $\times$ bid per click, are:
  - Bidder 1: \$20 for slot X, \$10 for slot Y
  - Bidder 2: \$8 for slot X, \$4 for slot Y
  - Bidder 3: \$4 for slot X, \$2 for slot Y

Allocation and Payments in GSP:

- **Allocation:**
  - Bidder 1 wins slot X.
  - Bidder 2 wins slot Y.
- **Payments:**
  - Bidder 1 pays \$8 (the effective bid of Bidder 2 for slot X).
  - Bidder 2 pays \$2 (the effective bid of Bidder 3 for slot Y).

The total payments are:

- Bidder 1: \$8
- Bidder 2: \$2

6 Remarks

6.1 Myerson's Lemma

We first prove Myerson's lemma and then prove (3.1) based on this lemma.

Lemma 1 (Myerson's Lemma). *Let $G(b)$ be the probability that a bidder with bid b wins an auction, and let $H(b)$ be the expected payment that the bidder makes when bidding b. In any dominant-strategy incentive-compatible (DSIC) mechanism, the following relation holds:*

$$H(b) \;=\; b\,G(b) \;-\; \int_0^b G(z)\,\mathrm{d}z.$$

Proof. **Step 1: Setup.**

Consider a single bidder whose private valuation is b. Let $G(b)$ denote the probability that she *wins* the auction (or is allocated the good) when she bids b, and let $H(b)$ denote her *expected payment* in that situation. The bidder's *expected utility*, when she truthfully bids her valuation b, is

$$u(b) \;=\; b\,G(b) \;-\; H(b).$$

Step 2: Envelope Theorem Argument.

A mechanism is *dominant-strategy incentive-compatible* (DSIC) if bidding one's true value b is a best response regardless of others' bids. Informally, this implies that the utility $u(b)$ is the maximum possible utility the bidder can achieve, *given* her true value b. Hence

$$u(b) \;=\; \max_{b'}\Big\{b\,G(b') \;-\; H(b')\Big\}.$$

In such scenarios, the *envelope theorem* says that if $u(b)$ is differentiable, its derivative with respect to b is simply the partial derivative of the objective at the chosen maximizer $b' = b$. Concretely,

$$\frac{\mathrm{d}}{\mathrm{d}b}\,u(b) \;=\; \frac{\partial}{\partial b}\Big(b\,G(b') \;-\; H(b')\Big)\Big|_{b'=b}.$$

Since $G(b')$ and $H(b')$ depend on the *bid* b', rather than directly on the *true value* b, they are constant with respect to b when we evaluate the partial derivative at $b' = b$. Thus

$$\frac{\mathrm{d}}{\mathrm{d}b}\,u(b) \;=\; G(b).$$

This is the key result from the envelope theorem in the DSIC setting.

Step 3: Integrate $u'(b) = G(b)$.

We have established that

$$u'(b) \;=\; G(b).$$

Integrate both sides from 0 to b:

$$u(b) \;-\; u(0) \;=\; \int_0^b G(z)\,\mathrm{d}z.$$

Typically, in auction settings, when a bidder's valuation is 0, her utility is 0. Formally, $u(0) = 0$. Therefore,

$$u(b) \;=\; \int_0^b G(z)\,\mathrm{d}z.$$

Step 4: Solve for $H(b)$.

Recall that

$$u(b) \;=\; b\,G(b) \;-\; H(b).$$

Hence

$$b\,G(b) \;-\; H(b) \;=\; \int_0^b G(z)\,\mathrm{d}z.$$

Rearranging to isolate $H(b)$ gives

$$H(b) \;=\; b\,G(b) \;-\; \int_0^b G(z)\,\mathrm{d}z.$$

This is precisely the statement of Myerson's Lemma.

□

We can now proceed to prove (3.1), which we summarize in the following theorem:

Theorem 1 (Welfare-Maximizing k-Slot Auction Is VCG)**.** *Consider an ad auction with k ad slots whose click-through rates (CTRs) satisfy*

$$\alpha_1 \;\geq\; \alpha_2 \;\geq\; \ldots \;\geq\; \alpha_k \;\geq\; 0.$$

There are N bidders, each bidder i having a private value v_i for a single click. An assignment of the k slots must match exactly one slot to each of (up to) k highest bidders. Suppose we want a dominant-strategy incentive-compatible *(DSIC) mechanism that* always maximizes total welfare *(i.e. sum of bidder values × CTRs). Then, up to tie-breaking, the* only possible *DSIC welfare-maximizing allocation is:*

$$\textit{assign the } j\textit{-th highest bid to slot } j, \quad j = 1, \ldots, k,$$

and the corresponding unique *DSIC payment rule is given by*

$$p_i \;=\; \sum_{j=1}^{k} \Big(\alpha_j - \alpha_{j+1}\Big)\, b_{j+1}, \quad \textit{for the bidder } i \textit{ whose bid is } b_i \textit{ (the } j\textit{-th highest)}, \tag{3.2}$$

where we set $\alpha_{k+1} = 0$ and $b_{k+1} = 0$ for notational convenience.

Proof. **Step 1: Welfare Maximization Implies Sorting by CTR.**

Let us index the bidders so that $b_1 \geq b_2 \geq \cdots \geq b_N$ are their *bids*. Because the CTRs satisfy $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_k$, the *welfare-maximizing* assignment pairs the highest bid with α_1, the second-highest bid with α_2, and so on, up to the k-th highest bid with α_k (if $k \leq N$). This ensures total welfare

$$\alpha_1 b_1 + \alpha_2 b_2 + \cdots + \alpha_k b_k$$

is maximized.

Step 2: DSIC Requires Monotonic Allocation.

A fundamental requirement for *dominant-strategy* truthfulness is that each bidder's *allocation* be *monotone* in her own bid. In a single-slot setting, that means "if bidder i increases her bid, her probability of winning does not decrease." In a multi-slot context, monotonicity translates to: if bidder i raises her bid, she *cannot* receive a *lower* slot (one with a lower CTR). Concretely,

1. If you bid more, your slot assignment (and hence your expected number of clicks) can only become *better* (or stay the same), never worse.

2. Combining this with *welfare maximization* (which sorts by bids), one sees that the unique feasible allocation is:

$$\text{(highest bid)} \to \alpha_1, \quad \text{(2nd highest bid)} \to \alpha_2, \quad \dots, \quad \text{(k-th highest bid)} \to \alpha_k.$$

Step 3: Payments Are Forced by Myerson's Lemma (the Envelope Theorem). Once the *allocation* rule is fixed and is monotone, *Myerson's Lemma* says the *expected payment* function $H_i(\cdot)$ for each bidder i is determined by the envelope condition:

$$H_i(b) \;=\; b\,G_i(b) \;-\; \int_0^b G_i(z)\,\mathrm{d}z,$$

where $G_i(b)$ is the *(weighted) probability* that bidder i receives a slot (or the expected CTR she obtains) when her bid is b. In the k-slot environment:

$$G_i(b) \;=\; \text{the CTR of the slot assigned to } i\text{, given } b.$$

If bidder i is the j-th highest bidder, then $G_i(b_i) = \alpha_j$. As we *raise* the bidder's bid from 0 up to b_i, the bidder "moves up the ladder" of possible slots in a way determined by the sorted bids of others. Hence, the payment $H_i(b_i)$ is pinned down by how $G_i(\cdot)$ changes from 0 to b_i.

Step 4: Recovering the VCG (Sum-of-Externalities) Price Formula.

Concretely, in a discrete sense, if i is the j-th highest bidder, then to "get slot j" instead of slot $j+1$, the bid must exceed the $(j+1)$-th highest bid b_{j+1}. Tracking the changes in slot assignment across these "threshold bids" yields exactly:

$$p_i \;=\; \sum_{m=j}^{k-1} \Bigl(\alpha_m - \alpha_{m+1}\Bigr)\, b_{m+1}, \quad \text{where } \alpha_{k+1} = 0 \text{ and } b_{k+1} = 0.$$

That is the standard generalized Vickrey–Clarke–Groves (VCG) price formula: each bidder i pays the "externality" imposed on the lower-ranked bidders, weighted by the slot-quality (CTR) differences $\alpha_m - \alpha_{m+1}$. One may check that this is just a discrete rewrite of Myerson's envelope integral.

Conclusion.

Thus, under the twin requirements of (1) maximizing total welfare and (2) monotonicity in each bidder's own bid (to ensure DSIC), the *only* valid assignment is the "highest bidder to the highest CTR" rule, and Myerson's Lemma shows that there is *exactly one* payment scheme that makes it DSIC:

$$p_i \;=\; \sum_{j=1}^{k} \bigl(\alpha_j - \alpha_{j+1}\bigr)\, b_{j+1}.$$

This completes the proof. □

6.2 A Historical Note of GSP

As discussed in section 3, the payment rule derived there is the unique one that ensures the DSIC (Dominant Strategy Incentive Compatibility) property. Therefore, we can conclude that the GSP auction is **not DSIC**.

GSP can be regarded as an incorrectly implemented version of the DSIC auction. Nevertheless, it has gained widespread popularity despite its non-DSIC property. Google played a major role in popularizing the GSP auction. Interestingly, they considered transitioning from GSP to VCG during the summer of 2002. However, as mentioned in [62], three major issues prevented the change:

1. The existing GSP auction was growing rapidly and required significant engineering attention, making it challenging to develop a new auction.

2. The VCG auction was harder to explain to advertisers.

3. The VCG auction required advertisers to raise their bids above the levels they were accustomed to in the GSP auction.

As a result, the idea of transitioning to VCG was shelved in 2002.

6.3 Auction Mechanism Design for Auto-Bidding

The mechanisms discussed in this chapter focus exclusively on single-shot auctions. While these are widely adopted in industry, most bidding and pacing algorithms run across repeated auctions in real time. When an advertiser launches a campaign, it participates in multiple real-time auctions under a fixed budget, which introduces new design challenges. For example, in a repeated-auction setting, a bidder facing a tight budget may strategically shade its valuation—even under incentive-compatible mechanisms such as second-price auctions—in order to maximize overall performance. Designing efficient auction mechanisms for real-world digital-advertising scenarios under various constraints (e.g., budget caps, ROI targets) remains an active area of academic research. For more technical details, see [2].

CHAPTER 4

EXPERIMENT FRAMEWORK

In this chapter, we introduce the A/B testing frameworks used in the advertising domain, which serve as powerful statistical methods for quantitatively measuring the impact of new strategies applied to ad campaigns. We discuss two frameworks: the general campaign-level A/B test and the budget-split A/B test.

When implementing a new algorithm, whether it is a bidding strategy or an enhancement to a prediction model, it is essential to evaluate its effectiveness and compare it against the existing baseline model. This evaluation is conducted using the A/B testing framework.

The general procedure is as follows: for the objects to which you want to apply the new strategy, **randomly** split them into two groups: the control group and the treatment group. In the control group, the baseline strategy remains unchanged, while in the treatment group, the new strategy is applied. After a certain period, relevant metrics are collected and analyzed to determine whether the new strategy outperforms the baseline.

Nearly all modifications within the ad-serving funnel must undergo this form of testing, and only those that demonstrate a positive impact on business metrics are deployed to the production system. In this chapter, we introduce two of the most commonly used A/B testing frameworks in the advertising domain: Campaign-Level A/B Testing and Budget-Split A/B Testing.

1 Campaign-Level A/B Test

In this section, we discuss the Campaign-Level A/B Test framework, in which the split is performed at the campaign level. Suppose there are N campaigns in the pool (N can be a large number—some major platforms may have over one million campaigns running simultaneously). Engineers develop a new pacing algorithm, P_1, aimed at improving budget utilization. To evaluate its effectiveness against the existing pacing strategy, P_0, the campaigns are split into two groups: the control group A and the treatment group B. The baseline strategy P_0 is applied to A, while P_1 is applied to B. The experiment is then run for a certain period, during which data is collected to compare the actual budget utilization between the two groups.

The key question is: how can we determine whether the results at the end of the test are statistically reliable for decision-making?

To address this, we use *hypothesis testing*, a statistical method that makes inferences about population parameters (such as budget utilization in our case) based on sample data. This process involves the following steps:

- **Define Null and Alternative Hypotheses**: The null hypothesis (H_0) assumes that the new strategy has no effect, while the alternative hypothesis (H_1) suggests otherwise. In our case, H_0 states that the new pacing algorithm P_1 does not alter budget utilization.

- **Choose a Significance Level** α[1]: This represents the probability of incorrectly rejecting the null hypothesis H_0. For instance, selecting $\alpha = 5\%$ implies a 5% chance of rejecting H_0 when it is actually true.

- **Select a Test Statistic**: A test statistic T is computed from the sample data collected in the experiment. In our case, we gather budget utilization data from campaigns in both the control and treatment groups and compute a test statistic to quantify how much the sample deviates from H_0.

[1] $1 - \alpha$ is referred to as the confidence level, representing how often we correctly fail to reject H_0 when it is true.

- **Compute the p-value**: Assuming H_0 is true, we calculate the p-value, which represents the probability of obtaining a test result at least as extreme as the observed test statistic T.
- **Make the Decision**: If $p < \alpha$, we reject H_0 because the probability of observing such an extreme result under H_0 is too low ($< \alpha$). Otherwise, we do not reject H_0.

To illustrate this process, consider the example above: suppose both groups contain $N = 100,000$ campaigns. At the end of the test, we obtain budget utilization data $\{bu_{A,i}\}$ and $\{bu_{B,i}\}$ from the control group A and the treatment group B, respectively. We compute the average budget utilization for each group as follows:

$$\text{mean}_A = \frac{1}{N}\sum_{i=1}^{N} bu_{A,i}, \quad \text{mean}_B = \frac{1}{N}\sum_{i=1}^{N} bu_{B,i}.$$

Suppose we obtain $\text{mean}_A = 95.14\%$ and $\text{mean}_B = 97.15\%$. At first glance, budget utilization appears to have improved by approximately 2%, but is this improvement statistically significant? To answer this, we perform a hypothesis test by defining the hypotheses:

$$H_0 : \text{mean}_A = \text{mean}_B, \quad H_1 : \text{mean}_A \neq \text{mean}_B.$$

Choosing a significance level of $\alpha = 5\%$, we compute the Student's t-statistic as:

$$T = \frac{\text{mean}_A - \text{mean}_B}{\sqrt{\frac{(n_A-1)\cdot std_A^2 + (n_B-1)\cdot std_B^2}{n_A+n_B-2}} \cdot \sqrt{\frac{1}{n_A} + \frac{1}{n_B}}} \tag{4.1}$$

where n_A and n_B are the sample sizes of groups A and B, respectively (in this case, $n_A = n_B = N$). The standard deviations std_A and std_B for each group are computed as:

$$\text{std}_A = \sqrt{\frac{1}{n_A - 1}\sum_{i=1}^{n_A}(bu_{A,i} - \text{mean}_A)^2}, \quad \text{std}_B = \sqrt{\frac{1}{n_B - 1}\sum_{i=1}^{n_B}(bu_{B,i} - \text{mean}_B)^2}.$$

We will later prove that, under the assumption that groups A and B have equal variance[2] ,T follows a Student's t-distribution, whose probability density function is illustrated in Figure 4.1. Suppose the computed t-statistic is -2.98. In the figure, the blue line represents this observed t-statistic, which lies in the left tail of the distribution. The red dotted line marks the critical value corresponding to the significance level $\alpha = 5\%$. Under H_0, the t-statistic is expected to be centered around zero, meaning that extreme values in the red regions are unlikely to occur if H_0 is true. Since our observed t-statistic falls within this critical region, we reject H_0, concluding that the budget utilization increase from 95.14% to 97.15% is statistically significant and attributable to the new algorithm P_1. As the impact on key metrics is positive, we can confidently deploy the new strategy to production.

[2]For the unequal-variance case, Welch's test can be used instead.

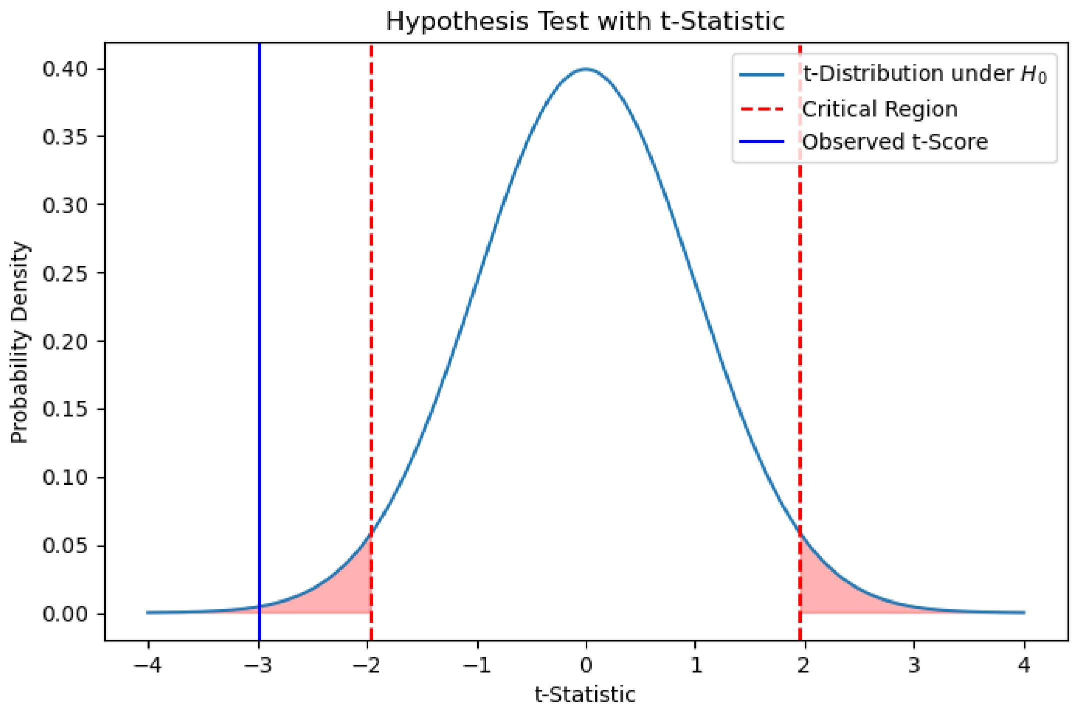


Figure 4.1: Hypothesis Test with *t*-Statistic

For further details on hypothesis testing, refer to standard statistical inference textbooks such as [17] and [57].

2 Budget Split A/B Test

The campaign-level experiment appears reasonable at first glance; however, upon closer examination of the design, several issues arise:

- **Skewed Budget Distribution:** The scale of budgets varies significantly across different campaigns. For small and medium-sized business (SMB) campaigns, the budget may be less than 100 dollars, whereas large enterprise branding campaigns can have budgets reaching hundreds of thousands or even exceeding one million dollars. This highly skewed distribution of campaign budgets diminishes the statistical power of the campaign-level A/B test, requiring a larger number of campaigns and a longer experiment duration to achieve statistically significant results.

- **Cannibalization Bias:** A more critical issue is known as *cannibalization bias*. Consider a scenario in which the baseline pacing algorithm P_0 already achieves a 100% budget utilization rate. At the same time, we aim to test a more aggressive pacing algorithm, P_1, using a campaign-level A/B test. Since campaigns in both groups may participate in the same auction, the treatment group, driven by the more aggressive P_1, is more likely to win the auction. Consequently, at the end of the test, we may observe a higher budget utilization rate in the treatment group than in the control group, even though P_0 already achieves 100% budget utilization.

The root cause of this phenomenon is that the control and treatment groups are competing against each other for auction opportunities. Some auction requests are effectively cannibalized by the more aggressive strategy P_1, introducing bias into the campaign-level A/B test results.

The Budget Split testing framework was introduced to address the limitations of campaign-level A/B testing in ad marketplace experiments. The underlying idea is straightforward: instead of randomly splitting campaigns, we divide the budget within each campaign, effectively creating two identical *sub-campaigns*. The control group consists of half of these sub-campaigns running the baseline strategy, while the treatment group comprises the other half, which retains identical budget settings but tests the new strategy. All auction requests are then randomly assigned and directed to one of these two groups.

This approach is conceptually equivalent to creating two parallel ad marketplaces with identical budget configurations, thereby effectively eliminating *cannibalization bias*. Additionally, it can be shown that the Budget Split framework is statistically more powerful than the campaign-level test, as it is capable of detecting smaller impacts of the new strategy that might be undetectable in a campaign-level experiment.

For more theoretical and practical analysis of the Budget Split framework, readers may refer to [45] and [10].

3 Remarks

3.1 Proof of Student's t-Test

Here we prove that T in (4.1) follows a Stduent's t-distribution with $n_A + n_B - 2$ degrees of freedom defined as follows:

If $Z \sim \mathcal{N}(0,1)$ follows a standard normal distribution and $W \sim \chi^2_k$ follows a chi-square distribution with k degrees of freedom, then the Student's t-distribution with degrees of freedom k is defined as:

$$\frac{Z}{\sqrt{W/k}} \sim t_k.$$

Test Statistic Derivation Let $X_1, X_2, \ldots, X_n$ be a random sample from a normal distribution $\mathcal{N}(\mu_X, \sigma^2)$ and $Y_1, Y_2, \ldots, Y_m$ be a random sample from $\mathcal{N}(\mu_Y, \sigma^2)$, both with unknown mean but common variance σ^2.

The sample means and variances are given by:

$$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i, \qquad S_X^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2,$$

$$\bar{Y} = \frac{1}{m}\sum_{i=1}^{m} Y_i, \qquad S_Y^2 = \frac{1}{m-1}\sum_{i=1}^{m}(Y_i - \bar{Y})^2.$$

Since we assume equal variances, we use the pooled variance estimator:

$$S_p^2 = \frac{(n-1)S_X^2 + (m-1)S_Y^2}{n+m-2}. \tag{4.2}$$

The test statistic is then defined as:

$$t = \frac{\bar{X} - \bar{Y}}{S_p\sqrt{\frac{1}{n} + \frac{1}{m}}}. \tag{4.3}$$

Distribution of the Test Statistic To prove that the test statistic follows a t-distribution, we analyze its components.

- **Step 1: Distribution of Sample Means** From properties of normal distributions,

  $$\bar{X} \sim \mathcal{N}\left(\mu_X, \frac{\sigma^2}{n}\right),$$
  $$\bar{Y} \sim \mathcal{N}\left(\mu_Y, \frac{\sigma^2}{m}\right).$$

  Thus, their difference follows:

  $$\bar{X} - \bar{Y} \sim \mathcal{N}\left(\mu_X - \mu_Y, \sigma^2\left(\frac{1}{n} + \frac{1}{m}\right)\right). \tag{4.4}$$

- **Step 2: Standardization** Define the standardization term:

  $$Z = \frac{(\bar{X} - \bar{Y}) - (\mu_X - \mu_Y)}{\sigma\sqrt{\frac{1}{n} + \frac{1}{m}}}. \tag{4.5}$$

  Since it is a linear transformation of a normal variable, it follows a standard normal distribution:

  $$Z \sim \mathcal{N}(0, 1). \tag{4.6}$$

- **Step 3: Distribution of the Pooled Variance** The pooled variance S_p^2 is the sum of two independent chi-square distributed variables:

  $$(n-1)S_X^2 \sim \sigma^2\chi^2_{n-1},$$
  $$(m-1)S_Y^2 \sim \sigma^2\chi^2_{m-1}.$$

  Thus, their sum follows a chi-square distribution with $n + m - 2$ degrees of freedom:

  $$(n+m-2)S_p^2 \sim \sigma^2\chi^2_{n+m-2}. \tag{4.7}$$

  Defining

  $$W = \frac{(n+m-2)S_p^2}{\sigma^2} \sim \chi^2_{n+m-2}, \tag{4.8}$$

we can express the test statistic as:

$$t = \frac{Z}{\sqrt{W/(n+m-2)}}. \tag{4.9}$$

Since $Z \sim \mathcal{N}(0,1)$ and $W \sim \chi^2_{n+m-2}$, it follows that:

$$t \sim t_{n+m-2}. \tag{4.10}$$

This completes the proof.

3.2 Other Considerations

Some additional factors to consider when conducting ad marketplace experiments include:

- **Simpson's Paradox**
- **ROI/Revenue Tradeoff**
- **Holdout Test**

Part II

Pacing Algorithms

CHAPTER 1

BIDDING PROBLEM FORMULATION

In this chapter, we provide a rigorous mathematical formulation of two primary bidding problems, namely max delivery and cost cap, in the context of repeated auction settings. We then employ the primal-dual method to derive the optimal bidding formulas. These results serve as the foundation for designing online control algorithms, which will be explored in the subsequent chapters.

In the first chapter of this part, we introduce a framework that formulates the budget pacing problem as a mathematical optimization problem through bidding. We focus specifically on the max delivery and cost cap problems, with all optimizations occurring at the campaign level unless stated otherwise. These two problems serve as examples to illustrate the core principles behind designing practical bidding algorithms.

For simplicity, unless explicitly stated otherwise, we assume throughout this book that the campaign follows a daily pacing strategy in an oCPM model, participating in a standard second-price auction where charges are incurred per impression and the objective/optimization goal is to maximize the total number of clicks.

1 Max Delivery

In the Max Delivery setting, advertisers set up a campaign with a specified budget. The objective is to optimize the clicks of the ad campaign while adhering to this budget constraint. Assuming this is a campaign operating under the standard Second Price Auction framework, a common goal is to maximize the total clicks for the campaign. Thus, the Max Delivery problem can be formulated as the following optimization problem:

$$\begin{aligned} \max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot r_t \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \leq B \end{aligned} \tag{1.1}$$

where,

- T represents the total (predicted) number of auction opportunities within a day.
- r_t is the predicted click-through rate (CTR) for the t-th auction.
- c_t denotes the cost for the t-th auction. In a second-price auction, this corresponds to the highest eCPM among other bidders.
- x_t is a binary decision variable indicating whether we win the t-th auction.

Under the rules of a second-price auction, $x_t = 1$ if and only if our bid per impression exceeds the highest competing bid, mathematically expressed as:

$$x_t = \mathbb{1}_{\{b_t > c_t\}}$$

where b_t is the bid per impression for t-th auction. We assume that both the sequences $\{r_t\}$ and $\{c_t\}$ follows some unknown independent and identically distributed (*i.i.d.*) distribution, such as a log-normal distribution.

1.1 Optimal Solution to Max Delivery Problem

It is challenging to solve this problem directly. Instead of addressing it in the primal space, we apply the primal-dual method to transform it into the dual space. The Lagrangian of (1.1) is given by:

$$\mathcal{L}(x_t, \lambda) = \sum_{t=1}^{T} x_t \cdot r_t - \lambda \cdot \left(\sum_{t=1}^{T} x_t \cdot c_t - B \right)$$

The dual is expressed as:

$$\min_{\lambda \geq 0} \mathcal{L}^*(\lambda) = \min_{\lambda \geq 0} \max_{x_t \in \{0,1\}} \mathcal{L}(x, \lambda).$$

We can rewrite $\mathcal{L}(x_t, \lambda)$ as:

$$\mathcal{L}(x, \lambda) = \sum_{t=1}^{T} x_t \cdot (r_t - \lambda c_t) + \lambda B.$$

To maximize $\mathcal{L}(x_t, \lambda)$, we can set $x_t = 1$ whenever $r_t - \lambda c_t > 0$, and $x_t = 0$ otherwise. Consequently, $\mathcal{L}^*(\lambda) = \max_{x_t \in \{0,1\}} \mathcal{L}(x_t, \lambda)$ becomes:

$$\mathcal{L}^*(\lambda) = \sum_{t=1}^{T} (r_t - \lambda c_t)_+ + \lambda B,$$

where $(z)_+ = \mathbb{1}_{\{z>0\}} \cdot z$ is the ReLU function. Therefore, the dual problem is:

$$\min_{\lambda \geq 0} \mathcal{L}^*(\lambda) = \min_{\lambda \geq 0} \sum_{t=1}^{T} \left[(r_t - \lambda c_t)_+ + \lambda \cdot \frac{B}{T} \right]. \tag{1.2}$$

Suppose the problem is feasible and

$$\lambda^* = \arg\min_{\lambda \geq 0} \mathcal{L}^*(\lambda)$$

The KKT conditions indicate that if $\lambda^* > 0$ is the optimal dual variable, budget constraint must satisfy the following:

$$\sum_{t=1}^{T} x_t \cdot c_t = B$$

The optimal bid per impression is determined as:

$$b_t^* = \frac{r_t}{\lambda^*}$$

The optimal bid per click is given by

$$b_{click}^* = \frac{1}{\lambda^*} \tag{1.3}$$

As the optimal bid is constant, under the assumption that r_t and c_t are subject to some unknown i.i.d. distribution, the expected cost for each eligible auction opportunity $\mathbb{E}[x_t \cdot c_t]$

in this campaign should also be constant. Therefore, the budget spend of the campaign within a time slot $\Delta\tau$ should be proportional to the number of eligible auction opportunities served during that time slot.,i.e.,

$$\sum_{\tau \le t \le \tau+\Delta\tau} x_t c_t \propto \# \text{ of auction opportunities in } (\tau, \tau+\Delta\tau).$$

For more technical details, one may refer to [31], [43] and [64].

Quick summary of our main results

The optimal bid per click for (1.1) in the stochastic setting is a constant bid:

$$b^*_{click} = \frac{1}{\lambda^*}$$

Suppose supply is sufficient (T big enough), the constant optimal bid b^*_{click} is the bid per click that exactly depletes the budget, it also suggests that the amount of budget depleted within a time interval is proportional to the number of auction opportunities, i.e.,

$$\sum_{\tau \le t \le \tau+\Delta\tau} x_t c_t \propto \# \text{ of auction opportunities in } (\tau, \tau+\Delta\tau).$$

2 Cost Cap

Cost Cap is a product designed for price-sensitive advertisers. In addition to specifying a budget in max delivery, the advertiser also defines a cost cap, which sets an upper limit on the average cost per result. This ensures that the average cost per result does not exceed the specified cap. Using the notation from the previous section, the cost cap problem for an oCPM daily campaign with click optimization goal can be formulated as follows:

$$\begin{aligned} \max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot r_t \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \le B \\ & \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} \le C \end{aligned} \tag{1.4}$$

where

- T represents the total (predicted) number of auction opportunities within a day.
- r_t is the predicted click-through rate (CTR) for the t-th auction.

- c_t denotes the cost for the t-th auction. In a second-price auction, this corresponds to the highest eCPM
- x_t is a binary decision variable indicating whether we win the t-th auction that can be expressed as $x_t = \mathbb{1}_{\{b_t > c_t\}}$.
- C is the cap for average CPC specified by the advertiser.

We make the same assumption that the sequences $\{r_t\}$ and $\{c_t\}$ follow some unknown *i.i.d.* distributions.

2.1 Optimal Solution to Cost Cap Problem

We apply the primal-dual method to solve (1.4), as was done for the maximum delivery problem. The key difference is that we now have two constraints. The Lagrangian for this problem is given by:

$$\mathcal{L}(x, \lambda, \mu) = \sum_{t=1}^{T} x_t \cdot r_t - \lambda \cdot \left(\sum_{t=1}^{T} x_t \cdot c_t - B \right) - \mu \cdot \left[\sum_{t=1}^{T} x_t \cdot c_t - C \cdot \left(\sum_{t=1}^{T} x_t \cdot r_t \right) \right]$$

The dual is expressed as:

$$\min_{\lambda \geq 0, \mu \geq 0} \mathcal{L}^*(\lambda, \mu) = \min_{\lambda \geq 0, \mu \geq 0} \max_{x_t \in \{0,1\}} \mathcal{L}(x, \lambda, \mu).$$

Note that $\mathcal{L}(x, \lambda, \mu)$ can be rewritten as:

$$\mathcal{L}(x, \lambda, \mu) = \sum_{t=1}^{T} x_t \cdot (r_t - \lambda c_t - \mu c_t + \mu C r_t) + \lambda B.$$

Similarly, to maximize $\mathcal{L}(x, \lambda, \mu)$, we set $x_t = 1$ whenever $r_t - \lambda c_t - \mu c_t + \mu C r_t > 0$, and $x_t = 0$ otherwise. $\mathcal{L}^*(\lambda, \mu) = \max_{x_t \in \{0,1\}} \mathcal{L}(x, \lambda, \mu)$ then becomes:

$$\mathcal{L}^*(\lambda, \mu) = \sum_{t=1}^{T} (r_t - \lambda c_t - \mu c_t + \mu C r_t)_+ + \lambda B,$$

where $(\cdot)_+$ again is the ReLU function. The dual problem of (1.4) is:

$$\min_{\lambda \geq 0, \mu \geq 0} \mathcal{L}^*(\lambda, \mu) = \min_{\lambda \geq 0, \mu \geq 0} \sum_{t=1}^{T} \left[(r_t - \lambda c_t - \mu c_t + \mu C r_t)_+ + \lambda \cdot \frac{B}{T} \right]. \tag{1.5}$$

Suppose we have feasible solution to this problem

$$\lambda^*, \mu^* = \arg\min_{\lambda \geq 0, \mu \geq 0} \mathcal{L}^*(\lambda, \mu)$$

The optimal bid per impression is determined as:

$$b_t^* = \frac{1 + \mu^* C}{\lambda^* + \mu^*} \cdot r_t$$

The optimal bid per click is given by

$$b^*_{click} = \frac{1}{\lambda^* + \mu^*} + \frac{\mu^*}{\lambda^* + \mu^*} \cdot C = \frac{\lambda^*}{\lambda^* + \mu^*} \cdot \frac{1}{\lambda^*} + \frac{\mu^*}{\lambda^* + \mu^*} \cdot C \tag{1.6}$$

Setting $\alpha = \lambda^*/(\lambda^* + \mu^*)$, we have

$$b^*_{click} = \alpha \cdot \frac{1}{\lambda^*} + (1 - \alpha) \cdot C \tag{1.7}$$

Note that $1/\lambda^*$ is the optimal bid in max delivery without considering the cost constraint. Therefore, the optimal bid for the cost cap is simply a linear combination of the unconstrained max delivery bid and the cost cap bid. When $\mu^* \to 0$ (i.e., $\alpha^* \to 1$), the cost constraint becomes invalid, and $b^*_{\text{click}} \to 1/\lambda^*$, reducing the problem to the max delivery problem. Conversely, when $\lambda^* \to 0$, the budget constraint becomes invalid, and $b^* \to \frac{1}{\mu^*}C$, meaning the campaign will bid at the maximum level allowed under the cost constraint.

Quick summary of our main results

The optimal bid per click for cost cap problem (1.4) in the stochastic setting is a constant, more specifically, simply a linear combination of the unconstrained max delivery bid and the cost cap bid:

$$b^*_{click} = \alpha \cdot \frac{1}{\lambda^*} + (1 - \alpha) \cdot C$$

where $\alpha = \frac{\lambda^*}{\lambda^* + \mu^*}$.

3 Remarks

3.1 Knapsack Problem

The bidding problems presented in this chapter are closely related to the well-known Knapsack problem. The name "Knapsack" originates from a scenario in which a person aims to fill a fixed-size knapsack with the most valuable items while adhering to weight constraints.

Mathematically, the problem can be formulated as follows: given N items, each with a weight w_i and a value v_i, the goal is to select a subset of items such that the total weight does not exceed a given knapsack capacity W, while maximizing the total value of the selected items:

$$\begin{aligned} \max_{x_i \in \{0,1\}} \quad & \sum_{i=1}^{N} x_i \cdot v_i \\ \text{s.t.} \quad & \sum_{i=1}^{N} x_i \cdot w_i \le W \end{aligned} \tag{1.8}$$

This is known as the 0-1 Knapsack problem. Notably, this formulation is essentially identical to the max delivery problem (1.1). For a general introduction to the Knapsack problem, readers may refer to [49]. A discussion on the online Knapsack problem can be found in [48] and the references cited therein.

The work of [77] was the first to model the max delivery problem as an online Knapsack problem. Readers interested in technical details can refer to this paper for further insights.

3.2 Budget Allocation Based on Conversion Rate Distribution

TBA

3.3 Duality Gap

TBA [3] [52]

3.4 Another Formulation of Cost Cap

TBA: with objective to maximize total spend

CHAPTER 2

THROTTLE-BASED PACING

In this chapter, we dicuss budget pacing algorithms via throttling. In the context of budget pacing, throttling refers to a mechanism that controls a campaign's participation in real-time auctions based on its actual budget spending relative to a target budget, which is determined by the supply pattern. This approach ensures that the ad campaign's budget is distributed in alignment with the supply pattern over its duration, thereby optimizing the campaign's performance.

1 Probabilistic Throttling

In throttle-based pacing, ad campaigns participate in online auctions with a fixed, pre-defined bid. We first consider a daily max-delivery campaign. If the fixed bid is set inappropriately and it's relatively high, it is highly likely that the campaign will win the majority of auction opportunities early on. As a result, the campaign may overspend at the start, rapidly exhausting the budget well before the end of the day. The probabilistic throttling algorithm is a mechanism used to control the participation of a campaign in real-time auctions based on its current spending relative to a target budget. The algorithm operates by dynamically adjusting a participation probability $p(t)$ that determines whether the campaign enters or skips a given auction. If the campaign is currently over-delivered (i.e., actual spending exceeds the expected spending), $p(t)$ is decreased, making it less likely to participate in the current auction, and vice versa when the campaign is under-delivered. This probabilistic control ensures that the campaign's budget is reasonably distributed over its time horizon while maximizing performance opportunities.

Modifications can be made to adapt this algorithm for a cost cap setting, where an additional performance constraint is imposed to ensure that the campaign achieves its goals within a specified cost threshold.

1.1 Throttling for Max-Delivery Campaign

From the discussion in section 1, still assumming i.i.d. distributions of the conversion rates $\{r_t\}$ and costs $\{c_t\}$, we know that the optimal budget allocation is achieved when the budget for each duration is distributed proportionally to the total number of eligible auction opportunities (supply) available during that period.

Suppose the prediction model estimates there are T auction opportunities for this campaign within a day(in practice, T is typically derived by analyzing historical time series data, and the prediction accuracy of T at the campaign level may vary, which can degrade the performance of the pacing algorithm. Some online adjustments might be implemented to reduce the prediction noise; however, we will not discuss those techniques here. For simplicity, we assume the prediction is perfect). At the t-th auction, with a perfect pacing algorithm, the spend should be $\alpha(t) = \frac{t}{T} \cdot B$, where B is the total budget. If the actual spend $S(t) > \alpha(t)$, meaning pacing is ahead of schedule, we should slow down the pacing rate. An intuitive approach is to set a participation probability $p(t)$, which determines the likelihood of the campaign participating in the t-th auction. In the case of over-delivery, we lower $p(t)$ to reduce the chance of participating in the auction, thereby decreasing the likelihood of spending during this round. Mathematically, we can update $p(t)$ by multiplying it by $1 - \lambda_t$, where $\lambda_t > 0$ is a control parameter to adjust the throttling level. Conversely, if the campaign is under-delivered, we increase $p(t)$ by multiplying it by $1 + \lambda_t$. Mathematically, the update rule of $p(t)$ can be expressed as follows:

$$p(t) = \begin{cases} \min\{p(t-1) \cdot (1+\lambda_t), 1\} & \text{if } S(t) \le \alpha(t), \\ \max\{p(t-1) \cdot (1-\lambda_t), 0\} & \text{if } S(t) > \alpha(t). \end{cases}$$

This motivates the following Algorithm 1:

Algorithm 1 Throttling-based Budget Pacing Algorithm

```
Require: $B$: Total budget of the campaign
Require: $T$: Total number of auction opportunities
Require: $t$: Current auction round
Require: $S(t)$: Spend so far at $t$-th auction
Require: $p(t)$: Throttling probability at $t$-th auction
Require: $\{\lambda_t\}$: Control parameters for throttling adjustment
 1: Initialize $p(0) \leftarrow 1.0$ and $S(0) \leftarrow 0.0$
 2: for each auction at $t$-th auction do
 3:     Calculate target spend: $target\_spend\alpha(t) \leftarrow \frac{t}{T} \times B$
 4:     if $S(t) \leq \alpha(t)$ then
 5:         Increase throttling probability: $p(t) \leftarrow \min\{1.0, p(t) \cdot (1 + \lambda_t\}$
 6:     else
 7:         Decrease throttling probability: $p(t) \leftarrow \max\{0.0, p(t) \cdot (1 - \lambda_t\}$
 8:     end if
 9:     Generate a random number $r \in [0, 1]$
10:     if $r \leq p(t)$ then
11:         Participate in the auction and get the spend in current auction $c_t$
12:         Update spend: $S(t) \leftarrow S(t-1) + c_t$
13:     else
14:         Skip the auction
15:     end if
16: end for
```

In practice, to simplify the implementation, we may set λ_t as a constant, e.g. 10%, as in [1]. Also, there is no need to update $p(t)$ for every auction, we may set the update granularity to, say, 1 minute. More technical implementation details could be found in [1]. The regret analysis and the optimality of throttle-based pacing can be found in [20], which also includes a comparison between throttle-based pacing and bid-based pacing, both of which we will introduce in the subsequent sections.

2 Remarks

some history of bid pacing algorithm throtte-based to bid-based pacing. other applications of throtte-based pacing

CHAPTER 3

PID CONTROLLER

In this chapter, we discuss how to leverage the PID controller to design pacing algorithms. We begin with a brief introduction to the fundamental principles of the PID control method. We then apply this method to design pacing algorithms for two key problems: maximum delivery and cost cap optimization.

1 Introduction to PID Controllers

A Proportional-Integral-Derivative (PID) controller is a widely used feedback-based control loop mechanism. It is designed to maintain a desired output by minimizing the error $e(t)$, which is the difference between a desired setpoint $r(t)$ and the measured process variable $y(t)$. The PID controller achieves this by adjusting the control input $u(t)$ based on three terms: proportional, integral, and derivative. The output of a PID controller, $u(t)$, is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \tag{3.1}$$

where:

- $e(t) = r(t) - y(t)$ is the error signal,
- K_p is the proportional gain, controlling the response proportional to the error,
- K_i is the integral gain, reducing steady-state error by integrating the error over time,
- K_d is the derivative gain, predicting future error by calculating the rate of change of the error.

These three terms can be interpreted as follows:

- **Proportional term** $K_p e(t)$ provides an immediate response proportional to the current error.
- **Integral term** $K_i \int_0^t e(\tau)\, d\tau$ accumulates past errors, addressing steady-state error by applying corrective action based on the error history.
- **Derivative term** $K_d \frac{de(t)}{dt}$ predicts future errors by responding to the rate of change of the error.

In practice, the discrete form of (3.1) is often used for implementation:

$$u(t) = K_p e(t) + K_i \sum_{\tau=1}^{t} e(\tau)\Delta t + K_d \cdot \frac{e(t) - e(t-1)}{\Delta t}.$$

2 PID Contoller in Max Delivery

2.1 Main Algorithm

In previous chapter, we mentioned that, optimally, the amount of budget depleted within a time interval should be proportional to the number of auction opportunities during that time interval, i.e.,

$$\sum_{\tau \le t \le \tau + \Delta\tau} x_t c_t \propto \#\text{ of auction opportunities in } (\tau, \tau + \Delta\tau).$$

Remember that the optimal bid is the bid that just depletes the campaign's budget, assuming there is abundant supply. In practice, we can first construct the target spend per interval and use this target spend as the setpoint to apply the PID controller discussed above. Suppose we have a CPC campaign with a daily budget of $100. We bucketize one day by choosing the bucket interval $\Delta\tau = 15$ minutes and obtain the predicted number of eligible requests (auction opportunities) for these intervals from the prediction model, as shown in Figure 3.1.

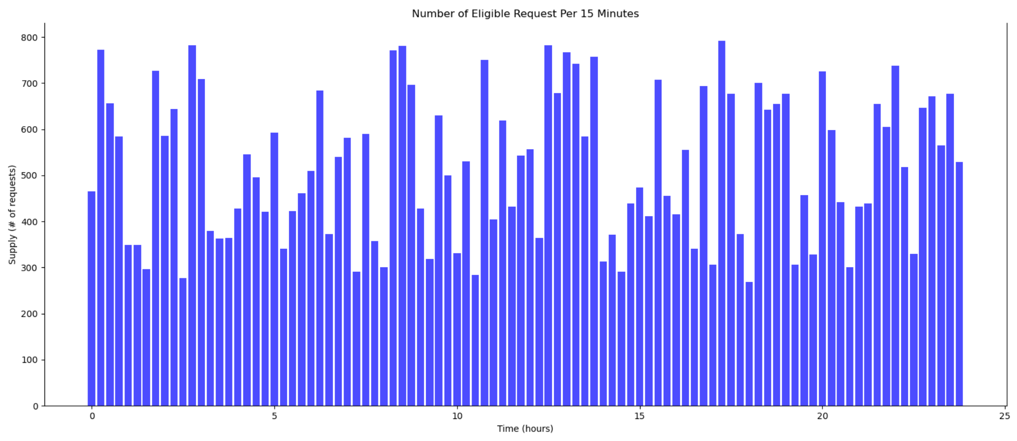


Figure 3.1: Supply Pattern (Number of Eligible Requests Per 15 Minutes)

Based on the supply pattern in Figure 3.1, we can construct the target spend per bucket(15-minute interval) for this campaign. For simplicity, within each target spend interval $\Delta\tau$, we assume the budget is consumed linearly over time. Suppose $TS(t)$ is the target spend in the t-th interval, $NR(t)$ is the number of eligible requests in the t-th interval, $B = 100$ is the total daily budget, and $T = 96$ is the number of target spend intervals. The proportional budget allocation rule per interval is given by:

$$TS(t) = \frac{NR(t)}{\sum_{s=1}^{T} NR(s)} \cdot B$$

The per-interval target spend is then computed and plotted in Figure 3.2:

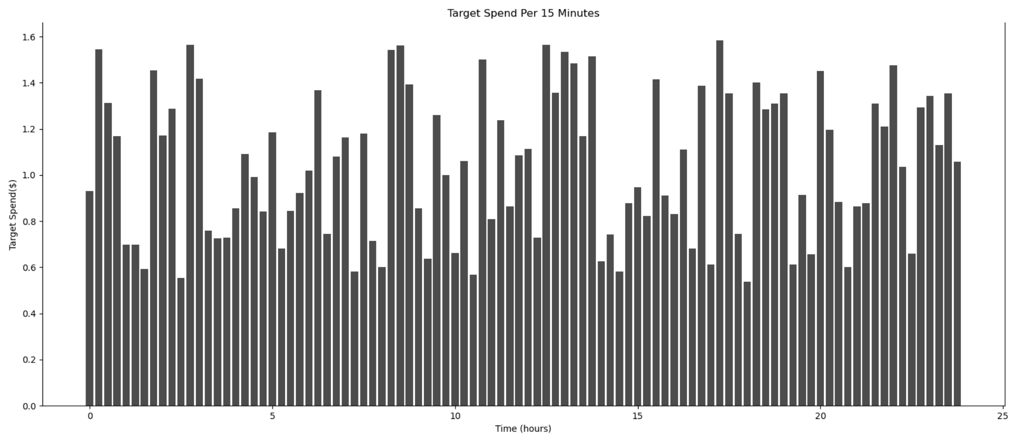


Figure 3.2: Target Spend Per 15 Minutes

We summarize the budget allocation algorithm as follows:

Algorithm 2 Compute Target Budget per Bucket

```
Require: $B$: Total daily budget
Require: $NR(t)$: Number of eligible requests in interval $t$
Require: $T$: Total number of buckets in a day
1: Initialize $TS(t) \leftarrow 0$ for all $t = 1, \dots, T$
2: Compute the total eligible requests:
       $$\text{TotalRequests} \leftarrow \sum_{t=1}^{T} NR(t)$$
3: for $t = 1$ to $T$ do
4:     Compute the proportional share of the budget for bucket interval $t$:
       $$TS(t) \leftarrow \frac{NR(t)}{\text{TotalRequests}} \cdot B$$
5: end for
6: return $TS(t)$ for all $t = 1, \dots, T$
```

Suppose the bid price gets updated every Δt time (Δt is a tunable bid update interval, which we may set to, e.g., 1 minute) at $t_0, t_1, t_2, \cdots, t_N$, where $N = \text{MinutesInOneDay}/\Delta t$. We define the error factor and control signal as:

$$e(t_k) = r_{t_k} - s(t_k),$$
$$u(t_k) \leftarrow K_p e(t_k) + K_i \sum_{j=1}^{k} e(t_j)\Delta t + K_d \frac{\Delta e(t_k)}{\Delta t},$$

where:

- r_{t_k}: Observed spend during the k-th pacing update interval,
- $s(t_k)$: Target spend derived by proportionally allocating within the target budget interval $\Delta\tau$ that contains it, e.g., if $[t_{k-1}, t_k]$ lies within the t-th target budget interval, then

  $$s(t_k) = \frac{\Delta t}{\Delta \tau} \cdot TS(t),$$

  where $TS(t)$ is the target spend for the t-th interval.

The bid can be updated by leveraging an actuator that takes the current control signal $u(t_k)$ to adjust the current bid price $b(t_k)$ as:

$$b(t_{k+1}) \leftarrow b(t_k)\exp\{u(t_k)\}.$$

The PID algorithm for max delivery problem is summarized in Algorithm 3. For more in-depth knowledge on applying PID controllers to the max delivery pacing problem, one may refer to [72], [64] and [75].

2.2 Bidding Dynamics

We show here the detailed bidding dynamics for this campaign. Figure 3.3 is a simplified plot demonstrating how the PID controller operates in a real-world dynamic environment.

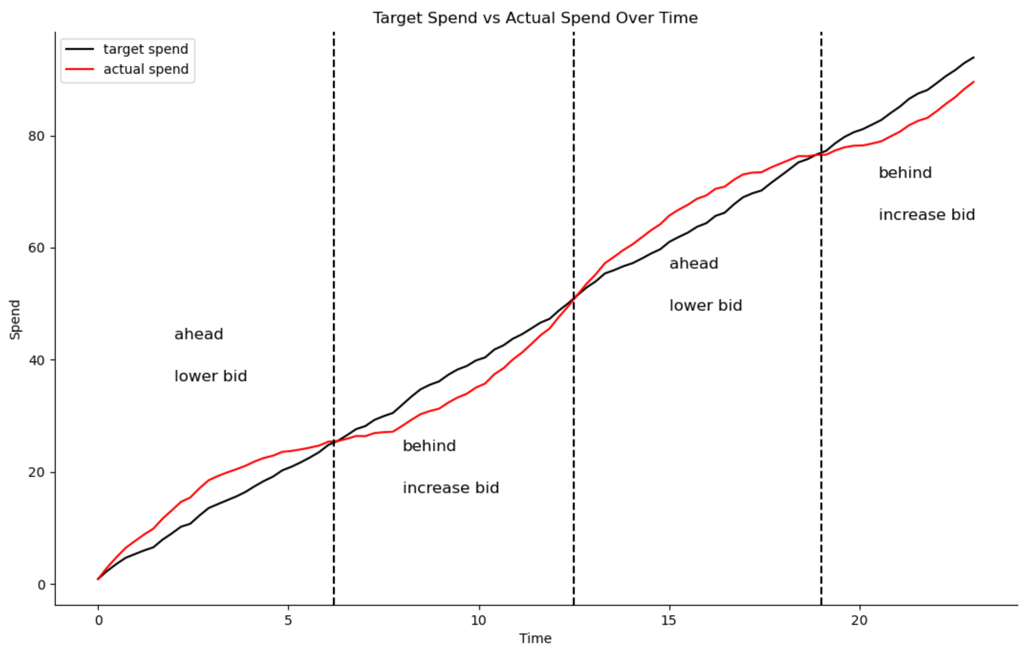


Figure 3.3: Target Spend vs Actual Spend Over Time

Algorithm 3 PID Controller for Max Delivery

```
Require: $\Delta t$: Time interval for bid updates(e.g. 1 minute)
Require: $T$: Total campaign duration
Require: $B$: Total campaign budget
Require: $\Delta\tau$: Time interval of target budget per bucket
Require: $\{TS(t)\}$: Target spend for each target budget interval $t$
Require: $K_p, K_i, K_d$: PID controller gains
 1: Initialize $b(t_0) \leftarrow$ initial_bid
 2: Initialize $u(t_0) \leftarrow 0$
 3: Initialize cumulative error: $CE \leftarrow 0$
 4: Initialize previous error: $PE \leftarrow 0$
 5: for $k = 1$ to $N$ do                                  ▷ $N =$ MinutesInOneDay$/\Delta t$
 6:     Find the $t$-th target budget interval that contains $t_k$ update interval, compute:
            $$s(t_k) \leftarrow \frac{\Delta t}{\Delta\tau} \cdot TS(t)$$
 7:     Measure observed spend during the interval $r(t_k)$
 8:     Compute the error factor:
            $$e(t_k) \leftarrow s(t_k) - r(t_k)$$
 9:     Update the cumulative error:
            $$CE \leftarrow CE + e(t_k) \cdot \Delta t$$
10:     Update the control signal using the PID formula:
            $$u(t_k) \leftarrow K_p \cdot e(t_k) + K_i \cdot CE + K_d \cdot \frac{e(t_k) - PE}{\Delta t}$$
11:     Update the per click bid price:
            $$b(t_k) \leftarrow b(t_{k-1}) \cdot \exp(u(t_k))$$
12:     Update the previous error:
            $$PE \leftarrow e(t_k)$$
13:     Get click through rate $r_t$ from the prediction model
14:     Compute the per impression bid
            $$b_t = b(t_k) \cdot r_t$$
15: end for
```

The dynamics of a PID controller adjusts bids based on the relationship between the **target spend** (black line) and the **actual spend** (red line) over time. Target spend represents the ideal cumulative budget spending at any point in time to evenly distribute the budget. Actual spend reflects the real cumulative spend achieved by the campaign over time. The graph highlights two key conditions: being **ahead of the schedule** or **behind the schedule**, and how these conditions affect bid modulation.

If the delivery is behind the schedule, i.e. when the actual spend (red line) is below the target spend (black line), the campaign is under-delivering. The PID controller **increases the bid** to catch up with the target spend. Higher bids make the campaign more competitive in auctions, increasing the likelihood of winning more impressions and spending more. In the plot, the actual spend curve slopes upward more steeply after the "increase bid" label in the "behind" regions; If the delivery is ahead of the schedule, i.e. when the actual spend (red line) exceeds the target spend (black line), the campaign is over-delivering. The PID controller **lowers the bid** to slow down the spending rate and realign with the target spend. Lower bids reduce the competitiveness of the campaign in auctions, resulting in fewer impressions and slower spend. In the plot, the actual spend curve becomes less steep after the "lower bid" label in the "ahead" regions.

This feedback mechanism ensures a balanced distribution of the budget over time, optimizing performance while adhering to pacing constraints.

2.3 Practical Considerations

We list here some practical considerations for implementing the PID controller in real-world production sytem:

- **Constructing budget allocating distribution**: aggregation across different dimensions, more details TBA.

- **Normalization of error signal**: The error factor $e(t_k)$ in Algorithm 3 is defined as $r(t_k) - s(t_k)$, the gap between actual spend and target spend. However, the issue is that the scale of the budget varies significantly across different campaigns, requiring different controller gains to accommodate these budget fluctuations. In practice, it is challenging to maintain different controller gains for each campaign, and it is more common for all ad campaigns to share the same controller gains. In this situation, a better approach is to compute the error factor in a normalized way, i.e.,

$$e(t_k) = 1 - \frac{r(t_k)}{s(t_k)},$$

  which normalizes the error factors for campaigns with different budget scales to a common scale.

- **Different Actuator**: The update rule in Algorithm 3 is $b(t_k) \leftarrow b(t_{k-1}) \cdot \exp(u(t_k))$, where the exp function is chosen as the actuator. In practice, other functions can also be used as the actuator function, such as linear functions or sigmoid functions.

- **Satruated Control**: to avoid drastic bid updates

- **Choice of $\Delta\tau$ and Δt**: Some trade-offs should be considered when choosing the target spend bucket $\Delta\tau$ and the bid update interval Δt.
  - **Target Spend Bucket $\Delta\tau$**:
    - A small $\Delta\tau$ provides finer-grained predictions of the supply pattern, reducing the accuracy requirements for interpolation within the bucket. However, the prediction itself may become noisier.
    - A large $\Delta\tau$, on the other hand, gives a more accurate estimate of the overall supply. However, the assumption of a linear distribution of supply within $\Delta\tau$ is less likely to hold, which can negatively impact interpolation accuracy.
  - **Bid Update Interval Δt**:
    - A small Δt results in more responsive updates to the market, but the computed error factor may contain more noise.
    - A large Δt helps mitigate statistical noise in the error factor, but delayed bid updates may fail to respond to market changes in a timely manner.

  These two parameters can be tuned online. In practice, we find that setting $\Delta\tau$ to a duration ranging from a few minutes to an hour, and setting Δt to a comparable range, typically works well.
- **CPC vs oCPM**: discuss spend signal delay
- **Uncertainty of T**: [7], [28], [4]
- **Non-i.i.d. Distribution**: [25]

3 PID Contoller in Cost Cap

We present several PID control-based algorithms to solve the cost cap problem.

3.1 Cost-Min Alogrithm

The idea behind the "Cost-Min" algorithm is to adaptively compute the maximum bid that can be submitted to achieve the cost control goal and set this as the upper bound for the normal bid price, which is computed by only considering the budget constraint. Under the configuration of Equation 1.4, a naive implementation of the algorithm is as follows: the total budget is B and the upper bound of the average cost per click is C. This means we need to collect at least $N = B/C$ clicks if the budget is completely depleted. At any time t, suppose the actual accumulated spend is S_t and the accumulated number of observed clicks is N_t. The remaining budget is $B - S_t$, and the remaining click goal is $N - N_t$. The new upper bound of the average cost for the remaining delivery is then:

$$U_t = \frac{B - S_t}{N - N_t}.$$

Suppose b_t is the bid price derived from the max-delivery algorithm without the cost constraint. The "Cost-Min" algorithm sets the final bid as:

$$\hat{b}_t = \min\{b_t, U_t\}.$$

During each bid update interval Δt, we collect the signals for the actual spend and the number of conversions (in this case, clicks), derive the remaining budget and the remaining target number of conversions, and compute the up-to-date upper bound for the bid. At the same time, we update the delivery bid as if there were no cost constraints using the PID controller discussed in the previous section. The new bid is then set to the minimum of the delivery bid and the upper bound bid. The "Cost-Min" algorithm is summarized in Algorithm 4.

Algorithm 4 Cost-Min Algorithm for Cost Cap

```
Require: $B$: Total budget, $C$: Upper bound for the average cost per click, $\Delta t$: Bid update interval
Require: $b_t$: Bid price computed by max-delivery without cost constraint
Ensure: $\hat{b}_t$: Final bid price considering cost cap constraint
 1: Compute the total click target: $N = \frac{B}{C}$
 2: Initialize: $U_0 \leftarrow C$                                ▷ Initial Bid Upper Bound
 3: Initialize: $S_0 \leftarrow 0$, $N_0 \leftarrow 0$                    ▷ Accumulated spend and clicks
 4: for each bid update interval $\Delta t$ do
 5:     Observe the spend for the interval: $\Delta S_t$
 6:     Observe the number of clicks for the interval: $\Delta N_t$
 7:     Update accumulated spend: $S_t \leftarrow S_{t-1} + \Delta S_t$
 8:     Update accumulated clicks: $N_t \leftarrow N_{t-1} + \Delta N_t$
 9:     Compute remaining budget: $B_\mathrm{r} = B - S_t$
10:     Compute remaining click goal: $N_\mathrm{r} = N - N_t$
11:     Compute the updated upper bound for the average cost:
            $$U_t = \frac{B_\mathrm{r}}{N_\mathrm{r}}$$
12:     Update the delivery bid $b_t$ without considering cost constraints using PID control(e.g., Algorithm 3)
13:     Compute the final bid price:
            $$\hat{b}_t = \min\{b_t, U_t\}$$
14: end for
15: return $\hat{b}_t$
```

The upper bound U_t discussed here represents the target cost per result after time t. If we bid using U_t and win the auction, then under a second-price auction mechanism, the actual payment per impression corresponds to $eCPM_2$, the second-highest eCPM. Effectively, the average cost per conversion is given by:

$$\frac{eCPM_2}{CTR_1},$$

where CTR_1 denotes the click-through rate (CTR) of this campaign. We can rewrite this expression as:

$$\frac{eCPM_2}{CTR_1} = \frac{eCPM_2}{CTR_1 \cdot U_t} \cdot U_t = \frac{eCPM_2}{eCPM_1} \cdot U_t = \sigma \cdot U_t,$$

where σ represents the ratio between the highest eCPM and the second-highest eCPM. Consequently, the actual cost per click (CPC) is given by:

$$\sigma \cdot U_t,$$

which is lower than the upper bound U_t. If we assume that the ratio σ remains relatively stable over time, we can relax this constraint slightly. In the algorithm described above, we may compute U_t as:

$$U_t = \frac{B_r}{N_r \cdot \sigma},$$

For more technical details on the "Cost-Min" algorithm, one may refer to [40].

3.2 Dynamic Cap

The "Cost-Min" algorithm is static and conservative in some sense. U_t represents the upper bound for the average cost per conversion in the remaining delivery schedule. For a single auction opportunity, the bid could go higher as long as the average cost is controlled under the target cap. In practice, we can make the cap more dynamic and responsive to the real-time cost control quality.

To understand how we should dynamically tweak the bid in real-time, we ignore the budget constraint and consider only the optimization problem with the cost control constraint. Specifically, we want to explore how the upper bound of the bid can impact conversions and cost control. Suppose u is the upper bound; the number of conversions for an auction opportunity given u is a random variable denoted by $M_r(u)$, and the cost per conversion for this auction is $M_c(u)$. Our goal is to solve the following optimization problem:

$$\begin{aligned} \max_{u} \quad & \mathbb{E}\left[M_r(u)\right] \\ \text{s.t.} \quad & \mathbb{E}\left[M_c(u)\right] \leq C \end{aligned}$$

where C is the target cost cap.

Both $\mathbb{E}\left[M_r(u)\right]$ and $\mathbb{E}\left[M_c(u)\right]$ are monotonically non-decreasing with respect to u (i.e., the higher the bid, the more likely you are to win the auction, and the higher the price you pay). The solution to the optimization problem is given by:

$$\mathbb{E}\left[M_c(u)\right] = C.$$

This is equivalent to solving:

$$u^* = \arg\min_u \mathbb{E}\left[M_c(u) - C\right]^2. \tag{3.2}$$

To solve this, we can apply the Robbins-Monro algorithm (see [58]) and iteratively update u using:

$$u \leftarrow u - \epsilon \cdot \nabla_u \left[M_c(u) - C\right]^2 = u - \epsilon \cdot 2M_c'(u) \cdot \left[M_c(u) - C\right],$$

where $M_c'(u) \geq 0$ because $M_c(u)$ is monotonically non-decreasing. Setting $\epsilon'(u) = \epsilon \cdot 2M_c'(u) \geq 0$, the update rule becomes:

$$u \leftarrow u - \epsilon'(u) \cdot \left[M_c(u) - C\right].$$

This is essentially a proportional controller (P-controller). When the actual cost per acquisition (CPA) is less than the target C, we increase the dynamic bid cap (upper bound); otherwise, we decrease the bid cap.

Based on the analysis above, we design the algorithm for the dynamic bid cap as follows: for every update interval Δt, we collect the actual spend and actual conversions, compute the actual average cost per conversion, and based on the gap between the actual CPA and the target cost cap, we update the upper bound using a proportional controller (for simplicity, in the actual implementation, we may choose a fixed ϵ as the controller gain). The dynamic bid cap variant of the "Cost-Min" algorithm is summarized in the following Algorithm 5:

Algorithm 5 Dynamic Bid Cap Variant of "Cost-Min" Algorithm

```
Require: $B$: Total budget, $C$: Target cost cap, $\Delta t$: Bid update interval
Require: $b_t$: Delivery bid computed by max-delivery without cost constraint
Require: $\epsilon$: Controller gain for the proportional controller
Ensure: $\hat{b}_t$: Final bid price considering cost cap constraint
 1: Compute the total click target: $N = \frac{B}{C}$
 2: Initialize: $S_0 \leftarrow 0$, $N_0 \leftarrow 0$, $u \leftarrow C$        ▷ Accumulated spend, clicks, and initial bid cap
 3: for each bid update interval $\Delta t$ do
 4:     Observe the spend for the interval: $\Delta S_t$
 5:     Observe the number of conversions for the interval: $\Delta N_t$
 6:     Update accumulated spend: $S_t \leftarrow S_{t-1} + \Delta S_t$
 7:     Update accumulated conversions: $N_t \leftarrow N_{t-1} + \Delta N_t$
 8:     Compute the actual CPA: $\text{CPA}_t = \frac{S_t}{N_t}$
 9:     Update the dynamic bid cap using the proportional controller:
            $u \leftarrow u - \epsilon \cdot (\text{CPA}_t - C)$
10:     Ensure the bid cap is non-negative: $u \leftarrow \max(u, 0)$
11:     Compute the final bid price:
            $\hat{b}_t = \min\{b_t, u\}$
12: end for
13: return $\hat{b}_t$
```

3.3 Dual-PID

Recall that the optimal per-click bid for cost cap is given by

$$b^*_{click} = \frac{1 + \mu^* C}{\lambda^* + \mu^*},$$

where λ and μ are the dual parameters associated with the budget and cost constraints, respectively. The KKT conditions imply that when these constraints are active, we have

$$\sum_{t=1}^{T} x_t \cdot c_t = B, \quad \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} = C.$$

The optimal dual parameters are those for which the pacing algorithm exactly depletes the total budget and the cost per click equals the cost cap. This insight motivates the design of a PID controller to update λ and μ.

For each pacing interval t_k, we collect the spend $r(t_k)$ and the number of clicks $n(t_k)$ for the campaign, and compute the average cost per click as

$$c(t_k) = \frac{r(t_k)}{n(t_k)}.$$

We then compare the actual spend $r(t_k)$ and the actual cost per click $c(t_k)$ to the target spend $s(t_k)$ (computed based on the traffic pattern as described in the previous section) and the target cost per click C (the cost cap), respectively, to define the error signals for λ and μ:

- Budget constraint error for λ:
$$e_\lambda(t_k) = r(t_k) - s(t_k)$$
- Cost constraint error for μ:
$$e_\mu(t_k) = C - \frac{r(t_k)}{n(t_k)}$$

The PID update equations for λ and μ are then defined as

$$u_\lambda(t_k) = K_{p,\lambda}\, e_\lambda(t_k) + K_{i,\lambda} \sum_{j=0}^{k} e_\lambda(t_j) + K_{d,\lambda} \frac{e_\lambda(t_k) - e_\lambda(t_{k-1})}{\Delta t},$$
$$u_\mu(t_k) = K_{p,\mu}\, e_\mu(t_k) + K_{i,\mu} \sum_{j=0}^{k} e_\mu(t_j) + K_{d,\mu} \frac{e_\mu(t_k) - e_\mu(t_{k-1})}{\Delta t}.$$

The control signals u_λ and u_μ are then applied to update the dual parameters:

$$\lambda \leftarrow \lambda \cdot \exp(u_\lambda), \quad \mu \leftarrow \mu \cdot \exp(u_\mu).$$

The new bid per click is computed as

$$b_{click} = \frac{1 + \mu C}{\lambda + \mu}.$$

We summarize this algorithm in Algorithm 6.

Algorithm 6 PID Controller for Cost Cap

```
Require: $\Delta t$: Time interval for bid updates(e.g., 1 minute), $\Delta\tau$: Time interval of target budget per bucket
Require: $T$: Total campaign duration, $B$: Total campaign budget, $C$: Cost per click cap
Require: $\{TS(t)\}$: Target spend for each target budget interval $t$
Require: $K_{p,\lambda}, K_{i,\lambda}, K_{d,\lambda}$: controller gains for budget constraint $\lambda$
Require: $K_{p,\mu}, K_{i,\mu}, K_{d,\mu}$: controller gains for cost constraint $\mu$
 1: Initialize $\lambda(t_0) \leftarrow \lambda_0, \mu(t_0) \leftarrow \mu_0$
 2: Initialize cumulative errors: $CE_\lambda \leftarrow 0, CE_\mu \leftarrow 0$
 3: Initialize previous errors: $PE_\lambda \leftarrow 0, PE_\mu \leftarrow 0$
 4: for $k = 1$ to $N$ do                                  ▷ $N = \text{MinutesInOneDay}/\Delta t$
 5:     Find the $t$-th target budget interval that contains $t_k$ update interval, compute:
            $$s(t_k) \leftarrow \frac{\Delta t}{\Delta\tau} \cdot TS(t)$$
 6:     Observe spend $r(t_k)$ and click $n(t_k)$ during the interval
 7:     Compute the error factor for $\lambda$ and $\mu$
            $$e_\lambda(t_k) \leftarrow r(t_k) - s(t_k), \quad e_\mu(t_k) \leftarrow C - \frac{r(t_k)}{n(t_k)}$$
 8:     Update the cumulative error:
            $$CE_\lambda \leftarrow CE_\lambda + e_\lambda(t_k) \cdot \Delta t, \quad CE_\mu \leftarrow CE_\mu + e_\mu(t_k) \cdot \Delta t$$
 9:     Update the control signal using the PID formula:
            $$u_\lambda(t_k) \leftarrow K_{p,\lambda} \cdot e_\lambda(t_k) + K_i \cdot CE_\lambda + K_d \cdot \frac{e_\lambda(t_k) - PE_\lambda}{\Delta t}$$
            $$u_\mu(t_k) \leftarrow K_{p,\mu} \cdot e_\mu(t_k) + K_i \cdot CE_\mu + K_d \cdot \frac{e_\mu(t_k) - PE_\mu}{\Delta t}$$
10:     Update the dual variables:
            $$\lambda(t_k) \leftarrow \lambda(t_{k-1}) \cdot \exp(u_\lambda(t_k)), \quad \mu(t_k) \leftarrow \mu(t_{k-1}) \cdot \exp(u_\mu(t_k))$$
11:     Update the previous error:
            $$PE_\lambda \leftarrow e_\lambda(t_k), \quad PE_\mu \leftarrow e_\mu(t_k)$$
12:     Get conversion rate $r_t$ from predction model
13:     Compute the new per impression bid price:
            $$b_t = \frac{1 + \mu(t_k)C}{\lambda(t_k) + \mu(t_k)} \cdot r_t$$
14: end for
```

3.4 Remarks

- non-linear interaction
- signal delay
- second price ratio for cost-min algorithm
- parameter tuning

CHAPTER 4

MPC CONTROLLER

In this chapter, we discuss the Model Predictive Control (MPC) framework. MPC leverages predictive modeling to plan future actions, enabling more robust control performance. We begin with a brief introduction to MPC, followed by its application to both the maximum delivery and cost cap problems. Additionally, we explore target CPA bidding, a bidding product similar to cost cap, and demonstrate how MPC can be applied to optimize this new bidding strategy.

1 Introduction to Model Predictive Control

1.1 From PID to MPC

In the previous chapter, we explored how to apply PID control to solve both the max delivery and cost cap problems. While PID controllers provide a simple and effective framework for designing pacing algorithms, they lack the ability for future planning, making them sometimes myopic and leading to unstable pacing dynamics. This can be detrimental to delivery performance, especially in complex problems like cost cap, where multiple constraints must be taken into account.

Model Predictive Control (MPC) is a more advanced framework that addresses these challenges by incorporating predictive modeling and optimization into the control process. Unlike PID controllers, which rely solely on feedback to correct errors, MPC leverages a dynamic model of the system to predict its future states over a defined time horizon. This forward-looking capability enables MPC to make informed decisions that proactively optimize performance while respecting constraints such as daily budget limits, cost caps, and pacing targets.

To illustrate the difference between PID control and MPC control, consider a simple example: imagine driving a car and aiming to maintain a speed of 60 km/h on a highway. However, the terrain is constantly changing—there are uphill slopes, downhill stretches, and areas of heavy traffic. How can we ensure that the speed stays close to the target?

A PID controller works reactively. It adjusts the accelerator and brake based on the difference (error) between the current speed and the target speed (proportional term), how long this error has persisted (integral term), and how quickly the error is changing (derivative term). While effective in many situations, PID cannot predict future changes in terrain or traffic. It reacts only after the error occurs, potentially leading to delays in adjusting to a steep hill or a sudden slowdown ahead.

Now consider an MPC framework. Instead of reacting only to the current speed, MPC incorporates a model of the car's dynamics and the environment. It predicts how the car's speed will change over the next several seconds based on factors like engine power, road gradient, and traffic conditions. With this foresight, MPC calculates the optimal sequence of accelerator and brake adjustments to keep the speed close to 60 km/h over the next few seconds while respecting constraints (e.g., not exceeding speed limits or ensuring smooth acceleration).

At each step, MPC forecasts the car's speed trajectory based on current inputs (e.g., accelerator position, road incline) and system dynamics. It then solves an optimization problem to find the best control actions (accelerator/brake adjustments) that minimize the deviation from the target speed over a finite time horizon. The first control action from the optimized sequence is applied, and the process repeats in the next time step with updated data.

In the context of ad pacing, think of the "car" as the ad campaign, the "target speed" as the daily spend goal, and the "terrain" as auction dynamics and traffic fluctuations. Just as driving on a changing road requires continuous adjustments to speed, budget pacing requires continuous adjustments to bid amounts and budget allocations. A PID controller can help maintain spend close to the target but struggles when traffic conditions change rapidly or constraints need to be enforced. MPC, with its predictive capabilities, anticipates traffic patterns and optimizes bidding strategies ahead of time, ensuring smoother and more efficient pacing.

This analogy demonstrates how MPC brings a forward-looking, constraint-aware, and adaptive approach to control problems, making it a superior choice for complex scenarios like real-time ad pacing.

1.2 General Procedure of the MPC Method

The general procedure of MPC control is as follows: at each time step, at each time step, we solve an optimization problem to determine a plan of action over a fixed time horizon and then apply the first input from this plan (this approach is known as Receding Horizon Control, or RHC). At the next time step, the process is repeated—solving a new optimization problem with the time horizon shifted one step forward.

The key advantage of RHC is its ability to handle constraints directly while requiring significantly less parameter tuning (e.g., controller gains) compared to conventional control methods such as PID control. In essence, RHC allows constraints to be explicitly incorporated into the optimization process, whereas PID control requires extensive manual tuning of controller gains to indirectly manage constraints.

For a more in-depth discussion of MPC, we encourage readers to refer to [71], [16], or [51].

2 MPC Controller for Max Delivery

In PID control, supply predictions are used to compute the target trajectory (e.g., expected per bucket spend targets based on predicted supply pattern). This adjusts the desired reference signal, but the PID controller still operates reactively to minimize the error between the actual spend rate and the predicted target spend rate. The MPC approach, on the other hand, directly integrates predictions into the control process, enabling proactive adjustments that optimize campaign performance over a prediction horizon.

More specifically, in the max delivery problem, we use the same ad configuration: the campaign has a daily budget B and a predicted number of auction opportunities T. At a specific timestamp, suppose the consumed budget is B_t and the observed auction count is t. The predicted future target spend rate for the remainder of the day can be computed as:

$$TS(t) = \frac{B - B_t}{T - t}.$$

We aim to answer the question: how should we adjust the bid so that the corresponding spend rate matches $TS(t)$?

The PID controller compares the currently observed spend rate to the reference $TS(t)$. If the observed spend rate is lower than the target, the PID controller increases the bid. However, the magnitude of this adjustment depends on the error signal and the controller gains (proportional, integral, and derivative components).

MPC, on the other hand, directly models the relationship between bid and spend rate. This model belongs to the class of **bid landscape forecasting models**, which predict the distribution of winning bid prices in online auctions. Suppose we have such a model $s = f(b)$ that maps a bid b to a spend rate s. Given a target spend rate $TS(t)$, the bid can be computed as:

$$b_t = f^{-1}\left(TS(t)\right).$$

Next, we discuss how to model f for the max delivery problem:

2.1 Online Methods

The online auction data (e.g., bid, spend, impressions, etc.) reflects the dynamics of both the ad campaign itself and the marketplace. Therefore, an intuitive idea is to leverage the most recent and fresh auction data from this campaign to project the future bid-spend relationship.

Longest Increasing Subsequence

Recall that, within each bid update interval Δt, the bid per click b_t remains unchanged. We may collect the most recent N interval bid-spend pairs $\{b_k, s_k\}$, where s_k represents the spend over the fixed interval Δt at time k (and thus can be considered as the spend rate).

The bid-spend rate relationship for the next interval Δt can then be learned from these N data points, assuming that the number of auction opportunities does not change significantly over small intervals of Δt. For a specific auction, if we bid higher, the spend should increase (or at least not decrease). Therefore, $f(b)$ should be a monotonically nondecreasing function. However, $\{b_k, s_k\}$ does not necessarily form a monotonic sequence, as the data points are collected from different time intervals. To preserve the monotonicity property, the most straightforward method is to manually extract the longest increasing subsequence (LIS) from $\{b_k, s_k\}$. We may maintain a dynamic list L to store the smallest ending values of increasing subsequences of different lengths, and an auxiliary array P to track predecessors for LIS reconstruction. The time complexity of this approach is $O(N \log N)$. Once the LIS is extracted, we can interpolate between adjacent b_k values to construct a piecewise linear monotonic function f. For any target spend rate $TS(t)$, the corresponding bid b_t can be computed as $f^{-1}\left(TS(t)\right)$. If $TS(t)$ is outside the range of observed spend rates, a simple extrapolation can be applied to compute the target bid. The idea discussed here is summarized in the following Algorithm 7:

Algorithm 7 Monotonic Bid-Spend Model for MPC Using LIS

```
Require: $N$: Number of most recent bid-spend pairs, $\Delta t$: Bid update interval, $TS(t)$: Target spend rate
Ensure: $b_t$: Bid value corresponding to $TS(t)$
1: Collect the most recent $N$ bid-spend pairs $\{b_k, s_k\}$, where $s_k$ is the spend rate over interval $\Delta t$
2: Extract the longest increasing subsequence $\{b'_k, s'_k\}$ using Algorithm 8
3: Construct the piecewise linear monotonic function $f(b)$:
     • Sort $\{b'_k, s'_k\}$ in ascending order of $b'_k$
     • For $b \in [b'_i, b'_{i+1}]$, interpolate linearly:
         $f(b) = s'_i + \frac{(s'_{i+1} - s'_i)}{(b'_{i+1} - b'_i)} \cdot (b - b'_i)$
4: Compute the bid $b_t$ by inverting $f$:
     • If $TS(t) < s'_1$:                                   ▷ Extrapolation below the range
         $b_t = \min\left(0, b'_1 + \frac{TS(t) - s'_1}{s'_2 - s'_1} \cdot (b'_2 - b'_1)\right)$
     • If $TS(t) > s'_m$ (where $m = |\{b'_k, s'_k\}|$):     ▷ Extrapolation above the range
         $b_t = b'_m + \frac{TS(t) - s'_m}{s'_m - s'_{m-1}} \cdot (b'_m - b'_{m-1})$
     • If $TS(t) \in [s'_i, s'_{i+1}]$:                      ▷ Interpolation within the range
         $b_t = b'_i + \frac{(b'_{i+1} - b'_i)}{(s'_{i+1} - s'_i)} \cdot (TS(t) - s'_i)$
```

Algorithm 8 Longest Increasing Subsequence (LIS)

```
Require: $\{b_k, s_k\}$: Sequence of bid-spend pairs
Ensure: $\{b'_k, s'_k\}$: Longest increasing subsequence
 1: Initialize an empty list $L$ to store indices of the LIS and an auxiliary array $P$ of size $N$
    to track predecessors
 2: for $i = 1$ to $N$ do
 3:     Perform binary search on $L$ to find the largest index $j$ such that $s_{L[j]} \leq s_k$
 4:     if $j$ exists then
 5:         Replace $L[j+1]$ with $i$
 6:     else
 7:         Append $i$ to $L$
 8:     end if
 9:     Update $P[i]$ with the index of $L[j]$ (or set $P[i] = -1$ if $j$ does not exist)
10: end for
11: Reconstruct the LIS by backtracking from the last index in $L$ using the array $P$
12: Return the bid-spend pairs corresponding to the LIS indices in $L$: $\{b'_k, s'_k\}$
```

Isotonic Regression and PAVA algorithm

Extracting the longest increasing subsequence (LIS) can help maintain the monotonicity properties of a bid spend-rate curve. However, this approach discards a significant number of data points, leading to data inefficiency and potentially undermining the predictive accuracy of the model. To leverage all available data while maintaining monotonicity, isotonic regression offers an effective solution.

Isotonic regression is a type of regression analysis designed for situations where the target variable is expected to be non-decreasing (or non-increasing) with respect to an independent variable. It imposes an order constraint on the data, ensuring the resulting function remains monotonic.

A key algorithm used for isotonic regression is the **Pool Adjacent Violators Algorithm (PAVA)**. PAVA efficiently solves isotonic regression problems by iteratively merging adjacent data points that violate the monotonicity constraint. It is computationally lightweight, operating in linear time $O(n)$, making it well-suited even for large datasets. For more technical details and implementations, one may refer to [5], [44].

Compared to the LIS algorithm, PAVA offers significant improvements in computational efficiency, reducing the time complexity from $O(n \log n)$ to $O(n)$. Additionally, PAVA utilizes all data points during the computation process, which reduces prediction noise and enhances model accuracy.

For further technical details, refer to the method described in [19]. The steps of PAVA and how it can be applied to our problem can be summarized in the following algorithms:

Algorithm 9 Pool Adjacent Violators Algorithm (PAVA) with Preserved Bid-Spend Pairs

```
Require: $\{b_k, s_k\}$: Input bid-spend pairs, where $b_k$ is the bid and $s_k$ is the spend rate, sorted
    such that $b_1 \leq b_2 \leq \cdots \leq b_n$
Ensure: $\{b'_k, s'_k\}$: Adjusted monotonic bid-spend pairs of the same length as the input
 1: Initialize $y'_i \leftarrow s_i$ for all $i = 1, \ldots, n$
 2: Initialize weights $w_i \leftarrow 1$ for all $i = 1, \ldots, n$
 3: Initialize $i \leftarrow 1$
 4: while $i < n$ do
 5:     if $y'_i > y'_{i+1}$ then                                    ▷ Check monotonicity
 6:         Merge $y'_i$ and $y'_{i+1}$:
                $y'_i \leftarrow \frac{w_i y'_i + w_{i+1} y'_{i+1}}{w_i + w_{i+1}}$
 7:         Update weights: $w_i \leftarrow w_i + w_{i+1}$
 8:         Remove $y'_{i+1}$ and $w_{i+1}$ from their respective arrays
 9:         Remove $b_{i+1}$ from the bid array
10:         Adjust indices: $i \leftarrow \max(1, i - 1)$
11:     else
12:         Increment index: $i \leftarrow i + 1$
13:     end if
14: end while
15: Expand the adjusted spend values to match the original length:
16: for $j = 1$ to $n$ do
17:     Assign each original bid $b_j$ the adjusted spend value $y'_i$ corresponding to its current
    segment
18: end for
19: Return $\{b_k, y'_k\}$: Adjusted monotonic bid-spend pairs
```

Algorithm 10 Monotonic Bid-Spend Model for MPC Using PAVA

```
Require: $\{b_k, s_k\}$: Recent bid-spend pairs (bid $b_k$ and spend rate $s_k$) sorted such that $b_1 \leq b_2 \leq \cdots \leq b_n$
Require: $TS(t)$: Target spend rate,
Ensure: $b_t$: Bid value corresponding to $TS(t)$
1: Apply Algorithm 9 to $\{b_k, s_k\}$ to compute monotonic spend rates $\{b_k, s'_k\}$
2: Compute the bid $b_t$:
     • If $TS(t) < s'_1$:                                    ▷ Extrapolation below range
           $$b_t = b_1 + \frac{TS(t) - s'_1}{s'_2 - s'_1} \cdot (b_2 - b_1)$$
     • If $TS(t) > s'_n$:                                    ▷ Extrapolation above range
           $$b_t = b_n + \frac{TS(t) - s'_n}{s'_n - s'_{n-1}} \cdot (b_n - b_{n-1})$$
     • If $TS(t) \in [s'_i, s'_{i+1}]$:                      ▷ Interpolation within range
           $$b_t = b_i + \frac{(b_{i+1} - b_i)}{(s'_{i+1} - s'_i)} \cdot (TS(t) - s'_i)$$
3: Return $b_t$
```

2.2 Offline Methods

The online algorithms described above are lightweight and well-suited for leveraging fresh data points that represent the up-to-date bid landscape distributions. However, for ad campaigns with deep funnel conversions (e.g., CPA and CPL ads), the conversion signals are often sparse and delayed, as they need to be collected from third-party platforms.

If the campaign is an oCPM campaign, there is no issue because it is charged by impressions, and the delay in spend information is negligible. However, for campaigns charged by actual results (e.g., CPA or CPL), we may encounter situations where the spend remains at zero for an extended period, followed by a sudden spike in spend when a conversion occurs. This results in highly non-smooth pacing dynamics, making it challenging to maintain consistent performance.

In such cases, the online algorithms may not be the optimal strategy. Instead, it becomes necessary to leverage historical data to model the bid spend-rate curve for the max delivery problem.

eCPM Distribution

Suppose the probability density function of the eCPM distribution of the marketplace for this campaign is $p(z)$, and the campaign-level average CTR is r. Under the Second Price Auction mechanism, the expected spend per impression given a bid per impression b_i can be computed as:

$$g(b_i) = \frac{\int_0^{b_i} zp(z)\mathrm{d}z}{\int_0^{b_i} p(z)\mathrm{d}z}. \tag{4.1}$$

In the following section 4, we will show that $g(\cdot)$, as defined, is monotonically non-decreasing with respect to b.

Now, suppose the target spend for the next interval Δt is $TS(t)$, and the number of auction opportunities within this interval is $NR(t)$. The bid spend-rate curve $f(b)$ (where b is the bid per click) can be computed as:

$$f(b) = g\left(b \cdot r\right) \cdot NR(t).$$

To achieve the target spend $TS(t)$, set $f(b_t) = TS(t)$ and solve:

$$TS(t) = g\left(b_t \cdot r\right) \cdot NR(t).$$

Rearranging gives:

$$b_t = \frac{g^{-1}\left(\frac{TS(t)}{NR(t)}\right)}{r}.$$

The question now boils down to finding the eCPM distribution $p(\cdot)$. This can be achieved through statistical modeling of historical auction data, where the distribution of observed eCPM values can be estimated using techniques such as constructing empirical histograms, kernel density estimation, or parametric methods, depending on the nature of the data.

To demonstrate how to derive $p(\cdot)$ for a campaign, we use the empirical histogram approach offline. First, for a given campaign, collect all auctions it participated in over the past K days. For each auction, record the eCPMs from the GSP ranking stage and denote these eCPMs as $\{v_i\}$. Define the range of eCPM values:

$$\overline{v} = \max_i v_i, \quad \underline{v} = \min_i v_i.$$

Discretize the range $[\underline{v}, \overline{v}]$ into N buckets, $\{[z_{j-1}, z_j]\}_{j=1}^N$, where N is a large number such that each bucket is small enough to represent the distribution accurately. Count the number of eCPMs falling into each interval $[z_{j-1}, z_j]$, denoted as n_j. Compute the discretized probability density function (p.d.f.) for each interval:

$$p_j = \frac{n_j}{\sum n_j}.$$

Once p_j is computed, $g(\cdot)$ can be calculated based on the buckets $\{[z_{j-1}, z_j]\}_{j=1}^N$. For $b \in [z_{j-1}, z_j]$, set:

$$g(b) = g_j,$$

where:

$$g_j = \frac{\sum_{l \le j} z_l \cdot p_l}{\sum_{l \le j} p_l}.$$

For any target spend $TS(t)$, to compute b_t, find the index k such that g_k is closest to $\frac{TS(t)}{NR(t)}$. Then set:

$$b_t = \frac{z_k}{r}.$$

We summarize the idea above in the following algorithms:

Algorithm 11 Compute p_j (Discretized p.d.f.)

```
Require: $v$: Historical eCPM values for the campaign
Require: $N$: Number of buckets for discretization
Ensure: $p_j$: Discretized p.d.f., $z$: Bucket boundaries
 1: Define $\overline{v} = \max(v)$, $\underline{v} = \min(v)$
 2: Discretize $[\underline{v}, \overline{v}]$ into $N$ buckets: $\{[z_{j-1}, z_j]\}_{j=1}^N$
 3: Initialize $n_j = 0$ for all $j = 1, \ldots, N$
 4: for each $v_i \in v$ do
 5:     Find the bucket index $j$ such that $z_{j-1} \leq v_i < z_j$
 6:     Increment $n_j \leftarrow n_j + 1$
 7: end for
 8: Compute the total number of samples: $\text{total\_n} = \sum_{j=1}^N n_j$
 9: for each bucket $j = 1, \ldots, N$ do
10:     $p_j = \frac{n_j}{\text{total\_n}}$
11: end for
12: return $p_j$, $z$
```

Algorithm 12 Compute $g(b)$ and b_t

```
Require: $p_j$: Discretized p.d.f.
Require: $z$: Bucket boundaries
Require: $r$: Campaign-level average CTR
Require: $TS$: Target spend rate
Require: $NR$: Number of auction opportunities
Ensure: $b_t$: Optimal bid per click
 1: Compute $g(b)$ for each bucket:
 2: for each bucket $j = 1, \ldots, N$ do
 3:     $g_j = \frac{\sum_{l \leq j} z_l \cdot p_l}{\sum_{l \leq j} p_l}$
 4: end for
 5: Find the index $k$ such that $g_k$ is closest to $\frac{TS}{NR}$
 6: Compute the optimal bid:
        $$b_t = \frac{z_k}{r}$$
 7: return $b_t$
```

2.3 Practical Considerations

- **Estimate $p(\cdot)$ using aggregated methods**

- **Other Estimation Methods for $p(\cdot)$:** Kernel Density Estimation, Parametric Methods

3 MPC Controller for Cost Cap

3.1 MPC for Cost Cap

In the previous section, we discussed the MPC control for the max delivery problem. Since the max delivery problem only has a budget constraint, solving the receding horizon problem requires only the bid-to-spend rate model $s = f(b)$, which helps adjust the bid based on the current delivery status to maximize the objective function.

The cost cap problem, however, introduces an additional cost constraint, adding complexity because the algorithm must now balance both budget pacing and cost efficiency.

Problem Formulation Recall the formulation of the cost cap problem from Equation 1.4:

$$\begin{aligned}
\max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot r_t \\
\text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \leq B, \\
& \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} \leq C.
\end{aligned}$$

At each time step τ, with a receding horizon H (e.g., 1 hour), we solve a model predictive control (MPC) problem iteratively to optimize the objective. To do so, we must determine the new budget $B_{\tau,H}$ and the cost per result cap $C_{\tau,H}$ for the time interval $(\tau, \tau + H)$.

Computing Budget and Cost Constraints We compute $B_{\tau,H}$ and $C_{\tau,H}$ based on the delivery status at time τ:

- **Computing $B_{\tau,H}$:** As discussed in Algorithm 2, the target budget per time interval is allocated proportionally to the expected number of auction requests in that interval. If at time τ, the remaining budget is $B_{\tau,r}$, the predicted total remaining auction requests is $TR_{\tau,r}$, and the predicted auction requests within $(\tau, \tau+H)$ is $NR_{\tau,H}$, then the budget for this interval is:

$$B_{\tau,H} = \frac{NR_{\tau,H}}{TR_{\tau,r}} \cdot B_{\tau,r}.$$

- **Computing $C_{\tau,H}$:** Suppose at time τ, we have observed NC_τ conversions. To satisfy the cost cap constraint, we must collect at least $B/C - NC_\tau$ conversions before the campaign ends if we aim to fully spend the total budget B. The cost per result upper bound for the remainder of the campaign, $C_{\tau,r}$, is given by:

$$C_{\tau,r} = \frac{B_{\tau,r}}{\frac{B}{C} - NC_\tau}.$$

Assuming the conversion rate follows an i.i.d. distribution, the proportional share of the number of conversions in $(\tau, \tau + H)$ should match the proportional share of the requests in this time interval. Thus, the new cost per result cap for $(\tau, \tau + H)$ is the same as $C_{\tau,r}$:

$$C_{\tau,H} = \frac{B_{\tau,r}}{\frac{B}{C} - NC_\tau}.$$

Receding Horizon Optimization Problem In addition to the new budget and cost constraints, practical implementations may impose a lower bound b_l and an upper bound b_u on the bid to prevent extreme values. The receding horizon optimization problem for $(\tau, \tau + H)$ is formulated as:

$$\begin{aligned} \max_{x_t \in \{0,1\}} \quad & \sum_{\tau \le t \le \tau+H} x_t \cdot r_t \\ \text{s.t.} \quad & \sum_{\tau \le t \le \tau+H} x_t \cdot c_t \le B_{\tau,H}, \\ & \frac{\sum_{\tau \le t \le \tau+H} x_t \cdot c_t}{\sum_{\tau \le t \le \tau+H} x_t \cdot r_t} \le C_{\tau,H}, \\ & b_l \le b_t \le b_u. \end{aligned} \tag{4.2}$$

From Equation 1.6, we have shown that the optimal solution for the cost cap problem above is a constant bid. Thus, to solve the receding horizon problem, we simply need to find a constant bid between b_l and b_u such that the resulting spend and cost per result within $(\tau, \tau + H)$ satisfy the constraints in the optimization problem.

For the max delivery problem, the bid-to-spend model $s = f(b)$ is sufficient to determine the optimal bid given a target spend rate. However, for the cost cap problem, an additional bid-to-cpx model $cpx = h(b)$ is required to capture the relationship between bid price and cost per result. This enables the algorithm to balance both budget pacing and cost efficiency for $(\tau, \tau + H)$.

To construct $h(b)$, it suffices to model the bid-to-number-of-conversions function $n = g(b)$ and compute:

$$cpx = \frac{s}{n} = \frac{f(b)}{g(b)}.$$

Constructing $f(b)$ and $g(b)$ Recall that in the max delivery problem, $s = f(b)$ is constructed by first collecting the most recent N interval bid-spend pairs and then applying the longest increasing subsequence algorithm (e.g., Algorithm 8) or PAVA algorithm (e.g., Algorithm 9) and interpolation methods to obtain a monotonic bid-spend sequence.

The bid-to-number-of-conversions model $n = g(b)$ can be constructed similarly:

- Collect the most recent N pacing intervals and extract $\{b_k, n_k\}$, where n_k represents the number of conversions over the fixed interval k.
- Apply LIS or PAVA algorithms to $\{b_k, n_k\}$ to obtain a monotonic function $n = g(b)$.
- Renormalize $f(b)$ and $g(b)$ to represent spend and conversions per H-time interval.

Finding the Optimal Bid Since the bid-to-number-of-conversions function $n = g(b)$ is monotonically non-decreasing, the larger the bid, the more conversions we may get. With $h(b) = f(b)/g(b)$, the optimal bid b^* can be determined by iterating over all possible values of b and selecting the largest value satisfying the constraints:

$$f(b) \le B_{\tau,H}, \quad h(b) \le C_{\tau,H}, \quad b_l \le b \le b_u.$$

In practice, we can start with b_l and increment b by a small Δb at each step in the search process[1]. The idea discussed here is summarized in Algorithm 13.

Practical Considerations We list some practical considerations here for implementing cost cap MPC controller method in real-world production system:

- **Data Sparsity and Signal Delay**: The MPC controller method discussed here assumes that the campaign follows an oCPM bidding strategy, where the campaign is charged per impression. It also assumes that a sufficient number of conversions can be observed within each pacing update cycle. This assumption is crucial because it allows the use of real-time data and LIS/PAVA algorithms to model the bid-to-spend function f and the bid-to-number-of-conversions function g.

  For upper-funnel objective campaigns such as CPM and CPV, this assumption generally holds, as the time from ad exposure to conversion (e.g., impressions, video views) is only a few seconds, making any delay negligible. However, for mid-to-lower funnel objective campaigns such as CPC, CPL, or CPA, this assumption no longer holds. The delay from an impression to the final conversion (e.g., app installs) can take hours or even days. As a result, leveraging real-time conversion data to model g becomes impractical.

  Additionally, if the campaign is not oCPM but CPX, where charges only occur when actual conversions are realized, the cost pattern can be highly irregular. In such cases, there may be long periods with no cost, followed by sudden large charges when conversions happen. This causes the bid-to-spend function f to resemble a discontinuous "bump function," making it inappropriate to use real-time data for modeling.

  In these scenarios, a common approach is to leverage prediction models to construct f, g, and h. These predictive techniques provide a more stable and reliable estimation

[1]Theoretically, the bid-to-spend f and bid-to-cpx h are both monotonically non-decreasing with respect to the bid price b. Thus, starting from b_l, we can stop the search as soon as the first b satisfies the constraints and $b + \Delta b$ does not. This approach significantly reduces computation overhead. However, in practice, the monotonicity property does not always hold due to various factors, such as modeling errors or noisy data. As a result, enumerating all possibilities can sometimes help achieve better results, especially when the iteration process is not overly time-consuming.

Algorithm 13 MPC-Based Cost Cap Bidding Algorithm

```
Require: $B_{\tau,r}$: Remaining budget; $B$: Total budget; $C$: Cost cap;
 1: $H$: Time horizon; $NC_\tau$: Observed conversions; $TR_{\tau,r}$: Predicted total remaining requests;
 2: $NR_{\tau,H}$: Predicted requests in $(\tau, \tau + H)$; $[b_l, b_u]$: Bid bounds; $\Delta b$: Search step size.
Ensure: $b^*$: Optimal bid for $(\tau, \tau + H)$.
 3: Step 1: Compute Budget and Cost Cap Constraints
 4: Compute the budget allocation for the receding horizon:
        $$B_{\tau,H} \leftarrow \frac{NR_{\tau,H}}{TR_{\tau,r}} \cdot B_{\tau,r}$$
 5: Compute the cost per result upper bound for the remaining time:
        $$C_{\tau,r} \leftarrow \frac{B_{\tau,r}}{\frac{B}{C} - NC_\tau}$$
 6: Set the cost cap for the horizon:
        $$C_{\tau,H} \leftarrow C_{\tau,r}$$
 7: Step 2: Construct Models $f(b)$ and $g(b)$
 8: Collect the most recent $N$ bid-spend pairs $\{b_k, s_k\}$ and apply LIS or PAVA to construct
    $f(b)$ normalized to $H$.
 9: Collect the most recent $N$ bid-conversion pairs $\{b_k, n_k\}$ and apply LIS or PAVA to con-
    struct $g(b)$ normalized to $H$.
10: Compute $h(b)$ as:
        $$h(b) \leftarrow \frac{f(b)}{g(b)}$$
11: Step 3: Search for Optimal Bid $b^*$
12: Initialize $b^* \leftarrow b_l$.
13: for $b$ from $b_l$ to $b_u$ with step size $\Delta b$ do
14:     if $f(b) \leq B_{\tau,H}$ and $h(b) \leq C_{\tau,H}$ then
15:         Update $b^* \leftarrow b$.
16:     end if
17: end for
18: Step 4: Return Optimal Bid
19: return $b^*$
```

of bid-to-spend and bid-to-conversion relationships. We will explore these methods in greater detail in the bid landscape forecasting section in the next part.

- **Bid-to-X Model Normalization**: Recall that in the algorithm, we construct f and g based on the bid-cost and bid-conversion pairs observed during each pacing interval. As a result, f and g represent the total spend and number of conversions over the pacing interval. However, if the receding horizon H differs from the pacing interval (which is usually the case), we must normalize both functions to reflect the appropriate spend rate over H.

  If we assume that the supply pattern remains relatively stable over short periods (i.e., when H is small), then the number of eligible requests can be considered uniformly distributed and proportional to the time duration. In such cases, we can normalize f and g by multiplying them by the ratio $H/\Delta t$, where Δt is the pacing interval. This approach allows us to avoid relying on predicted supply for every pacing interval when updating the bid price.

  However, if H is relatively long, the assumption of a stationary supply distribution no longer holds. In this case, we must rely on the predicted number of requests and compute the normalization factor as the ratio of the number of expected requests in the upcoming H to that of the most recent pacing interval.

3.2 MPC for Target CPA

Target CPA (sometimes referred to as target cost) is another popular automated bidding strategy in which advertisers set a target cost per action (conversion). The goal is to acquire as many conversions as possible at or around the specified CPA.

Compared to the cost cap strategy, which is generally considered more "strict" about not exceeding the cost threshold, target CPA is more flexible in practice. A small deviation from the specified CPA is often tolerable, allowing the algorithm to prioritize achieving a higher number of conversions while maintaining an average cost close to the target CPA. Target CPA is typically a good option for advertisers who have a clear, data-driven understanding of what the ideal CPA should be and can tolerate small deviations above the target.

Problem Formulation The target CPA differs from the cost cap in terms of its cost constraint. More specifically, suppose we allow the average CPA to fluctuate around the target C by a margin of δ (e.g., $\delta = 10\%$). The target CPA problem can then be formulated as:

$$\begin{aligned}
\max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot r_t \\
\text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \le B, \\
(1-\delta) \cdot C \le & \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} \le (1+\delta) \cdot C.
\end{aligned}$$

In practice, instead of imposing two constraints with $(1-\delta)\cdot C$ and $(1+\delta)\cdot C$, we can reformulate the problem to enforce that the posterior CPA is strictly equal to the target C. The problem then becomes:

$$\begin{aligned} \max_{x_t\in\{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot r_t \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \le B, \\ & \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} = C. \end{aligned} \tag{4.3}$$

The objective is now to maximize the total number of conversions while ensuring that the actual CPA remains as close as possible to the target C, subject to the budget constraint B. If the cost constraint is strictly satisfied—meaning that each conversion costs exactly C on average—then maximizing conversions is equivalent to maximizing overall spend delivery.

Computing New Constraints We choose the pacing lifetime of the campaign as the receding horizon. At each pacing interval starting at time τ, we need to determine the updated budget constraint B_τ and cost constraint C_τ.

The budget constraint is straightforward: suppose at time τ, the observed spend is S_τ, then the remaining budget is:

$$B_\tau = B - S_\tau.$$

For the cost constraint, suppose we have collected NC_τ conversions so far. If the average CPA is on target, the expected spend should be $C \cdot NC_\tau$. We define the deviation D_τ at time τ as the difference between this expected spend and the actual spend:

$$D_\tau = S_\tau - C \cdot NC_\tau. \tag{4.4}$$

This represents the **spend deviation** that must be adjusted within the remaining campaign lifetime. Let $S_{\tau,r}$ denote the projected future spend from time τ until the end of the campaign, and let $NC_{\tau,r}$ represent the expected number of conversions within this period. To maintain the target CPA, the remaining ad delivery must compensate for the accumulated spend deviation. This relationship can be expressed as:

$$(S_{\tau,r} - C \cdot NC_{\tau,r}) = -D_\tau.$$

Since:

$$S_{\tau,r} = \sum_{t\ge\tau} x_t \cdot c_t, \quad NC_{\tau,r} = \sum_{t\ge\tau} x_t \cdot r_t,$$

the cost constraint can be rewritten as:

$$\sum_{t\ge\tau} x_t \cdot c_t - C \cdot \left(\sum_{t\ge\tau} x_t \cdot r_t\right) = -D_\tau, \tag{4.5}$$

where D_τ is the accumulated spend deviation at τ as defined in Equation 4.4.

Receding Horizon Optimization Problem Based on the discussion in the previous paragraph, we can now formulate the receding horizon optimization problem at time τ:

$$\begin{aligned}
\max_{x_t \in \{0,1\}} \quad & \sum_{t \geq \tau} x_t \cdot r_t \\
\text{s.t.} \quad & \sum_{t \geq \tau} x_t \cdot c_t \leq B - S_\tau, \\
& \sum_{t \geq \tau} x_t \cdot c_t - C \cdot \sum_{t \geq \tau} x_t \cdot r_t = -D_\tau, \\
& b_l \leq b_t \leq b_u.
\end{aligned}$$

Here, S_τ represents the observed spend at time τ, and D_τ is the accumulated spend deviation, as defined in Equation 4.4, b_l and b_u are lower and upper bounds on the bid to prevent extreme values. As mentioned earlier, as long as the cost constraint is satisfied, maximizing total conversions is equivalent to maximizing total spend.

To capture the balance between spend and conversions, we define the **repay rate function** $\mathcal{R}(b)$ as:

$$\mathcal{R}(b) = f(b) - C \cdot g(b),$$

where:

- $f(b)$ is the **bid-to-spend** function, representing the expected cost as a function of bid price.
- $g(b)$ is the **bid-to-number-of-conversions** function, representing the expected number of conversions as a function of bid price.

Thus, our objective is to determine the optimal bid b^*_τ such that $\mathcal{R}(b^*_\tau)$ offsets the accumulated spend deviation D_τ before the campaign ends while simultaneously maximizing total spend.

Before solving for b^*_τ, we first need to construct the functions $f(b)$ and $g(b)$.

Constructing $f(b)$ and $g(b)$ Both $f(b)$ and $g(b)$ can be constructed similarly to the cost cap problem. We collect the most recent N interval bid-spend and bid-conversion pairs, then apply the LIS or PAVA algorithm along with interpolation methods to obtain a monotonic bid-to-X sequence.

Finding the Optimal Bid Now, we have everything ready to solve for b^*_τ. We define a **projected spend deviation offset function** as:

$$\mathcal{P}(b, \Omega) = \mathcal{R}(b) \cdot \frac{\Omega}{\Delta t} + D_\tau,$$

where:

- Δt is the pacing interval.

- Ω is the evaluation window in which we compute how much spend deviation can be repaid given a fixed bid b. Note that $\Delta t \leq \Omega \leq T_\tau$.
- T_τ is the remaining lifetime of the campaign.

Taking a closer look at $\mathcal{P}(b, \Omega)$, we observe:

- $\mathcal{R}(b)$ represents the spend deviation repaid in **one pacing interval** Δt for a given bid b.
- $\mathcal{R}(b) \cdot \frac{\Omega}{\Delta t}$ estimates the **total repayable spend deviation** over the next Ω time window if we maintain the bid at b.
- $\mathcal{P}(b, \Omega)$ thus represents the **remaining spend deviation** after Ω time, assuming a constant bid b.

Our objective is now clear: We need to iterate over the following values:

$$b \in \{b_l, b_l + \Delta b, b_l + 2 \cdot \Delta b, \ldots, b_u\}, \quad \Omega \in \{\Delta t, 2 \cdot \Delta t, 3 \cdot \Delta t, \ldots, T_\tau\}$$

to find the **maximum** b_τ^* along with some optimal Ω^* such that:

$$\mathcal{P}(b_\tau^*, \Omega^*) = 0.$$

Theoretically, there are three possible scenarios:

- **Existence of b, Ω such that $\mathcal{P}(b, \Omega) = 0$.**
  In this case, among all valid pairs $\{b, \Omega\}$, we select the **highest** b.
- $\mathcal{P}(b, \Omega)$ **is always negative.**
  This implies that the actual CPA remains lower than C regardless of bid level or window selection. To determine the optimal bid:
  1. Find all $\{b, \Omega\}$ where $\mathcal{P}$ is maximized.
  2. Select the **largest** b among those candidates.
- $\mathcal{P}(b, \Omega)$ **is always positive.**
  This implies that the actual CPA is always higher than C. To determine the optimal bid:
  1. Find all $\{b, \Omega\}$ where $\mathcal{P}$ is minimized.
  2. Select the **smallest** b among those candidates, to avoid overspending when conversions are too expensive.

The algorithm we discussed above is summarized in Algorithm 14.

Algorithm 14 MPC-Based Target CPA Bidding Algorithm

```
Require: $B$: Total budget; $C$: Target CPA; $\Delta t$: Pacing interval; $T$: Campaign lifetime;
 1: $[b_l, b_u]$: Bid range; $\Delta b$: Step size; $S_\tau$: Observed spend at $\tau$;
 2: $NC_\tau$: Observed conversions at $\tau$.
Ensure: $b_\tau^*$: Optimal bid for the next pacing interval.
 3: Step 1: Compute Remaining Budget and Spend Deviation
 4: Compute remaining budget:
        $B_\tau \leftarrow B - S_\tau$
 5: Compute accumulated spend deviation:
        $D_\tau \leftarrow S_\tau - C \cdot NC_\tau$
 6: Step 2: Construct Models $f(b)$ and $g(b)$
 7: Collect the most recent $N$ bid-spend pairs $\{b_k, s_k\}$ and apply LIS or PAVA to construct
    $f(b)$ normalized to $\Delta t$.
 8: Collect the most recent $N$ bid-conversion pairs $\{b_k, n_k\}$ and apply LIS or PAVA to con-
    struct $g(b)$ normalized to $\Delta t$.
 9: Define the repay rate function:
        $\mathcal{R}(b) \leftarrow f(b) - C \cdot g(b)$
10: Step 3: Search for Optimal Bid $b_\tau^*$
11: Initialize $b_\tau^* \leftarrow b_l$.
12: for $b$ from $b_l$ to $b_u$ with step size $\Delta b$ do
13:     for $\Omega$ from $\Delta t$ to $T_\tau$ with step size $\Delta t$ do
14:         Compute:
                $\mathcal{P}(b, \Omega) \leftarrow \mathcal{R}(b) \cdot \frac{\Omega}{\Delta t} + D_\tau$
15:         if $\mathcal{P}(b, \Omega) = 0$ then
16:             Update $b_\tau^* \leftarrow \max(b_\tau^*, b)$.
17:         else if $\mathcal{P}(b, \Omega) < 0$ then
18:             Store $(b, \Omega)$ where $\mathcal{P}$ is maximized.
19:         else if $\mathcal{P}(b, \Omega) > 0$ then
20:             Store $(b, \Omega)$ where $\mathcal{P}$ is minimized.
21:         end if
22:     end for
23: end for
24: Step 4: Determine Final Bid
25: if at least one $(b, \Omega)$ satisfies $\mathcal{P}(b, \Omega) = 0$ then
26:     Choose the largest $b$ among candidates.
27: else if $\mathcal{P}(b, \Omega)$ is always negative then
28:     Choose the largest $b$ where $\mathcal{P}$ is maximized.
29: else if $\mathcal{P}(b, \Omega)$ is always positive then
30:     Choose the smallest $b$ where $\mathcal{P}$ is minimized.
31: end if
32: Step 5: Return Optimal Bid
33: return $b_\tau^*$
```

Alternative way to construct $\mathcal{P}(b, \Omega)$ Based on supply instead of duration, TBA.

3.3 Cost Cap vs Target CPA

As we mentioned earlier, for target CPA, if the cost constraint is strictly satisfied—meaning that each conversion costs exactly C on average—then maximizing conversions is equivalent to maximizing overall spend delivery.

This results in a fundamental difference between target CPA and cost cap. In the cost cap strategy:

- If the advertiser sets the cap C too high, the cost constraint becomes ineffective, reducing the problem to a max delivery problem. In this case, the optimal strategy is to allocate the budget across the pacing lifetime based on the supply pattern.
- If C is set too low, the cost constraint becomes the limiting factor, and the actual spend under the optimal strategy will be less than that of the max delivery problem with the same settings (except for the cost constraint). In this case, the delivery is still distributed across the campaign lifetime, though the budget may not be fully depleted by the end of the campaign.

However, target CPA behaves quite differently, especially when the target C is relatively high compared to the market level. Unlike cost cap, which slows pacing and adjusts bids to match market conditions, target CPA dynamically adjusts bids to maintain an average CPA near C.

- If the target C is set at a reasonable level relative to the market, the algorithm dynamically adjusts bids to maximize conversions while maintaining an average CPA close to C. In this case, the pacing behavior is stable, and the budget is allocated efficiently throughout the campaign lifetime.
- If the target C is set too high, the algorithm may initially bid more aggressively to acquire conversions at a higher cost, potentially leading to faster budget depletion.

This is also the reason why we previously mentioned that target CPA is more suitable for advertisers who have a clearer understanding of their ideal CPA target. If the target CPA is set too high, advertisers may end up paying more than the competitive market level to acquire conversions, potentially leading to inefficient spending and faster budget depletion.

We summarize the differences between cost cap and target CPA in Table 4.1.

4 Remarks

4.1 Bidding Stability Considerations

For system stability, when designing a bid update algorithm, it is important to ensure that the overall bidding dynamics remain stable. Drastic fluctuations in bid levels can lead to performance issues. For instance, a sudden drop in bids may result in under-delivery, while a sharp increase in bids could cause rapid budget depletion within a short period.

To address this concern, the algorithms described above can be modified to maintain stability in a production environment. For example, when performing bid searches from b_l to b_u in cost cap and target CPA strategies, we can impose constraints on bid variation to prevent excessive deviation from the previous bid b_t [2]. A possible approach is to set:

$$b_l = (1 - 0.1) \cdot b_t, \quad b_u = (1 + 0.1) \cdot b_t.$$

This principle also applies to the max-delivery problem, where lower and upper bounds can be imposed based on the last bid b_t to prevent excessive fluctuations in the newly updated bid.

4.2 MPC Variants

MPC is a versatile framework, the methods described in this section are not the only way to use MPC algorithm to solve max delivery or cost cap problems.

4.3 Other Online Regression Method

other regression methods, e.g. parametric regression

4.4 Proof of Monotonicity

We prove that $g(\cdot)$, as defined in Equation 4.1, is monotonically non-decreasing. It suffices to show that for any $\tau \geq 0$:

$$g(x+\tau) - g(x) \geq 0.$$

Step 1: Expressing $g(x)$ From the definition of $g(x)$:

$$g(x) = \frac{\int_0^x zp(z)\mathrm{d}z}{\int_0^x p(z)\mathrm{d}z}.$$

Thus, for $g(x+\tau) - g(x)$, we compute:

$$\begin{aligned} g(x+\tau) - g(x) &= \frac{\int_0^{x+\tau} zp(z)\mathrm{d}z}{\int_0^{x+\tau} p(z)\mathrm{d}z} - \frac{\int_0^x zp(z)\mathrm{d}z}{\int_0^x p(z)\mathrm{d}z} \\ &= \frac{\left(\int_0^{x+\tau} zp(z)\mathrm{d}z\right) \cdot \left(\int_0^x p(z)\mathrm{d}z\right) - \left(\int_0^{x+\tau} p(z)\mathrm{d}z\right) \cdot \left(\int_0^x zp(z)\mathrm{d}z\right)}{\left(\int_0^{x+\tau} p(z)\mathrm{d}z\right) \cdot \left(\int_0^x p(z)\mathrm{d}z\right)}. \end{aligned}$$

[2]The choice of this percentile depends on the pacing interval. For example, if the algorithm is expected to adjust bids 10 times per hour, the percentile can be determined based on this number (10) and the number of bid adjustment opportunities available (i.e., 1 hour/pacing interval).

Step 2: Defining the Numerator Define the numerator as $\mathcal{I}(\tau)$:

$$\mathcal{I}(\tau) = \left(\int_0^{x+\tau} zp(z)\mathrm{d}z\right) \cdot \left(\int_0^{x} p(z)\mathrm{d}z\right) - \left(\int_0^{x+\tau} p(z)\mathrm{d}z\right) \cdot \left(\int_0^{x} zp(z)\mathrm{d}z\right).$$

It suffices to show that for all $\tau \geq 0$:

$$\mathcal{I}(\tau) \geq 0.$$

Step 3: Computing the Derivative $\mathcal{I}'(\tau)$ Since $\mathcal{I}(0) = 0$, we show that $\mathcal{I}'(\tau) \geq 0$ whenever $\tau \geq 0$:

$$\begin{aligned}
\frac{\mathrm{d}\mathcal{I}(\tau)}{\mathrm{d}\tau} &= (x+\tau) \cdot p(x+\tau) \cdot \int_0^x p(z)\mathrm{d}z - p(x+\tau) \cdot \int_0^x zp(z)\mathrm{d}z \\
&= p(x+\tau) \cdot \left[(x+\tau) \cdot \int_0^x p(z)\mathrm{d}z - \int_0^x zp(z)\mathrm{d}z\right] \\
&= p(x+\tau) \cdot \left[\int_0^x (x-z) \cdot p(z)\mathrm{d}z + \tau \cdot \int_0^x p(z)\mathrm{d}z\right] \\
&= p(x+\tau) \cdot (\mathcal{I}_1 + \mathcal{I}_2).
\end{aligned}$$

Step 4: Showing $\mathcal{I}'(\tau) \geq 0$ Since $x - z \geq 0$ for $z \in [0, x]$, we obtain:

$$\mathcal{I}_1 = \int_0^x (x-z) \cdot p(z)\mathrm{d}z \geq 0.$$

Moreover, since $\tau \geq 0$:

$$\mathcal{I}_2 = \tau \cdot \int_0^x p(z)\mathrm{d}z \geq 0.$$

Since $p(x+\tau) \geq 0$, it follows that:

$$\frac{\mathrm{d}\mathcal{I}(\tau)}{\mathrm{d}\tau} \geq 0 \quad \text{for } \tau \geq 0.$$

Step 5: Conclusion Since $\mathcal{I}(0) = 0$ and $\mathcal{I}'(\tau) \geq 0$ for all $\tau \geq 0$, we conclude that:

$$\mathcal{I}(\tau) \geq 0.$$

Thus, $g(\cdot)$ is **monotonically non-decreasing**.

	Cost Cap	**Target CPA**
Objective	Maximizes conversions while ensuring the average cost per action (CPA) does not exceed a strict threshold C.	Maximizes conversions while keeping the average CPA close to C, allowing for some fluctuation.
Bid Adjustment Behavior	Adjusts bids dynamically to maintain cost efficiency, **slowing down delivery if necessary** to ensure CPA does not exceed C.	Adjusts bids dynamically to acquire as many conversions as possible **without strictly enforcing an upper CPA limit**. May allow temporary deviations.
C **Too High**	If C is set too high, the cost constraint becomes inactive, and the system behaves like a **max delivery** strategy, distributing spend based on supply.	If C is set too high, the system **may initially bid aggressively** to acquire expensive conversions, potentially leading to **faster budget depletion**.
C **Too Low**	If C is set too low, the system becomes **overly restrictive**, limiting auction participation and leading to **under-delivery** (budget may not be fully spent).	If C is set too low, the system **reduces bid competitiveness**, potentially limiting auction wins but still prioritizing conversion volume within constraints.
Budget Consumption Behavior	Budget pacing is **stable and controlled** to ensure delivery throughout the campaign lifetime.	Budget **may be depleted earlier than scheduled** if the system needs to bid higher to meet the target CPA.

Table 4.1: Comparison of Cost Cap vs. Target CPA

CHAPTER 5

DUAL ONLINE GRADIENT DESCENT

In this chapter, we introduce an adaptive optimal control method called Dual Online Gradient Descent (DOGD). This method updates the bid price by adaptively updating the parameters in the dual space using a stochastic gradient descent algorithm. We demonstrate how this approach can be applied to solve both the max delivery and cost cap problems.

1 Max Delivery

1.1 Main Algorithm

The intuition behind controller-based approaches introduced in previous sections is to tweak the bid of a campaign by comparing the current delivery status to the target delivery schedule. This is based on the fact that the optimal budget consumption rate should be proportional to the distributional density of eligible auction opportunities for the campaign. We can take another perspective by directly solving this problem in the dual space. The key observation is as follows:

Recall that the optimal bid per impression is given by:

$$b_t^* = \frac{r_t}{\lambda^*}.$$

Therefore, to find b_t^*, it is sufficient to determine λ^*. Note that the dual problem is given by Equation 1.2:

$$\min_{\lambda \geq 0} \mathcal{L}^*(\lambda) = \min_{\lambda \geq 0} \sum_{t=1}^{T} \left[(r_t - \lambda c_t)_+ + \lambda \cdot \frac{B}{T} \right].$$

Denoting $(r_t - \lambda c_t)_+ + \lambda \cdot \frac{B}{T}$ by $f_t(\lambda)$, we have:

$$\min_{\lambda \geq 0} \sum_{t=1}^{T} f_t(\lambda).$$

Readers who are familiar with optimization should recognize that this is a standard one-dimensional convex optimization problem. As auction requests arrive online in a streaming manner, it can naturally be solved using the Stochastic Gradient Descent (SGD) method. The update rule for λ is given by:

$$\lambda_t \leftarrow \lambda_t - \epsilon_t \cdot \nabla_\lambda f_t(\lambda) = \lambda_t - \epsilon_t \cdot \left(\frac{B}{T} - \mathbb{1}_{\{r_t > \lambda_t c_t\}} \cdot c_t \right), \tag{5.1}$$

where ϵ_t is the step size(learning rate), $\mathbb{1}_{\{r_t > \lambda_t c_t\}}$ is the indicator function, and c_t is the highest competing bid per impression in the market. Note that $\mathbb{1}_{\{r_t > \lambda_t c_t\}} \cdot c_t$ represents the observed spend in the t-th auction, while B/T is the expected spend per auction. The gradient thus quantifies the deviation from the expected spend. The bid at the t-th auction round is:

$$b_t = \frac{r_t}{\lambda_t}.$$

The discussion above highlights the core idea of the **D**ual **O**nline **G**radient **D**escent (**DOGD**) algorithm for the max delivery problem, which we summarize in Algorithm 15:

Algorithm 15 DOGD for Max Delivery

```
Require: $B$: Total budget, $T$: Predicted total number of auction opportunities, $\epsilon_t$: Step
    size schedule
1: Initialize $\lambda_0 \leftarrow \lambda_{\text{init}}$                              ▷ Initial dual variable
2: for all incoming auction requests indexed by $t$ do         ▷ Iterate over auction rounds
3:     Observe $r_t$: pCTR, $c_t$: highest competing eCPM in auction $t$
4:     Compute the gradient of $f_t(\lambda)$:
           $$\nabla_\lambda f_t(\lambda) = \frac{B}{T} - \mathbb{1}_{\{r_t > \lambda_t c_t\}} \cdot c_t$$
5:     Update $\lambda_t$ using SGD:
           $$\lambda_t \leftarrow \lambda_t - \epsilon_t \cdot \nabla_\lambda f_t(\lambda)$$
6:     Compute the bid per impression for the $t$-th auction:
           $$b_t \leftarrow \frac{r_t}{\lambda_t}$$
7:     Submit $b_t$ for auction $t$
8: end for
```

As we discussed before, instead of updating bids for every auction, it is more common to update bids in a batch manner. Suppose the update interval is Δt and $R(t)$ is the number of observed auction requests within this interval. The mini-batch gradient within Δt is given by:

$$\sum_{s\in(t,t+\Delta t)} \nabla_\lambda f_s(\lambda) = \sum_{s\in(t,t+\Delta t)} \left(\frac{B}{T} - \mathbb{1}_{\{r_s > \lambda_s c_s\}} \cdot c_s\right) = \frac{R(t)}{T} \cdot B - \sum_{s\in(t,t+\Delta t)} \mathbb{1}_{\{r_s > \lambda_s c_s\}} \cdot c_s.$$

Note that $\sum_{s\in(t,t+\Delta t)} \mathbb{1}_{\{r_s > \lambda_s c_s\}} \cdot c_s$ represents the actual spend during Δt, which we denote as $S(t)$. The mini-batch gradient can then be written as:

$$\sum_{s\in(t,t+\Delta t)} \nabla_\lambda f_s(\lambda) = \frac{R(t)}{T} \cdot B - S(t).$$

The mini-batch update rule is given by:

$$\lambda_t \leftarrow \lambda_t - \epsilon_t \cdot \left(\frac{R(t)}{T} \cdot B - S(t)\right).$$

The bid per click remains unchanged within $(t, t+\Delta t)$ and is given by:

$$b_{\text{click},t} = \frac{1}{\lambda_t}$$

For each auction request $s \in (t, t+\Delta t)$, the bid per impression is computed as:

$$b_s = b_{\text{click},t} \cdot r_s = \frac{r_s}{\lambda_t}$$

We summarize this Mini-Batch DOGD algorithm in Algorithm 16:

Algorithm 16 Mini-Batch DOGD for Max Delivery Problem

```
Require: $B$: Total budget, $T$: Predicted total number of auction opportunities, $\Delta t$: Mini-
    batch update interval, $\epsilon_t$: Step size schedule
Ensure: Optimal dual variable $\lambda_t$ and corresponding bids per impression $b_s$
1: Initialize $\lambda_0 \leftarrow \lambda_{\text{init}}$                          ▷ Initial dual variable
2: for $t = 0$ to EndOfDay with step size $\Delta t$ do                ▷ Iterate over mini-batches
3:     Count the number of auction requests $R(t)$ and observe the actual spend $S(t)$ during
       interval $(t, t + \Delta t)$
4:     Compute the mini-batch gradient:
           $$\text{BatchGrad}_t = \sum_{s \in (t, t+\Delta t)} \nabla_\lambda f_s(\lambda) = \frac{R(t)}{T} \cdot B - S(t)$$
5:     Update the dual variable using the mini-batch gradient:
           $$\lambda_t \leftarrow \lambda_t - \epsilon_t \cdot \text{BatchGrad}_t$$
6:     Compute the bid per click for all auctions in $(t, t + \Delta t)$:
           $$b_{click,t} = \frac{1}{\lambda_t}$$
7:     Compute bid per impression $b_s$ for all $s \in (t, t + \Delta t)$ with pCTR $r_s$:
           $$b_s = b_{click,t} \cdot r_s$$
8: end for
```

Some remarks on the DOGD algorithm: The optimization is performed at the campaign level. Interestingly, under certain regularity conditions, this campaign-level optimization leads to a marketplace Nash equilibrium. Furthermore, regret analysis can be conducted to demonstrate that this algorithm is theoretically optimal. For more technical details, one may refer to [8], [6], and [32].

1.2 Practical Considerations

Someome practical considerations for implementing the DOGD algorithm for max delivery in real-world production sytem:

- λ initialization
  More details will be discussed in next part chapter 1
- Normalization
  - Normalization of update rule

  - Normalization of λ

- Target Ratio Choice
- Mirror Gradient Descent relation between PID and mirror gradient descent: [9]
- Sparsity Problem

2 Cost Cap

2.1 Main Algorithm

The Cost Cap problem can also be solved using the DOGD algorithm. Recall that the optimal bid per click for the cost cap is given by Equation 1.6:

$$b^*_{\text{click}} = \frac{\lambda^*}{\lambda^* + \mu^*} \cdot \frac{1}{\lambda^*} + \frac{\mu^*}{\lambda^* + \mu^*} \cdot C,$$

where λ^* and μ^* are the dual variables.

Similar to the method discussed in the Max Delivery problem in the previous section, solving this problem reduces to determining the dual variables λ^* and μ^*. The dual problem of the cost cap is given by Equation 1.5:

$$\min_{\lambda \geq 0, \mu \geq 0} \mathcal{L}^*(\lambda, \mu) = \min_{\lambda \geq 0, \mu \geq 0} \sum_{t=1}^{T} \left[\left(r_t - \lambda c_t - \mu c_t + \mu C r_t \right)_+ + \lambda \cdot \frac{B}{T} \right].$$

Define the per-time step loss function as:

$$f_t(\lambda, \mu) = \left(r_t - \lambda c_t - \mu c_t + \mu C r_t \right)_+ + \lambda \cdot \frac{B}{T}.$$

The dual problem can then be rewritten as:

$$\min_{\lambda \geq 0, \mu \geq 0} \mathcal{L}^*(\lambda, \mu) = \min_{\lambda \geq 0, \mu \geq 0} \sum_{t=1}^{T} f_t(\lambda, \mu).$$

Using stochastic gradient descent (SGD), the update rules for λ and μ are:

$$\lambda_{t+1} \leftarrow \lambda_t - \epsilon_t \cdot \nabla_\lambda f_t(\lambda, \mu) = \lambda_t - \epsilon_t \cdot \left(\frac{B}{T} - c_t \cdot \mathbb{1}_{\{r_t - \lambda c_t - \mu c_t + \mu C r_t > 0\}} \right),$$
$$\mu_{t+1} \leftarrow \mu_t - \epsilon_t \cdot \nabla_\mu f_t(\lambda, \mu) = \mu_t - \epsilon_t \cdot \left(C r_t - c_t \right) \cdot \mathbb{1}_{\{r_t - \lambda c_t - \mu c_t + \mu C r_t > 0\}},$$

where ϵ_t is the learning rate, and $\mathbb{1}_{\{\cdot\}}$ is the indicator function. If we examine the update rules more closely, c_t / r_t represents the actual cost per click in the sense of expectation. From this, we can observe that the gradients of λ and μ quantify the deviations from the target spend and the target cost per click (CPC), respectively.

Algorithm 17 Dual Online Gradient Descent (DOGD) for Cost Cap

```
Require: $B$: Total budget, $C$: Target cost per click (CPC), $T$: Predicted total number of
    auction opportunities, $\epsilon_t$: Step size schedule
Ensure: Bid values for auctions
1: Initialize $\lambda_0 \leftarrow \lambda_{\text{init}}, \mu_0 \leftarrow \mu_{\text{init}}$          ▷ Initial dual variables
2: for all incoming auction requests indexed by $t$ do
3:     Observe $r_t$: predicted click-through rate (pCTR), $c_t$: highest competing eCPM in
       auction $t$
4:     Compute the gradient of $f_t(\lambda, \mu)$:
           $\nabla_\lambda f_t(\lambda, \mu) \leftarrow \frac{B}{T} - c_t \cdot \mathbb{1}_{\{r_t - \lambda c_t - \mu c_t + \mu C r_t > 0\}}$
           $\nabla_\mu f_t(\lambda, \mu) \leftarrow \big(C r_t - c_t\big) \cdot \mathbb{1}_{\{r_t - \lambda c_t - \mu c_t + \mu C r_t > 0\}}$
5:     Update $\lambda_t$ and $\mu_t$ using SGD:
           $\lambda_{t+1} \leftarrow \lambda_t - \epsilon_t \cdot \nabla_\lambda f_t(\lambda, \mu)$
           $\mu_{t+1} \leftarrow \mu_t - \epsilon_t \cdot \nabla_\mu f_t(\lambda, \mu)$
6:     Compute the bid per impression for the $t$-th auction:
           $b_t \leftarrow \frac{1 + \mu_t \cdot C}{\lambda_t + \mu_t} \cdot r_t$
7:     Submit $b_t$ for auction $t$
8: end for
```

As we discussed in the previous section, in practice, it is more common to implement the batch update algorithm. The batch update of λ is quite similar to the formula used in max delivery. The mini-batch gradient is given by:

$$\sum_{s \in (t, t+\Delta t)} \nabla_\lambda f_s(\lambda, \mu) = \frac{R(t)}{T} \cdot B - S(t),$$

where Δt is the update time interval, $R(t)$ is the number of observed auction requests, and $S(t)$ is the actual spend during Δt. As for μ, note that:

$$\sum_{s \in (t, t+\Delta t)} r_s \mathbb{1}_{\{r_s - \lambda c_s - \mu c_s + \mu C r_s > 0\}}$$

is the expected number of conversions (in this case, clicks) during Δt. If, within Δt, there are sufficient actual conversions (denoted as $N(t)$), then $N(t)$ is a good approximation of the sum above. Thus:

$$\sum_{s \in (t, t+\Delta t)} \nabla_\mu f_s(\lambda, \mu) \approx C \cdot N(t) - S(t).$$

The mini-batch update rule is then:

$$\lambda_{t+1} \leftarrow \lambda_t - \epsilon_t \cdot \left(\frac{R(t)}{T} \cdot B - S(t)\right),$$
$$\mu_{t+1} \leftarrow \mu_t - \epsilon_t \cdot (C \cdot N(t) - S(t)),$$

where ϵ_t is the step size.

Similar to max delivery, in the mini-batch update, the bid per click remains unchanged within $(t, t + \Delta t)$ and is given by:

$$b_{\text{click},t} = \frac{1 + \mu_t C}{\lambda_t + \mu_t}.$$

The bid per impression for any $s \in (t, t + \Delta t)$ is:

$$b_s = b_{\text{click},t} \cdot r_s = \frac{1 + \mu_t C}{\lambda_t + \mu_t} \cdot r_s.$$

Algorithm 18 Mini-Batch DOGD for Cost Cap Problem

```
Require: $B$: Total budget, $C$: Target cost per click (CPC), $T$: Predicted total number of
    auction opportunities, $\Delta t$: Mini-batch update interval, $\epsilon_t$: Step size schedule
Ensure: Optimal dual variables $\lambda_t$, $\mu_t$, and corresponding bids per impression $b_s$
1: Initialize $\lambda_0 \leftarrow \lambda_{\text{init}}, \mu_0 \leftarrow \mu_{\text{init}}$        ▷ Initial dual variables
2: for $t = 0$ to EndOfDay with step size $\Delta t$ do        ▷ Iterate over mini-batches
3:     Count the number of auction requests $R(t)$ and observe the actual spend $S(t)$ during
       interval $(t, t + \Delta t)$
4:     Count the number of conversions (clicks) $N(t)$ during interval $(t, t + \Delta t)$
5:     Compute the mini-batch gradients:
           $\text{BatchGrad}_{\lambda,t} = \frac{R(t)}{T} \cdot B - S(t)$
           $\text{BatchGrad}_{\mu,t} = C \cdot N(t) - S(t)$
6:     Update the dual variables using the mini-batch gradients:
           $\lambda_{t+1} \leftarrow \lambda_t - \epsilon_t \cdot \text{BatchGrad}_{\lambda,t}$
           $\mu_{t+1} \leftarrow \mu_t - \epsilon_t \cdot \text{BatchGrad}_{\mu,t}$
7:     Compute the bid per click for all auctions in $(t, t + \Delta t)$:
           $b_{\text{click},t} = \frac{1 + \mu_t C}{\lambda_t + \mu_t}$
8:     Compute bid per impression $b_s$ for all $s \in (t, t + \Delta t)$ with pCTR $r_s$:
           $b_s = b_{\text{click},t} \cdot r_s = \frac{1 + \mu_t C}{\lambda_t + \mu_t} \cdot r_s$
9: end for
```

Related algorithms can be found in [32].

2.2 Practical Considerations

Some practical considerations for implementing the DOGD algorithm for cost cap in real-world production system are listed as follows:

- Initialization of λ and μ
- Normalization
  - Normalization of the update rule
  - Normalization of λ and μ
  - Sparsity Problem

3 Remarks

3.1 Other Applications of DOGD

Versatile framework, will demonstrate in the following chapters for multi-channel and multi-campaign optimizations.

3.2 DOGD and PID

Connection between DOGD and PID for max delivery, mirror descent

3.3 Nash Equilibrium

Part III

Applications

CHAPTER 1

INITIALIZATION OF CAMPAIGN BID

The initial bid of a campaign is crucial in the pacing problem, as it influences how quickly the bidding algorithm converges to the optimal level. In this chapter, we present two approaches—parametric and non-parametric methods—to demonstrate how algorithms can be designed to derive reasonable initial bids, thereby accelerating the convergence of the bidding algorithm.

Most of the algorithms discussed in Part II focus on how to update bids for pacing during a campaign's delivery. However, setting the initial bid properly is equally important for both individual campaign performance and the overall stability of the marketplace.

A poorly chosen initial bid can significantly impact the performance of the campaign:

- If the initial bid is **too low**, the campaign might take longer to adjust its bid to a reasonable market level, delaying its ability to win auction opportunities and meet delivery objectives.
- If the initial bid is **too high**, the campaign might win auction opportunities at unreasonably high prices, leading to rapid budget depletion without maximizing conversions or other objectives.

From a macroscopic perspective, poorly set initial bids across campaigns can influence the stability of the entire marketplace. If many campaigns start with bids significantly deviating from the equilibrium market level, it can cause fluctuations in auction dynamics, leading to inefficient resource allocation.

In this chapter, we discuss how to compute a **reasonable initial bid** for a given ad campaign, balancing the need to quickly achieve competitive performance while avoiding excessive costs or instability.

1 Parametric Approach

In this section, we discuss the computation of the optimal initial bid using parametric methods. The algorithms described below rely on certain model predictions as input parameters. We do not delve too deeply into the underlying modeling techniques; instead, we primarily focus on the methodologies for determining the initial bids, assuming that all relevant signals are ready for use.

1.1 Max Delivery

We first discuss how to compute the initial bid for a cold start max delivery campaign. As shown in Part II, the optimal bid for a max delivery campaign is a constant that exactly depletes the campaign's budget. For a cold start campaign, where no prior data is available, we need to leverage models that estimate market conditions and campaign-specific parameters to make the best guess for the initial bid.

More specifically, suppose the highest eCPMs for this campaign among other bidders follow a log-normal distribution with parameters μ and σ. Additionally, assume the conversion rate[1] of the ad (independently) follows a log-normal distribution with parameters μ' and σ'.[2] Given

[1] Conversion rate refers to the probability of achieving the campaign's objective conditioned on an impression. For example, for CPC campaigns, it corresponds to CTR, while for CPA campaigns, it corresponds to PVR, etc.

[2] Both eCPM and conversion rate are non-negative. It is common in practice to assume these data are subject to log-normal distributions. Empirical evidence suggests that the log-normal distribution is a good candidate for fitting these data.

a budget B and a total opportunity forecast T, under a second price auction, we claim that the optimal bid per conversion b^* can be determined by solving the following equation:

$$e^{\mu+\frac{\sigma^2}{2}}\Phi\left(\frac{\mu'-\mu+\ln b^*-\sigma^2}{\sqrt{(\sigma')^2+\sigma^2}}\right)=\frac{B}{T}, \tag{1.1}$$

where Φ denotes the cumulative distribution function (CDF) of a standard Gaussian distribution $\mathcal{N}(0,1)$.

We now break down Equation 1.1 to interpret the meaning of each term:

- $\exp(\mu+\sigma^2/2)$: Suppose the clearing price(the highest eCPM among other bidders) Z follows a log-normal distribution:

  $$Z \sim \mathcal{LN}(\mu,\sigma^2).$$

  Then, the expected clearing price per impression is given by:

  $$\mathbb{E}[Z]=\exp\left(\mu+\frac{\sigma^2}{2}\right).$$

  Thus, the term $\exp(\mu+\sigma^2/2)$ represents the expected clearing price per impression.

- $\Phi\left(\frac{\mu'-\mu+\ln b^*-\sigma^2}{\sqrt{(\sigma')^2+\sigma^2}}\right)$: Suppose the conversion rate is represented by a log-normal random variable:

  $$R \sim \mathcal{LN}(\mu',(\sigma')^2).$$

  Given a bid b^*, the auction is won if and only if:

  $$b^*\cdot R > Z.$$

  The probability of winning an auction, conditioned on R, Z, and b^*, is given by:

  $$P(b^*\cdot R > Z).$$

  Since R and Z are independent, we can prove:

  $$P(b^*\cdot R > Z)=\Phi\left(\frac{\mu'-\mu+\ln b^*-\sigma^2}{\sqrt{(\sigma')^2+\sigma^2}}\right).$$

  Therefore, the Φ term represents the probability that the advertiser's eCPM bid, given b^*, exceeds the highest competing eCPM bid.

- B/T: This term represents the target cost per impression for the campaign.

In summary, Equation 1.1 implies that b^* is the bid level at which the expected cost per impression matches the target cost per impression, which is consistent with our previous derivations. We will prove this result in section 3 of Remarks. For now, we assume this is correct. The problem is then equivalent to finding $\mu, \sigma, \mu', \sigma'$, and T. Below, we describe methods for estimating these parameters:

- μ, σ: These represent the distribution of eCPMs for this campaign. There are two main approaches to estimate these parameters:
    1. **Sampling-Based Approach:** Sample auctions from the target audience and record the highest eCPMs observed for each auction. Fit these observed eCPMs to a log-normal distribution using methods such as maximum likelihood estimation (MLE) to obtain μ and σ.
    2. **Historical Data Aggregation:** If historical data is available for similar campaigns with comparable targeting criteria, aggregate the eCPMs and fit a log-normal distribution to derive μ and σ.
- μ', σ': These parameters describe the distribution of conversion rates, reflecting the quality of the ad campaign. Possible approaches include:
    1. **Regression-Based Prediction:** Leverage features such as the campaign's ad creatives, targeting criteria, and historical performance data to train a regression model (e.g., linear regression, gradient boosting, or neural networks). Use this model to predict μ' and σ'.
    2. **Lookalike Campaigns:** Analyze the conversion rates of similar past campaigns to estimate μ' and σ'. Use transfer learning techniques if datasets from lookalike campaigns are small.
- T: This represents the predicted number of auction requests and we may agregate historical supply data for the given targeting criteria of the campaign to forecast the total opportunity.

Once these parameters are determined, we can compute b^* directly by applying the inverse cumulative distribution function Φ^{-1} of the standard Gaussian distribution to solve Equation 1.1. The algorithm is summarized in Algorithm 19.

1.2 Cost Cap

Deciding the optimal initial bid for a cost cap campaign is more complex than for a max delivery campaign. Recall that the objective of cost cap bidding is to maximize conversions while ensuring that the cost per result does not exceed a specified cap C set by the advertiser. The initial bid computed in Algorithm 19 only considers the budget constraint. However, the posterior cost per result when bidding with this initial value may exceed the cap threshold. To address this issue, a natural approach is to cap the initial bid in Algorithm 19 by the specified cost cap. Formally, we define the initial bid for a cost cap campaign as:

$$\min\left(b^*, C\right).$$

If we assume that the ratio σ between the highest eCPM and the second-highest eCPM remains stable over time, we can further refine the formula as:

$$\min\left(b^*, \frac{C}{\sigma}\right).$$

Algorithm 19 Optimal Initial Bid For Max Delivery Problem

```
Require: $B$: Campaign budget; $T$: Predicted total opportunity forecast;
 1: $\mu, \sigma$: Parameters of eCPM distribution (log-normal);
 2: $\mu', \sigma'$: Parameters of campaign-specific conversion rate distribution (log-normal).
Ensure: $b^*$: Optimal initial bid per conversion.
 3: Step 1: Compute Auxiliary Values
 4: Compute the expected spend per impression:
        $$\text{Spend} \leftarrow e^{\mu + \frac{\sigma^2}{2}}$$
 5: Compute the standard deviation for combined log-normal distributions:
        $$\text{SD} \leftarrow \sqrt{(\sigma')^2 + \sigma^2}$$
 6: Compute the RHS of the CDF equation:
        $$\text{RHS} \leftarrow \frac{B}{T \cdot \text{Spend}}$$
 7: Step 2: Solve for Optimal Bid
 8: Compute the inverse CDF value:
        $$z \leftarrow \Phi^{-1}(\text{RHS})$$
 9: Compute the logarithm of the optimal bid:
        $$\ln b^* \leftarrow \mu - \mu' + \sigma^2 + \text{SD} \cdot z$$
10: Compute the optimal bid:
        $$b^* \leftarrow \exp(\ln b^*)$$
11: Step 3: Return Optimal Bid return $b^*$
```

For additional details, readers may refer to the methodologies described in the "Cost-Min" algorithm.

2 Non-Parametric Approach

For campaigns with historical auction data (e.g., daily pacing campaigns that reset the initial bid at the beginning of each new pacing cycle), non-parametric methods can be employed to derive the initial bid directly from past auction data.

Unlike parametric methods, which rely heavily on the accuracy of prediction models, non-parametric approaches are less dependent on model quality. When sufficient auction data are available, non-parametric methods provide a robust alternative for determining the initial bid.

2.1 Auction Replay

Suppose we want to compute the initial bid for a daily pacing campaign and have access to its historical auction data from the past N days. If we are confident that the auction data from a particular day reflects a similar auction environment—such as having the same budget and a comparable supply or traffic pattern—with the upcoming pacing day,[3] then this historical auction data can be leveraged to run simulations(auction replay) and estimate the optimal initial bid.

More specifically, assume that the historical data from the selected day includes all the auctions in which the campaign has participated. For each auction opportunity i, we have access to the highest eCPM c_i and the predicted click-through rate (pCTR) r_i for the campaign in the given auction slot. Suppose there are T auction opportunities. Given the set of pairs $\{c_i, r_i\}_{i=1}^{T}$, our objective is to determine a constant bid that ensures the daily budget B is fully depleted by the end of the day.

The simulation can be executed as follows: for a given bid price b, we iterate over all auction opportunities. For each auction opportunity i, we compute the eCPM as:

$$eCPM_i = b \cdot r_i.$$

This computed $eCPM_i$ is then compared to the highest competing eCPM c_i. If $eCPM_i > c_i$, we win the auction and pay the second-highest price c_i; otherwise, we lose the auction and incur no cost.

The total cost is accumulated across all T auctions and compared to the daily budget B. If the total spend is less than B, the campaign is under-delivered, indicating that the bid needs to be increased. Conversely, if the spend exceeds B, the bid should be decreased.

This process is repeated iteratively until we determine the optimal bid b^*, where the budget is fully spent by the end of the day. Since a higher bid generally leads to higher spending, a **binary search** approach can be employed to accelerate the search for b^*. We predefine a search interval $[b_l, b_u]$ and apply binary search to efficiently converge on the optimal bid. We summarize this idea in Algorithm 20.

[3] In practice, the traffic pattern might be similar to the previous day or the same day from the prior week.

Algorithm 20 Optimal Initial Bid via Auction Replay

```
Require: Historical auction data $\{(c_i, r_i)\}_{i=1}^T$, budget $B$, search interval $[b_l, b_u]$, convergence
    threshold $\epsilon$
Ensure: Optimal initial bid $b^*$
 1: while $b_u - b_l > \epsilon$ do
 2:     $b \leftarrow \frac{b_l+b_u}{2}$                                  ▷ Midpoint of search interval
 3:     total_cost $\leftarrow 0$
 4:     for $i = 1$ to $T$ do
 5:         $eCPM_i \leftarrow b \cdot r_i$
 6:         if $eCPM_i > c_i$ then
 7:             total_cost $\leftarrow$ total_cost $+ c_i$
 8:         end if
 9:     end for
10:     if total_cost $< B$ then
11:         $b_l \leftarrow b$                                         ▷ Increase bid to spend more
12:     else
13:         $b_u \leftarrow b$                                         ▷ Decrease bid to spend less
14:     end if
15: end while
16: return $b^* \leftarrow \frac{b_l+b_u}{2}$                          ▷ Final bid approximation
```

2.2 Converged Bids Average

The simulation method discussed above relies on auction data, including second-highest prices and predicted click-through rates (pCTRs). An alternative approach to estimating the initial bid requires only the historical bids from the campaign itself.

The key assumption behind this method is that the bidding algorithm dynamically adjusts bids online to reach an optimal level. If we observe that the bidding dynamics have converged over time, the average of these converged bids provides a good approximation of the optimal bid.

Suppose we have a total of T bid data points, denoted as $\{b_i\}_{i=1}^T$, collected throughout the day. To determine whether a consecutive subsequence of bids has converged, a simple approach is to compute the variance of the subsequence. If the variance is below a predefined threshold δ, we consider the sequence to be convergent.

To further reduce variance fluctuations, we impose a constraint that the subsequence length must be at least a predefined value, say K. The objective, therefore, is to find the longest consecutive subsequence with a minimum length of K and variance less than δ.

A brute-force approach would require $O(T^3)$ time complexity in the worst-case scenario—since there are $O(T^2)$ possible consecutive subsequences, and computing the variance for each subsequence takes $O(m)$ time (where m is the length of the subsequence). This becomes computationally expensive when the bid sequence is long; for instance, if the pacing interval is 30 seconds, there could be up to 2,880 bid data points per day.

A more efficient approach utilizes a sliding window combined with two pointers and prefix sums:

- Prefix sums allow us to compute the mean and variance of a subsequence efficiently.
- Sliding window and two-pointer techniques help search for the longest valid subsequence while maintaining the variance constraint.

This optimized method significantly improves computational efficiency to $O(T)$ compared to the brute-force approach, making it more efficient for large-scale bidding data computation. We summarize this approach in the following Algorithm 21:

Algorithm 21 Optimal Initial Bid via Converged Bids Average

```
Require: Sequence $\{b_i\}_{i=1}^T$, variance threshold $\delta$, minimum length $K$
Ensure: Longest consecutive subsequence with variance $\leq \delta$ and length $\geq K$, and its average bid
 1: Compute prefix sums:
        $$S(i) = \sum_{j=1}^{i} b_j, \quad Q(i) = \sum_{j=1}^{i} b_j^2$$
 2: Initialize $L \leftarrow 1$, $max\_length \leftarrow 0$, $best\_interval \leftarrow (0, 0)$
 3: for $R = 1$ to $T$ do
 4:     while $R - L + 1 \geq K$ do
 5:         Compute mean:
                $$\mu = \frac{S(R) - S(L-1)}{R - L + 1}$$
 6:         Compute variance:
                $$\sigma^2 = \frac{Q(R) - Q(L-1)}{R - L + 1} - \mu^2$$
 7:         if $\sigma^2 \leq \delta$ then
 8:             if $R - L + 1 > max\_length$ then
 9:                 $max\_length \leftarrow R - L + 1$
10:                 $best\_interval \leftarrow (L, R)$
11:                 $best\_average \leftarrow \mu$          ▷ Store the best average bid
12:             end if
13:             break                                        ▷ Expand the window further
14:         else
15:             $L \leftarrow L + 1$                         ▷ Shrink the window
16:         end if
17:     end while
18: end for
19: return $best\_interval, best\_average$
```

In practice, bid scales can vary significantly across different campaigns. To enhance the robustness of the algorithm, we can rescale the bid data points $\{b_i\}_{i=1}^T$ using a reference value, such as the initial bid b_1 or the average bid over the dataset. By normalizing bids across different campaigns to a comparable scale, we reduce variability and improve the effectiveness of the algorithm discussed above. This normalization ensures that the variance-based convergence detection remains consistent across campaigns with different bidding magnitudes, making the approach more robust and generalizable.

3 Remarks

3.1 Proof of Equation 1.1

We assume:

- The clearing price Z follows a lognormal distribution Lognormal(μ, σ^2) with PDF

$$p_{\mu,\sigma}(z) \;=\; \frac{1}{z\,\sigma\sqrt{2\pi}}\exp\Bigl(-\tfrac{(\ln z-\mu)^2}{2\,\sigma^2}\Bigr), \quad z>0.$$

- A random factor R follows a lognormal distribution Lognormal(μ', σ'^2) and scales our bid so that the *actual* bid is $b\,R$.
- We pay the second price, i.e. we pay Z if $Z < b\,R$, and we pay 0 otherwise.

Define $c(b)$ to be the expected cost per impression when bidding b (per unit of R). Formally,

$$c(b) \;=\; \iint_{\{\,z<b\,r\,\}} z\,p_{\mu,\sigma}(z)\,p_{\mu',\sigma'}(r)\,dz\,dr \;=\; \int_0^\infty\Bigl(\int_0^{b\,r} z\,p_{\mu,\sigma}(z)\,dz\Bigr)p_{\mu',\sigma'}(r)\,dr.$$

Step 1. Evaluate the inner integral. Since $Z \sim$ Lognormal(μ, σ^2), one has the standard identity:

$$\int_0^x z\,p_{\mu,\sigma}(z)\,dz \;=\; \exp\Bigl(\mu+\tfrac{1}{2}\,\sigma^2\Bigr)\,\Phi\Bigl(\tfrac{\ln x-\mu-\sigma^2}{\sigma}\Bigr),$$

where $\Phi(\cdot)$ is the CDF of the standard normal distribution. By setting $x = b\,r$, it follows that

$$\int_0^{b\,r} z\,p_{\mu,\sigma}(z)\,dz \;=\; \exp\bigl(\mu+\tfrac{1}{2}\,\sigma^2\bigr)\,\Phi\Bigl(\tfrac{\ln(b\,r)-\mu-\sigma^2}{\sigma}\Bigr).$$

Thus

$$c(b) \;=\; \exp\bigl(\mu+\tfrac{1}{2}\,\sigma^2\bigr)\;\int_0^\infty \Phi\Bigl(\tfrac{\ln(b\,r)-\mu-\sigma^2}{\sigma}\Bigr)\,p_{\mu',\sigma'}(r)\,dr.$$

Step 2. Substitute for the outer integral (the lognormal R). Since $R \sim$ Lognormal(μ', σ'^2),

$$p_{\mu',\sigma'}(r)\,dr \;=\; \frac{1}{r\,\sigma'\sqrt{2\pi}}\exp\Bigl(-\tfrac{(\ln r-\mu')^2}{2\,\sigma'^2}\Bigr)\,dr.$$

Make the change of variable $x := \ln r$. Then $r = e^x$, $dr = e^x\,dx$, and

$$p_{\mu',\sigma'}(r)\,dr \;=\; \frac{1}{\sigma'\sqrt{2\pi}}\exp\Bigl(-\tfrac{(x-\mu')^2}{2\,\sigma'^2}\Bigr)\,dx.$$

Hence

$$c(b) \;=\; \exp\bigl(\mu + \tfrac{1}{2}\,\sigma^2\bigr) \int_{-\infty}^{\infty} \Phi\Bigl(\tfrac{\ln b + x - \mu - \sigma^2}{\sigma}\Bigr)\, \frac{1}{\sigma'\sqrt{2\pi}} \exp\Bigl(-\tfrac{(x-\mu')^2}{2\,\sigma'^2}\Bigr)\, dx.$$

Step 3. Apply a bivariate-normal identity. Set $x = \sigma'\,y + \mu'$, i.e. $y = \frac{x-\mu'}{\sigma'}$. Then $dx = \sigma'\,dy$, and

$$\ln b + x - \mu - \sigma^2 \;=\; \bigl(\ln b \;+\; \mu' \;-\; \mu \;-\; \sigma^2\bigr) \;+\; \sigma'\,y.$$

Inside the integral,

$$\Phi\Bigl(\tfrac{\ln b + x - \mu - \sigma^2}{\sigma}\Bigr) \quad \text{and} \quad \exp\Bigl(-\tfrac{(x-\mu')^2}{2\,\sigma'^2}\Bigr)\, \frac{dx}{\sigma'\sqrt{2\pi}}$$

become a classic form:

$$\int_{-\infty}^{\infty} \Phi(\alpha + \beta\,y)\, \frac{1}{\sqrt{2\pi}}\, e^{-\frac{y^2}{2}}\, dy \;=\; \Phi\Bigl(\tfrac{\alpha}{\sqrt{1+\beta^2}}\Bigr),$$

with $\alpha = \frac{\ln b + \mu' - \mu - \sigma^2}{\sigma}$ and $\beta = \frac{\sigma'}{\sigma}$. This well-known identity yields

$$c(b) \;=\; \exp\bigl(\mu + \tfrac{1}{2}\,\sigma^2\bigr)\;\; \Phi\Bigl(\tfrac{\ln b + \mu' - \mu - \sigma^2}{\sqrt{\sigma^2 + \sigma'^2}}\Bigr).$$

By slightly rearranging parameters, one typically writes

$$c(b) \;=\; \exp\Bigl(\mu + \tfrac{1}{2}\,\sigma^2\Bigr)\, \Phi\Bigl(\tfrac{\mu' - \mu + \ln b - \sigma^2}{\sqrt{\sigma'^2 + \sigma^2}}\Bigr),$$

which is the standard *double-lognormal* closed-form expression.

Budget constraint and conclusion. If the total budget is B for T impressions, *exhausting* that budget exactly means

$$c(b^*)\,T \;=\; B \quad\Longrightarrow\quad c(b^*) \;=\; \frac{B}{T}.$$

Thus we set

$$\exp\Bigl(\mu + \tfrac{1}{2}\,\sigma^2\Bigr)\, \Phi\Bigl(\tfrac{\mu' - \mu + \ln(b^*) - \sigma^2}{\sqrt{\sigma'^2 + \sigma^2}}\Bigr) \;=\; \frac{B}{T},$$

which is exactly Equation (1.1). Hence there is a unique b^* solving $c(b^*) = B/T$, and the proof is complete.

3.2 Bid Cap Suggestion Problem

When an advertiser creates a cost cap campaign in an Ad Manager[4], the platform typically provides a suggested bid for the cap threshold to prevent misconfiguration by the advertiser.

This suggested bid plays a critical role in the ad platform's business. If the suggested bid is too low, the cost cap campaign may suffer from low delivery, resulting in revenue loss for

[4]An Ad Manager is a tool or platform that enables businesses, advertisers, and marketers to create, manage, and optimize their advertising campaigns across various digital channels. Examples include Facebook Ads Manager, Google Ads, LinkedIn Campaign Manager, and Amazon DSP.

the platform. Conversely, if the suggested bid is too high, it may deter advertisers, leading to a lower adoption rate.

From the platform's perspective, the optimal strategy is to suggest a bid that allows the campaign budget to be fully spent by the end of its lifetime. Consequently, the bid cap suggestion problem can be formulated as an instance of the initial bid optimization problem for max delivery with the same budget. All approaches discussed for computing the initial bid in max delivery campaigns can therefore be applied to derive an appropriate bid suggestion.

CHAPTER 2

BID RESPONSE PREDICTION

In this chapter, we demonstrate how to model response prediction—how different metrics (e.g., spend, conversions) respond to bid changes—by leveraging real-time auction data alongside prediction models. This approach helps address the data sparsity and signal delay issues encountered in deep-funnel conversion products when applying the MPC algorithm in previous chapters.

In this chapter, we discuss Bid Response Prediction. The bid response prediction model aims to understand how changes in bid prices can potentially impact the market and key campaign metrics. The bid landscape forecasting model is one example of such models, as it predicts how bid adjustments affect spending by analyzing the winning price distribution. We have also briefly described other models in Part II. For instance, in the MPC controller, we construct bid-to-spend, bid-to-conversions, and bid-to-cost-per-result functions by leveraging real-time auction data. In this chapter, we explore additional approaches that utilize predictive models to achieve this goal.

1 Bid Cost Prediction

In the previous chapter, when modeling bid response prediction, we assumed that the campaign follows an oCPM(optimized Cost Per Mille) charging model, meaning the campaign is charged per impression. Under this assumption, there is no need to account for signal delay issues, and we can directly use real-time bid-cost pairs to build the model.

However, as previously discussed, this assumption does not hold when the charging rule is based on actual results, particularly for ads with deep-funnel objectives such as post-click conversions, app installs, and lead generation. In such cases, conversions are sparse, and it may take hours or even days before a new conversion is observed.

In this section, we introduce a new method for predicting the bid-cost relationship in real-time bidding for in-house campaigns with deep conversion objectives. To ensure broader applicability, we no longer assume a second-price auction. Instead, we consider the generalized second-price (GSP) auction, in which multiple ad slots are auctioned simultaneously.

Let's first review how the GSP auction works in practice. For simplicity, we use pay-per-click(PPC) model as an example. Assume that for a given auction request, there are N campaigns optimizing for clicks and k ($k < N$) available ad slots. Let s_j denote the j-th slot from the top, p_i be the predicted click-through rate (CTR) for campaign i in the first ad slot. The position bias discount factor for slot j IS α_j, where:

$$\alpha_1 = 1 \geq \alpha_2 \geq \cdots \geq \alpha_k.$$

The predicted CTR for campaign i in slot j is then $p_i \cdot \alpha_j$. Suppose that bidder i submits a bid b_i per click. The GSP auction operates as follows:

- Compute the eCPM for the first ad slot for all campaigns based on their bids and predicted CTRs:
$$eCPM_i = b_i \cdot p_i.$$
- Rank all campaigns in descending order based on eCPM values. If necessary, reindex them so that:
$$eCPM_1 \geq eCPM_2 \geq \cdots \geq eCPM_N.$$
- Assign ad slots: The top-ranked campaign receives the first slot, the second-ranked campaign receives the second slot, and so on.

- Compute the pay-per-click (PPC) for each advertiser i and charge the bidder only if a click event occurs:
$$c_i = \frac{eCPM_{i+1}}{p_i}.$$
Here, $eCPM_{i+1}$ represents the next highest eCPM in the ranking, ensuring that each advertiser pays a price determined by the bidder ranked below them. This ensures that the cost-per-click (CPC) aligns with the generalized second-price rule, where advertisers are charged based on the minimum bid required to maintain their slot.

In the previous discussion, we assumed that the campaign follows an oCPM charging model. Under this model, advertisers are charged per impression, regardless of whether a click event occurs. As a result, it is feasible to collect bid-cost pairs without concern that most of the cost values will be zero. Under pay-per-click model, this does not work as most of cost values are 0. One workaround to get rid of this issue is to use accumulated pay per impression cost as the approximation to the pay-per-click cost by running the auction replay with real-time auction data, if the pCTR is unbiased, such approximation will be accurate when number of impressions is sufficient. More specifically, for all auction requests of the given campaign in the last pacing interval(suppose there are T requests), we collect the auction data $\{A_t\}_{t=1}^T$ with each A_t as:

$$A_t = \{p_t, eCPM_{t,1}, \cdots, eCPM_{t,k}\}$$

where k is the number of ad slots, p_t is the pCTR of this campaign for t-th auction request at the first ad slot, $\{eCPM_{t,j}\}_{j=1}^k$ are the first k eCPMs in the ranking stage(i.e., the eCPMs computed for the first ad slot) in t-th auction. We also assume the position bias decaying factors $\{\alpha_i\}_{i=1}^k$ are available and fixed across all requests.

We now compute the bid-cost prediction function $c = f(b)$, where f is the pay-per-impression spend rate function of bid price b. This function represents the expected spend of the campaign within one pacing interval[1].

For a fixed bid price b, the corresponding spend is computed by iterating through all auction requests $\{A_t\}_{t=1}^T$. For each auction A_t, we compute the eCPM of this campaign for the first ad slot as:

$$eCPM = b \cdot p_t.$$

If the campaign participates in auction A_t, it wins a slot if and only if:

$$eCPM \geq eCPM_{t,k}.$$

Otherwise, it loses the auction. If it wins, we determine the lowest slot $j(t)$ $(1 \leq j(t) \leq k)$ that the campaign qualifies for, such that:

$$eCPM_{t,j(t)-1} > eCPM > eCPM_{t,j(t)}.$$

[1]For simplicity, we assume that two adjacent pacing intervals have similar traffic patterns, predicted click-through rates (pCTR), and winning price distributions. Under this assumption, it is reasonable to use data from one interval to infer metrics in the subsequent interval.

This means that the campaign secures the $j(t)$-th ad slot in the t-th auction. The corresponding pay-per-impression cost for winning this slot is:

$$eCPM_{t,j(t)} \cdot \alpha_{j(t)},$$

where $\alpha_{j(t)}$ accounts for position bias adjustment.

The accumulated pay-per-impression cost over the pacing interval is then:

$$\sum_{t=1}^{T} eCPM_{t,j(t)} \cdot \alpha_{j(t)}.$$

This sum represents the expected cost for the pay-per-click (PPC) campaign over the pacing interval and defines the function $f(b)$, assuming that the predicted click-through rate p_t is unbiased for this campaign.

The procedure is illustrated in Figure 2.1. Each group of blue dots along the vertical lines represents the eCPMs of different ad slots in an auction. The dotted line indicates the eCPM of the campaign for a given bid.

The dots marked with a red "X" indicate the ad slots the campaign wins in each auction. The expected cost for the given bid is then computed as the accumulated sum of the pay-per-impression costs for each winning ad slot.

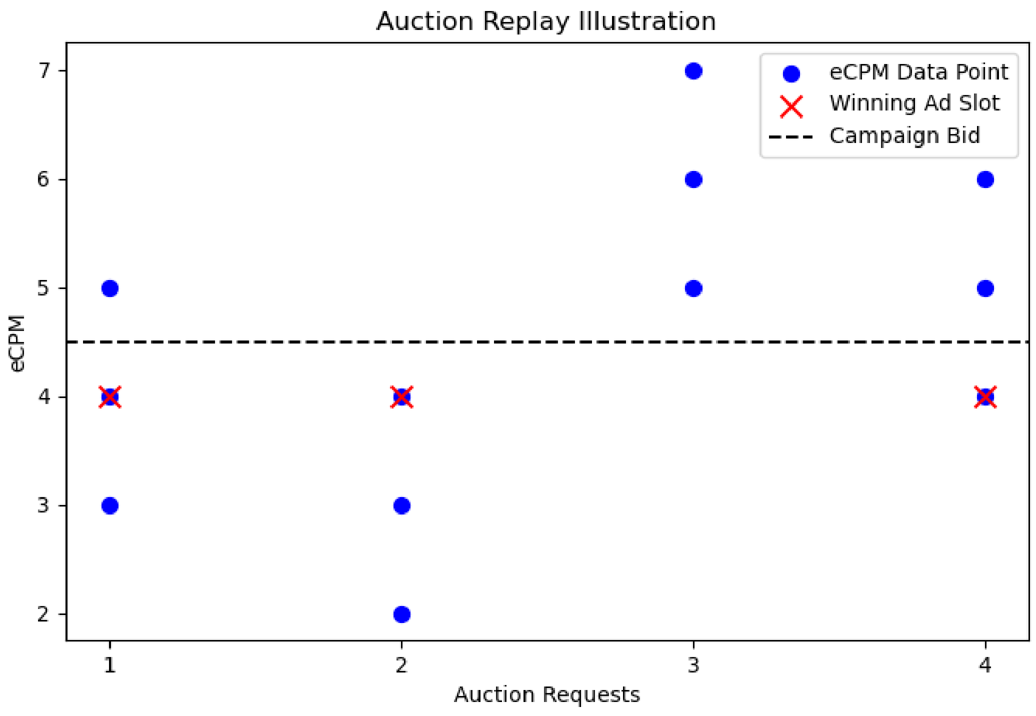


Figure 2.1: Illustration of Deriving Bid-Cost Relation via Auction Replay

We summarize the idea in the following algorithm

Algorithm 22 Bid-Cost Prediction Algorithm

Require:
- $\{A_t\}_{t=1}^{T}$ — Set of auction requests over the pacing interval, where each auction request A_t contains:
  - p_t — Predicted click-through rate (pCTR) for the campaign in auction A_t.
  - k — Number of available ad slots in the auction.
  - $\{eCPM_{t,j}\}_{j=1}^{k}$ — Sorted eCPMs of the top k advertisers who win ad slots.
- $\{\alpha_j\}_{j=1}^{k}$ — Position bias discount factors for each ad slot, where:

$$\alpha_1 = 1 \geq \alpha_2 \geq \cdots \geq \alpha_k.$$

- b_l, b_u — Lower and upper bounds of the bid range.
- Δb — Bid increment step size.

Ensure: Predicted bid-cost function $f(b)$ for $b \in [b_l, b_u]$.

```
1: for $b = b_l$ to $b_u$ with step size $\Delta b$ do          ▷ Iterate over bid values
2:     Initialize $f(b) \leftarrow 0$                          ▷ Initialize total cost for bid $b$
3:     for $t = 1$ to $T$ do                                   ▷ Iterate over all auction requests
4:         Extract $A_t = (p_t, \{eCPM_{t,j}\}_{j=1}^{k})$
5:         Compute $eCPM \leftarrow b \cdot p_t$                ▷ Calculate campaign eCPM for the first slot
6:         if $eCPM \geq eCPM_{t,k}$ then                       ▷ Check if the campaign wins a slot
7:             Find $j(t)$ such that $eCPM_{t,j(t)-1} > eCPM > eCPM_{t,j(t)}$
8:             Compute impression cost: $c_t \leftarrow eCPM_{t,j(t)} \cdot \alpha_{j(t)}$
9:             Update total cost: $f(b) \leftarrow f(b) + c_t$
10:        end if
11:    end for
12: end for
13: return $f(b)$                                              ▷ Return the bid-cost function $f(b)$
```

2 Bid Conversion Prediction

For completeness, we briefly discuss how to use the same method to model the bid-to-conversion relationship. The principle described in the previous section applies with only a slight modification.

We adopt the exact same problem configuration as in the previous section, except that in this case, we aim to compute the expected number of conversions (clicks) for a given bid level b. The procedure remains identical, with the only difference being that instead of computing the impression cost, we use the predicted click-through rate (pCTR) per impression to approximate the total number of clicks when the campaign wins the auction.

The algorithm is presented as follows:

Algorithm 23 Bid-Conversion Prediction Algorithm

Require:
- $\{A_t\}_{t=1}^T$ — Set of auction requests over the pacing interval, where each auction request A_t contains:
  - p_t — Predicted click-through rate (pCTR) for the campaign in auction A_t.
  - k — Number of available ad slots in the auction.
  - $\{eCPM_{t,j}\}_{j=1}^k$ — Sorted eCPMs of the top k advertisers who win ad slots.
- $\{\alpha_j\}_{j=1}^k$ — Position bias discount factors for each ad slot, where:

$$\alpha_1 = 1 \geq \alpha_2 \geq \cdots \geq \alpha_k.$$

- b_l, b_u — Lower and upper bounds of the bid range.
- Δb — Bid increment step size.

Ensure: Predicted bid-conversion function $g(b)$ for $b \in [b_l, b_u]$.

```
1:  for $b = b_l$ to $b_u$ with step size $\Delta b$ do                ▷ Iterate over bid values
2:      Initialize $g(b) \leftarrow 0$                              ▷ Initialize total cost for bid $b$
3:      for $t = 1$ to $T$ do                                      ▷ Iterate over all auction requests
4:          Extract $A_t = (p_t, \{eCPM_{t,j}\}_{j=1}^k)$
5:          if $eCPM \geq eCPM_{t,k}$ then                          ▷ Check if the campaign wins a slot
6:              Find $j(t)$ such that $eCPM_{t,j(t)-1} > eCPM > eCPM_{t,j(t)}$
7:              Compute position bias adjusted pCTR: $\tilde{p}_t \leftarrow p_t \cdot \alpha_{j(t)}$
8:              Update total cost: $g(b) \leftarrow g(b) + \tilde{p}_t$
9:          end if
10:     end for
11: end for
12: return $g(b)$                                   ▷ Return the bid-conversion function $g(b)$
```

3 Remarks

bid landscape: [23] [56] [69] [42] [66]

CHAPTER 3

BID SHADING

In this chapter, we discuss bid shading techniques, which are crucial for bidding algorithm design under the first-price auction model. We begin by examining the industry's shift from waterfall bidding to header bidding, which has made bid shading a necessary component of bidding algorithm design. Next, we demonstrate how bid shading algorithms can be developed under various campaign configurations. Additionally, we include a section outlining the structure of bidding algorithms under arbitrary auction mechanisms. Finally, we conclude the chapter by listing several winning probability estimation methods, an indispensable component of bid shading.

1 From Waterfall Bidding to Header Bidding

First price auction became more and more popular for real-time bidding due to the shift from the traditional waterfall bidding to header bidding.

1.1 Waterfall Bidding

Waterfall bidding is an early method used in programmatic advertising where ad impressions are sequentially offered to demand partners (e.g., DSPs or ad exchanges) based on a predefined priority order set by the publisher. If the first demand partner does not fill the impression, the request moves down the chain to the next partner, continuing until the impression is filled or no suitable ad is found.

For a given publisher (e.g., a website), when a user visits the platform, an ad request is triggered from the publisher's ad server. The ad server then calls the first demand partner (e.g., an SSP or an ad exchange) based on predefined priorities. If the demand partner provides an ad that meets or exceeds the publisher's floor price, the impression is filled, and the process stops. Otherwise, the ad server sends the request to the next available demand partner, continuing sequentially until the ad slot is filled. This method was dominant in programmatic advertising for a long time.

Although waterfall bidding is simple to implement, it has several significant limitations:

- **Sequential Process Delays Auction Speed:** Since requests are processed one by one, the decision-making process is inherently slow.
- **Lack of Fair Competition:** From a bidder's perspective, not all bidders get to participate simultaneously, as the auction operates based on a predefined priority list.
- **Potential Revenue Loss:** From a publisher's perspective, a lower-priority bidder may be willing to pay more than the first bidder but never gets the opportunity to compete, leading to potential revenue loss.

1.2 Header Bidding

Header bidding is a bidding strategy introduced around 2014–2015 to address the limitations of waterfall bidding. Unlike waterfall bidding, which runs auctions sequentially based on predefined priorities of demand partners, header bidding allows all demand partners to participate in the auction simultaneously. This ensures that the demand partner with the highest bid wins the ad slot, thereby increasing competition and boosting revenue for publishers.

Challenges for DSPs Under Header Bidding: With the advent of header bidding, the traditional second-price auction strategy used by DSPs faces new challenges. To illustrate the issue, consider the following example:

Suppose two DSPs, D_1 and D_2, compete for an ad request from a publisher. Within each DSP i, there are two campaigns, $C_{i,1}$ and $C_{i,2}$, interested in this impression. For simplicity, assume all campaigns submit bids for the impression as follows:

$$C_{1,1} = \$1, \quad C_{1,2} = \$4, \quad C_{2,1} = \$2, \quad C_{2,2} = \$3.$$

When the ad request arrives, each DSP conducts an internal auction among its campaigns using a second-price auction:

- **DSP 1:** Campaign $C_{1,2}$ wins the internal auction but, under the second-price rule, submits the second-highest price within DSP 1, which is **$1**, to the ad server.
- **DSP 2:** Campaign $C_{2,2}$ wins the internal auction and submits **$2**, the second-highest price within DSP 2, to the ad server.

Under header bidding, DSP 1 and DSP 2 submit bids of **$1** and **$2**, respectively, to the ad server. Regardless of whether the ad server runs a **first-price** or **second-price** auction, DSP 2 will ultimately win the impression.

The Problem: This scenario highlights a fundamental inefficiency when we run second price auction with DSPs in header bidding:

- **From the bidder's perspective:** The campaign with the highest original bid, $C_{1,2}$ (\$4), does not win the auction. This contradicts the expected outcome of an efficient auction system, where the highest bidder should secure the impression.
- **From the publisher's perspective:** There is a clear revenue loss. If all campaigns participated in a unified auction directly, the second-highest bid among them would be **$3**, meaning the publisher could have earned **$3** instead of **$2** (under a first-price auction) or **$1** (under a second-price auction at the ad server level).

This example illustrates how the traditional second-price auction model within DSPs leads to inefficiencies in the presence of header bidding, ultimately resulting in suboptimal outcomes for both advertisers and publishers.

Another scenario arises when a publisher sends requests to multiple SSPs, and each SSP, in turn, forwards these requests to multiple DSPs. If the SSP runs a second-price auction, the same issues persist, reinforcing the challenges inherent in this model.

These inefficiencies have been a driving force behind the industry's shift in recent years from second-price auctions to first-price auctions, aiming to enhance transparency and improve fairness in programmatic advertising. In this chapter, we will explore strategies for designing optimal bidding approaches under first-price auctions.

2 Bid Shading under First Price Auction

In this section, we explore the design of bidding strategies under first-price auctions. The general formulation follows a similar structure to that used in second-price auctions, as discussed in Part II. The bidding problem can be framed as an optimization problem that seeks to maximize a specific objective while satisfying given constraints, such as a budget constraint in the case of the max delivery problem.

The objective can take different forms:

- **Welfare Maximization**: The goal is to maximize the total number of conversions for a given campaign.
- **Utility Maximization**: The objective is to maximize the campaign's surplus.

Additionally, other factors must be considered. For instance, if a DSP bids on behalf of a campaign for external impressions, the cost incurred by the campaign represents only part of the equation. The DSP often applies a markup to generate additional profit for the platform itself. In such cases, margin profit optimization becomes an important factor in the bidding strategy.

This chapter will discuss these variations in detail, including strategies to handle different optimization objectives and practical challenges in first-price auction environments.

2.1 Welfare Maximization

We discuss how to solve the max delivery problem in a first-price auction (FPA) setting. One possible formulation of the bidding problem is to optimize the total welfare (conversions) of a campaign for a given budget, similar to our previous discussion in Part II.

Problem Formulation We assume that a campaign from a DSP is bidding for external ad impressions through an ad exchange running a first-price auction. For simplicity, we consider a CPM campaign. The problem can be formulated as:

$$\begin{aligned} \max_{b_t \geq 0} \quad & \sum_{t=1}^{T} P_t(b_t) \\ \text{s.t.} \quad & \sum_{t=1}^{T} P_t(b_t) \cdot b_t \leq B. \end{aligned} \tag{3.1}$$

where:

- $P_t(b_t)$ [1] represents the probability of winning the impression at the t-th auction given bid per impression b_t.
- B is the total budget and T is the total number of auction opportunities.

If at time t, the highest competing bid from other bidders(winning price) is c_t, then the probability of winning is given by:

$$P_t(b_t) = \mathbb{P}(b_t \geq c_t).$$

$P_t(b_t)$ is typically increasing and nonlinear with respect to b_t, making direct optimization challenging.

[1] Recall that under a second-price auction, we write $P_t(b_t)$ as $\mathbb{1}_{\{b_t > c_t\}}$, where c_t represents the supporting price. For a Demand-Side Platform (DSP) or an ad network (adNet), the value of c_t is not always available, as the data are often censored. Ad exchanges often do not disclose losing bids or even the winning bid if the DSP is not the winner. In practice, DSPs often only see partial information about the auction results. The DSP typically knows if they won or lost. If they win, they know the price they paid (their own bid). If they lose, they usually don't know the winning bid.

Derivation of the Optimal Bid To solve for the optimal bid b_t^*, we introduce the Lagrangian function with multiplier $\lambda \geq 0$:

$$\mathcal{L}(b_t, \lambda) = \sum_{t=1}^{T} P_t(b_t) - \lambda \left(\sum_{t=1}^{T} P_t(b_t) b_t - B \right).$$

Differentiating $\mathcal{L}$ with respect to b_t:

$$\begin{aligned} \frac{\partial}{\partial b_t} \mathcal{L}(b_t, \lambda) &= \frac{\partial}{\partial b_t} \left[P_t(b_t) - \lambda P_t(b_t) b_t \right] \\ &= P_t^{'}(b_t) - \lambda b_t P_t^{'}(b_t) - \lambda P_t(b_t). \end{aligned}$$

To find the optimal bid b_t^*, we set $\frac{\partial \mathcal{L}(b_t^*, \lambda^*)}{\partial b_t} = 0$:

$$P_t^{'}(b_t^*) - \lambda^* b_t^* P_t^{'}(b_t^*) - \lambda^* P_t(b_t^*) = 0.$$

Rearranging for b_t^*:

$$b_t^* + \frac{P_t(b_t^*)}{P_t^{'}(b_t^*)} = \frac{1}{\lambda^*} \tag{3.2}$$

Since $P_t(b_t)$ is generally nonlinear, Equation 3.2 has no closed-form solution in most cases. Typically, numerical methods such as Newton's method or other root-finding algorithms are used to compute b_t^* iteratively. However, in some special cases where $P_t(b_t)$ has a tractable form, an analytical solution may exist.

To solve for b_t^*, we need to determine λ^* and the winning probability function $P_t(b_t)$ (hence its derivative $P_t'(b_t)$):

- **Determining λ^* [2].** To determine λ^*, note that if the budget constraint is active, the KKT condition implies that the optimal value of λ must satisfy the budget exactly:

  $$\sum_{t=1}^{T} P_t(b_t(\lambda^*)) \cdot b_t(\lambda^*) = B.$$

  where $b_t(\lambda^*)$ is obtained from (3.2). If we assume that $P_t(\cdot)$ is log-concave (which holds for most distributions discussed later), we can show that the function:

  $$b + \frac{P_t(b)}{P_t^{'}(b)}$$

  is strictly increasing with respect to b. Therefore, to find λ^*, we can leverage the PID controller discussed in Part II to iteratively update λ based on the pacing status (i.e., whether delivery is ahead or behind schedule).

- **Determining P_t:** $P_t(b_t)$ is the winning probability function, which depends on both b_t and the auction environment (e.g., the bids submitted by other bidders). It can be derived using either parametric methods (where we assume some parametrized distributions a priori and fit the distribution using real data) or non-parametric methods (e.g., quantile regression). We will discuss these methods in detail in section 4.

[2]We may also derive the update rule for λ using the DOGD method, which we discuss in subsection 5.1.

Assuming that both λ^* and P_t are determined and that P_t is log-concave, we solve for b_t^* using a numerical method. In practice, we search for b_t^* within a predefined range $[b_l, b_u]$. Since the left-hand side of Equation 3.2 is monotonically increasing with respect to b_t^*, a binary search method can be applied to efficiently determine the solution to Equation 3.2. The algorithm we discuss here is summarized in Algorithm 24.

Algorithm 24 Bid Strategy for Max Delivery Welfare Maximization Problem under FPA

```
Require: $B$: Total budget; $T$: Total auction opportunities;
 1: $\lambda_0$: Initial value for Lagrange multiplier;
 2: $[b_l, b_u]$: Search range for bid price; $\epsilon$: threshold for binary search;
 3: $P_t(b), P_t'(b)$: Winning probability function and its derivative;
 4: $PID$: PID controller in Algorithm 3;
Ensure: $b_t^*$: Optimal bid at auction opportunity $t$.
 5: Step 1: Initialize Parameters
 6: Set initial Lagrange multiplier $\lambda \leftarrow \lambda_0$
 7: Define budget pacing target: $\frac{B}{T}$
 8: Step 2: Solve for Optimal $b_t^*$
 9: for each pacing interval do
10:     Observe actual spend $S$
11:     Update $\lambda$ using PID controller: $\lambda \leftarrow PID(S, B, T, \lambda)$
12:     for each auction opportunity $t$ do
13:         Perform Binary Search to Solve for $b_t^*$
14:         Initialize search range: $b_{\text{low}} \leftarrow b_l$, $b_{\text{high}} \leftarrow b_u$
15:         while $b_{\text{high}} - b_{\text{low}} > \epsilon$ do
16:             Set $b_{\text{mid}} \leftarrow \frac{b_{\text{low}} + b_{\text{high}}}{2}$
17:             Compute $F(b_{\text{mid}}) = P_t'(b_{\text{mid}}) - \lambda b_{\text{mid}} P_t'(b_{\text{mid}}) - \lambda P_t(b_{\text{mid}})$
18:             if $F(b_{\text{mid}}) = 0$ then
19:                 $b_t^* \leftarrow b_{\text{mid}}$
20:                 break
21:             else if $F(b_{\text{mid}}) > 0$ then
22:                 Set $b_{\text{low}} \leftarrow b_{\text{mid}}$
23:             else
24:                 Set $b_{\text{high}} \leftarrow b_{\text{mid}}$
25:             end if
26:         end while
27:         Set $b_t^* \leftarrow b_{\text{mid}}$
28:     end for
29: end for
30: Step 3: Return Optimal Bids
31: return $b_t^*$ for all $t$
```

Interpretation of the Bidding Formula Rearranging Equation 3.2, we obtain:

$$b_t^* = \frac{1}{\lambda^*} - \frac{P_t(b_t^*)}{P_t'(b_t^*)}.$$

Since $P_t(b) = \mathbb{P}(b \geq c_t)$ represents the cumulative distribution function (CDF) of the winning price c_t, its derivative $P_t'(b)$ corresponds to the probability density function (PDF). Typically, $P_t'(b)$ is positive, implying that the ratio $P_t(b)/P_t'(b)$ is also positive.

Comparing this to the optimal bidding formula for the max delivery problem under second-price auctions (see (1.3)), we observe that:

- The term $\frac{1}{\lambda^*}$ acts as a **base bid** determined by the pacing multiplier.
- The positive correction term $\frac{P_t(b_t^*)}{P_t'(b_t^*)}$ serves as a **bid shading factor**, which depends on the winning probability distribution $P_t(\cdot)$ and adjusts the base bid accordingly.

This formulation captures the strategic nature of bid adjustments under first-price auctions, where bidders optimize their bids based on market competition.

2.2 Utility Maximization

We analyze the utility maximization problem for a maximum delivery campaign under a first-price auction in this section.

Problem Formulation The optimization framework presented above seeks to maximize the total number of conversions. In this context, the advertiser aims to secure as many conversions as possible while ensuring that the total expenditure remains within the allocated budget, without explicitly considering the individual valuation of each conversion.

However, in many scenarios, the advertiser assigns a specific valuation v_t to each conversion. As established in the Vickrey-Clarke-Groves (VCG) auction framework, an advertiser may wish to optimize for *utility* (or *surplus*), which is defined as the difference between the valuation and the corresponding payment. Consequently, the objective shifts to maximizing the expected surplus, leading to the following optimization problem:

$$\begin{aligned} \max_{b_t \geq 0} \quad & \sum_{t=1}^{T} P_t(b_t)\,[v_t - b_t] \\ \text{s.t.} \quad & \sum_{t=1}^{T} P_t(b_t) \cdot b_t \leq B. \end{aligned} \tag{3.3}$$

Derivation of the Optimal Bid We employ a Lagrangian approach similar to the one used previously, introducing $\lambda \geq 0$ as the Lagrange multiplier for the budget constraint:

$$\mathcal{L}(b_t, \lambda) = \sum_{t=1}^{T} P_t(b_t)\,[v_t - b_t] - \lambda \left(\sum_{t=1}^{T} P_t(b_t) b_t - B \right).$$

Differentiating with respect to b_t:

$$\begin{aligned}\frac{\partial}{\partial b_t}\mathcal{L}(b_t,\lambda) &= \frac{\partial}{\partial b_t}\left[P_t(b_t)\left[v_t - b_t\right] - \lambda P_t(b_t) b_t\right] \\ &= P_t^{'}(b_t)\left[v_t - b_t(1+\lambda)\right] - \left[1+\lambda\right] P_t(b_t).\end{aligned}$$

Setting $\frac{\partial \mathcal{L}(b_t^*,\lambda^*)}{\partial b_t} = 0$:

$$P_t^{'}(b_t)\left[v_t - b_t(1+\lambda)\right] - \left[1+\lambda\right] P_t(b_t) = 0.$$

Rearranging for b_t^*, we obtain:

$$b_t^* + \frac{P_t(b_t^*)}{P_t^{'}(b_t^*)} = \frac{v_t}{1+\lambda^*}. \tag{3.4}$$

To determine b_t^*, we first solve for λ^* and define $P_t(\cdot)$ analogously to the procedure outlined for (3.2). We then apply a binary search to identify the optimal bid b_t^* within a predefined range $[b_l, b_u]$. The complete procedure is summarized in Algorithm 25.

Interpretation of the Bidding Formula Rearranging (3.4), we obtain:

$$b_t^* = \frac{v_t}{1+\lambda^*} - \frac{P_t(b_t^*)}{P_t^{'}(b_t^*)}.$$

This formulation closely parallels the conversion maximization formula (3.2). In fact, under the same settings, except replacing the first-price auction with a second-price auction, the optimal bid simplifies to:

$$b_{t,\text{second}}^* = \frac{v_t}{1+\lambda^*}.$$

Thus, the first term, $\frac{v_t}{1+\lambda^*}$, can be interpreted as a base bid governed by the pacing multiplier. The second term, $\frac{P_t(b_t^*)}{P_t^{'}(b_t^*)}$, serves as a shading factor that adjusts the bid downward, reflecting the bidder's strategic behavior in a first-price auction environment.

2.3 Marginal Profit Optimization

The bidding problems discussed so far assume that all payments (revenues) remain within the platform itself. However, consider a scenario where a Demand-Side Platform (DSP) or an Ad Network (AdNet) bids for external ad exchange or Supply-Side Platform (SSP) ad slots on behalf of an advertiser. In such cases, additional pricing adjustments are necessary to ensure profitability for the bidding platform (DSP or AdNet). This marginal profit requirement introduces additional complexities beyond the standard formulations discussed earlier, necessitating a revised optimization framework that accounts for both the advertiser's objectives and the platform's profitability.

Problem Formulation To incorporate the platform's profit margin, we introduce a markup factor $m > 0$ and modify the optimization problem in (3.1) as follows:

Algorithm 25 Bid Strategy for Max Delivery Utility Maximization Problem under FPA

```
Require: $B$: Total budget; $T$: Total auction opportunities;
 1: $\lambda_0$: Initial value for Lagrange multiplier;
 2: $[b_l, b_u]$: Search range for bid price; $\epsilon$: threshold for binary search;
 3: $P_t(b), P_t^{'}(b)$: Winning probability function and its derivative;
 4: $PID$: PID controller in Algorithm 3;
Ensure: $b_t^*$: Optimal bid at auction opportunity $t$.
 5: Step 1: Initialize Parameters
 6: Set initial Lagrange multiplier $\lambda \leftarrow \lambda_0$
 7: Define budget pacing target: $\frac{B}{T}$
 8: Step 2: Solve for Optimal $b_t^*$
 9: for each pacing interval do
10:     Observe actual spend $S$
11:     Update $\lambda$ using PID controller: $\lambda \leftarrow PID(S, B, T, \lambda)$
12:     for each auction opportunity $t$ do
13:         Perform Binary Search to Solve for $b_t^*$
14:         Compute value of auction opportunity: $v_t$
15:         Initialize search range: $b_{\text{low}} \leftarrow b_l$, $b_{\text{high}} \leftarrow b_u$
16:         while $b_{\text{high}} - b_{\text{low}} > \epsilon$ do
17:             Set $b_{\text{mid}} \leftarrow \frac{b_{\text{low}}+b_{\text{high}}}{2}$
18:             Compute $F(b_{\text{mid}}) = P_t^{'}(b_{\text{mid}})v_t - (1+\lambda)b_{\text{mid}}P_t^{'}(b_{\text{mid}}) - (1+\lambda)P_t(b_{\text{mid}})$
19:             if $F(b_{\text{mid}}) = 0$ then
20:                 $b_t^* \leftarrow b_{\text{mid}}$
21:                 break
22:             else if $F(b_{\text{mid}}) > 0$ then
23:                 Set $b_{\text{low}} \leftarrow b_{\text{mid}}$
24:             else
25:                 Set $b_{\text{high}} \leftarrow b_{\text{mid}}$
26:             end if
27:         end while
28:         Set $b_t^* \leftarrow b_{\text{mid}}$
29:     end for
30: end for
31: Step 3: Return Optimal Bids
32: return $b_t^*$ for all $t$
```

$$\begin{aligned} \max_{b_t \geq 0} \quad & \sum_{t=1}^{T} P_t(b_t) \\ \text{s.t.} \quad & \sum_{t=1}^{T} P_t(b_t) \cdot b_t \cdot (1+m) \leq B. \end{aligned} \tag{3.5}$$

Compared to (3.1), an additional markup $b_t \cdot m$ is applied to each impression, ensuring that the platform (DSP or AdNet) earns a margin on top of the campaign's bid price.

Derivation of the Optimal Bid The approach to solving (3.5) follows the same methodology as (3.1). We introduce the Lagrangian:

$$\mathcal{L}(b_t, \lambda) = \sum_{t=1}^{T} P_t(b_t) - \lambda \left(\sum_{t=1}^{T} P_t(b_t) \cdot b_t \cdot (1+m) - B \right).$$

By differentiating with respect to b_t and following the same steps as before, we obtain the counterpart of (3.2):

$$b_t^* + \frac{P_t(b_t^*)}{P_t'(b_t^*)} = \frac{1}{\lambda^*(1+m)}. \tag{3.6}$$

The procedure for determining λ^* and estimating P_t remains unchanged from previous formulations. For completeness, we summarize the bidding strategy with marginal profit in Algorithm 26.

3 Bidding under Arbitrary Auction

In previous sections, we explored how different auction mechanisms influence bidding dynamics. Beyond first-price and second-price auctions, other widely used auction mechanisms in practice include Generalized Second Price (GSP) and Vickrey-Clarke-Groves (VCG). In this section, we present a framework for designing an optimal bidding strategy under an arbitrary auction mechanism. A similar treatment can be found in [32].

Problem Formulation We consider the max delivery problem for an oCPM ad campaign that aims to maximize total conversions. Suppose there are T auction opportunities and a total budget B. The optimization problem under an arbitrary auction mechanism can be formulated as follows:

$$\begin{aligned} \max_{b_t \geq 0} \quad & \sum_{t=1}^{T} r_t \cdot G_t(b_t) \\ \text{s.t.} \quad & \sum_{t=1}^{T} H_t(b_t) \leq B. \end{aligned} \tag{3.7}$$

where:

Algorithm 26 Bid Strategy for Max Delivery Welfare Maximization Problem with Marginal Profit under FPA

```
Require: $B$: Total budget; $T$: Total auction opportunities; $m$: Markup factor;
 1: $\lambda_0$: Initial value for Lagrange multiplier;
 2: $[b_l, b_u]$: Search range for bid price; $\epsilon$: threshold for binary search;
 3: $P_t(b), P_t^{'}(b)$: Winning probability function and its derivative;
 4: $PID$: PID controller in Algorithm 3;
Ensure: $b_t^*$: Optimal bid at auction opportunity $t$.
 5: Step 1: Initialize Parameters
 6: Set initial Lagrange multiplier $\lambda \leftarrow \lambda_0$
 7: Define budget pacing target: $\frac{B}{T}$
 8: Step 2: Solve for Optimal $b_t^*$
 9: for each pacing interval do
10:     Observe actual spend $S$
11:     Update $\lambda$ using PID controller: $\lambda \leftarrow PID(S, B, T, \lambda)$
12:     for each auction opportunity $t$ do
13:         Perform Binary Search to Solve for $b_t^*$
14:         Initialize search range: $b_{\text{low}} \leftarrow b_l$, $b_{\text{high}} \leftarrow b_u$
15:         while $b_{\text{high}} - b_{\text{low}} > \epsilon$ do
16:             Set $b_{\text{mid}} \leftarrow \frac{b_{\text{low}}+b_{\text{high}}}{2}$
17:             Compute $F(b_{\text{mid}}) = P_t^{'}(b_{\text{mid}}) - \lambda \cdot (1+m) \cdot b_{\text{mid}} P_t^{'}(b_{\text{mid}}) - \lambda \cdot (1+m) \cdot P_t(b_{\text{mid}})$
18:             if $F(b_{\text{mid}}) = 0$ then
19:                 $b_t^* \leftarrow b_{\text{mid}}$
20:                 break
21:             else if $F(b_{\text{mid}}) > 0$ then
22:                 Set $b_{\text{low}} \leftarrow b_{\text{mid}}$
23:             else
24:                 Set $b_{\text{high}} \leftarrow b_{\text{mid}}$
25:             end if
26:         end while
27:         Set $b_t^* \leftarrow b_{\text{mid}}$
28:     end for
29: end for
30: Step 3: Return Optimal Bids
31: return $b_t^*$ for all $t$
```

- $G_t(b_t)$ denotes the probability of winning the t-th auction given a per-impression bid b_t.
- $H_t(b_t)$ represents the expected cost incurred for the t-th auction opportunity.
- r_t represents the predicted conversion rate for the t-th auction.

For analytical convenience, we impose the following regularity conditions:

- Both G_t and H_t are differentiable functions with first derivatives defined as:
$$g_t = G_t'(b_t), \quad h_t = H_t'(b_t).$$
- Boundary conditions: $G_t(0) = 0$, $G_t(+\infty) = 1$, and $g_t \geq 0$; $H_t(0) = 0$ and $h_t \geq 0$.

These conditions hold for most auction mechanisms in practice, including first-price and second-price auctions, GSP, Myerson's optimal auction, and VCG.

Derivation of the Optimal Bid To solve (3.7), we introduce the Lagrangian function with multiplier $\lambda \geq 0$:

$$\mathcal{L}(b_t, \lambda) = \sum_{t=1}^{T} r_t \cdot G_t(b_t) - \lambda \left(\sum_{t=1}^{T} H_t(b_t) - B \right).$$

The optimal bid b_t^* satisfies the first-order condition:

$$\frac{\partial}{\partial b_t} \mathcal{L}(b_t^*, \lambda^*) = 0.$$

Expanding the derivative, we obtain:

$$r_t \cdot g_t(b_t^*) - \lambda^* h_t(b_t^*) = 0.$$

Solving for b_t^*, we derive:

$$b_t^* = \left(\frac{h_t}{g_t} \right)^{-1} \left(\frac{r_t}{\lambda^*} \right), \tag{3.8}$$

where $(h_t/g_t)^{-1}$ denotes the inverse function of h_t/g_t.

Interpretation of the Bidding Formula This formulation provides a general method for determining the optimal bid across various auction mechanisms. The exact computation of b_t^* depends on the specific forms of $G_t(b_t)$ and $H_t(b_t)$, which we will analyze below:

- **First Price Auction:** In the first price auction under the welfare maximization problem, we have:
$$G_t(b_t) = P_t(b_t), \quad H_t(b_t) = P_t(b_t) \cdot b_t.$$

Computing the derivatives, we obtain:

$$\frac{H_t'(b_t)}{G_t'(b_t)} = \frac{p_t(b_t) \cdot b_t + P_t(b_t)}{p_t(b_t)}.$$

Substituting into (3.8), we get:

$$\frac{p_t(b_t) \cdot b_t + P_t(b_t)}{p_t(b_t)} = \frac{r_t}{\lambda^*}.$$

For CPM campaigns where $r_t = 1.0$, this equation exactly matches (3.2).

Similarly, for the utility maximization problem, (3.8) recovers (3.4) through analogous derivations, which we leave as an exercise for the reader.

- **Second Price Auction:** By Myerson's Lemma, for any dominant strategy incentive-compatible (DSIC) auction, we have:

$$H_t(b_t) = b_t \cdot G_t(b_t) - \int_0^{b_t} G_t(z)\mathrm{d}z.$$

Taking the derivative on both sides:

$$h_t(b_t) = g_t(b_t) \cdot b_t + G_t(b_t) - G_t(b_t),$$

which simplifies to:

$$\frac{h_t(b_t)}{g_t(b_t)} = b_t.$$

Since h_t/g_t is an identity function, its inverse is also the identity function. Therefore, (3.8) reduces to:

$$b_t^* = \frac{r_t}{\lambda^*}.$$

We recover the well-known optimal bidding strategy for second-price auctions. More generally, since Myerson's Lemma applies to any DSIC auction mechanism, we can assert that the same bidding formula is optimal, e.g., for the VCG auction.

- **General Auction:** The general form of the auction mechanism determines the exact expressions for $G_t(b_t)$ and $H_t(b_t)$. By applying the optimal bidding framework, we can derive bid adjustments tailored to specific auction environments. The key challenge lies in correctly estimating $P_t(b_t)$ and $Q_t(b_t)$, which we discuss in section 4.

4 Winning Probability Estimation

From previous discussions we can see that one indispensible piece of information we need in bid shading strategy is the probability winning function $P_t(b)$ for a given bid level b. In this section, we briefly introduce two main approaches to estimate $P_t(b)$: parametric methods and non-parametric methods.

4.1 Parametric Methods

Parametric methods assume that the winning price c_t follows a known distribution or that the winning probability function $P_t(\cdot)$ follows a specific parametric form. Historical data is then used to estimate the parameters of this distribution or function.

Here, we discuss an approach proposed by [55], where they assume $P_t(\cdot)$ follows a sigmoid function:

$$P_t(b|X;\omega,\beta) = \frac{1}{1+\mathrm{e}^{-\left(w_0+\sum_{i=1}^{k} w_i\cdot x_i+\beta\cdot\log(b)\right)}}$$

where:

- $X = \{x_i\}$ are features characterizing the campaign, user, and bidding environment. Examples of such features include campaign type, device type, and temporal attributes (e.g., day of the week, hour of the day, etc.).
- $\omega = \{w_i\}$ and $\beta > 0$ are model parameters to be estimated using historical bidding data.

The use of $\log(b)$ instead of b in the sigmoid function ensures that $P_t(b|X;\omega,\beta) \to 0$ as $b \to 0$ and $P_t(b|X;\omega,\beta) \to 1$ as $b \to +\infty$, which aligns with the expected behavior of a winning probability function.

We will prove in the Remarks section of this chapter that $P_t(b)$ defined in this manner is log-concave with respect to b. This property then allows us to directly apply the bid shading strategies from Algorithm 24, Algorithm 25, and Algorithm 26.

Other parametric methods have been proposed based on various probability distributions. We list a few notable examples below; readers interested in further details can refer to the corresponding papers.

- **Gamma Distribution:** [78]
- **Gaussian Distribution:** [69]
- **Gumbel Distribution:** [68]
- **Lognormal Distribution:** [76]
- **Mixture of Gaussians:** [33]

4.2 Non-Parametric Methods

An alternative approach is to use non-parametric methods to estimate the winning probability. Quantile regression is a natural fit for estimating the cumulative distribution function (CDF) in scenarios where data are censored, such as in first-price auction environments in RTB. We outline the core idea here; for a more comprehensive introduction, see [34] and [41].

In our setting, quantile regression aims to estimate the inverse of the winning probability function $P(b \mid X)$. We partition the interval $[0,1]$ into buckets $0 \le \tau_0 < \tau_1 < \cdots < \tau_i < \cdots < \tau_n = 1$. For each quantile τ, define

$$q_\tau(X) = \inf\{b : P(b \mid X) \ge \tau\}.$$

The goal is to learn an estimator $\hat{q}_\tau(X \mid \omega)$ of $q_\tau(X)$, parameterized by ω, based on the observed auction data.

Suppose the auction data take the form

$$\{\mathcal{A}_t\}_{t=1}^T = \{b_t, X_t, y_t\}_{t=1}^T,$$

where:

- b_t is the bid price submitted in the t-th auction,
- X_t denotes the feature vector characterizing the auction context,
- y_t is an indicator variable for whether the campaign won the t-th auction opportunity ($y_t = 1$ for a win, $y_t = 0$ for a loss).

Let the supporting price c_t follow some unknown i.i.d. distribution. For each training sample $\mathcal{A}_t$, the *pinball loss* for quantile regression is computed as

$$\ell_\tau\big(\hat{q}_\tau(X_t \mid \omega); b_t, y_t\big) = (1 - y_t) \cdot \tau \cdot [\, b_t - \hat{q}_\tau(X_t \mid \omega)\,]_+ + y_t \cdot (1 - \tau) \cdot [\, \hat{q}_\tau(X_t \mid \omega) - b_t\,]_+,$$

where $[x]_+ = \max(0, x)$. The model parameters ω are then learned by solving

$$\omega^* = \arg\min_\omega \sum_{t=1}^T \ell_\tau\big(\hat{q}_\tau(X_t \mid \omega); b_t, y_t\big).$$

Given the estimated quantile functions $\{\hat{q}_\tau(X \mid \omega^*)\}_{\tau=0}^n$, we can recover the winning probability $P(b \mid X)$ as follows: find the bucket k such that

$$\hat{q}_{\tau_k}(X \mid \omega^*) \le b < \hat{q}_{\tau_{k+1}}(X \mid \omega^*),$$

and linearly interpolate to approximate

$$P(b \mid X) \approx \hat{P}(b \mid X) = \tau_k + (\tau_{k+1} - \tau_k) \cdot \frac{b - \hat{q}_{\tau_k}(X \mid \omega^*)}{\hat{q}_{\tau_{k+1}}(X \mid \omega^*) - \hat{q}_{\tau_k}(X \mid \omega^*)}.$$

The derivative can be similarly approximated as

$$P'(b \mid X) \approx \frac{\tau_{k+1} - \tau_k}{\hat{q}_{\tau_{k+1}}(X \mid \omega^*) - \hat{q}_{\tau_k}(X \mid \omega^*)}.$$

We summarize the quantile regression algorithm in the following Algorithm 27:

Algorithm 27 Estimation of Winning Probability $P(b \mid X)$ and Its Derivative $P'(b \mid X)$ via Quantile Regression

```
Require: Training data $\{\mathcal{A}_t\}_{t=1}^T = \{b_t, X_t, y_t\}_{t=1}^T$
    where $b_t$: bid price, $X_t$: feature vector, $y_t \in \{0,1\}$: win indicator
Require: Quantile levels $0 \le \tau_0 < \tau_1 < \cdots < \tau_n = 1$
1: Train quantile regression models:
2: for each quantile $\tau_i$ do
3:     Compute pinball loss for auction data:
       $$\ell_{\tau_i}(\hat{q}_{\tau_i}(X_t \mid \omega); b_t, y_t) = (1-y_t)\cdot\tau_i\cdot[\,b_t - \hat{q}_{\tau_i}(X_t \mid \omega)\,]_+ + y_t\cdot(1-\tau_i)\cdot[\,\hat{q}_{\tau_i}(X_t \mid \omega) - b_t\,]_+,$$
       where $[x]_+ = \max(0, x)$.
4:     Learn parameters by minimizing empirical loss:
       $$\omega^* = \arg\min_{\omega} \sum_{t=1}^{T} \ell_{\tau_i}\big(\hat{q}_{\tau_i}(X_t \mid \omega); b_t, y_t\big).$$
5: end for
6: Given a bid $b$ and feature vector $X$:
7: Find bucket $k$ such that
       $$\hat{q}_{\tau_k}(X \mid \omega^*) \le b < \hat{q}_{\tau_{k+1}}(X \mid \omega^*).$$
8: Estimate winning probability:
       $$P(b \mid X) \approx \tau_k + (\tau_{k+1} - \tau_k)\cdot\frac{b - \hat{q}_{\tau_k}(X \mid \omega^*)}{\hat{q}_{\tau_{k+1}}(X \mid \omega^*) - \hat{q}_{\tau_k}(X \mid \omega^*)}.$$
9: Estimate derivative:
       $$P'(b \mid X) \approx \frac{\tau_{k+1} - \tau_k}{\hat{q}_{\tau_{k+1}}(X \mid \omega^*) - \hat{q}_{\tau_k}(X \mid \omega^*)}.$$
Ensure: Approximations of $P(b \mid X)$ and $P'(b \mid X)$
```

There are some other non-parametric approaches, for example, one can check: non-parametric estimations: [39], [74], deep learning method: [56] [76] other: [70] [24] point estimation(estimate minbid directly): [35] ,

censored vs non-censored(importance weight, distribution shif due to bid range)

5 Remarks

5.1 λ Update with DOGD

In solving (3.1), we previously mentioned that the PID controller method can be used to update λ online, as the optimal λ^* is the value that exactly depletes the total budget. Here, we introduce an alternative approach by applying the DOGD (Dual Online Gradient Descent) method. We apply this method to the more general problem (3.7):

$$\begin{aligned}\max_{b_t \geq 0} \quad & \sum_{t=1}^{T} r_t \cdot G_t(b_t) \\ \text{s.t.} \quad & \sum_{t=1}^{T} H_t(b_t) \leq B.\end{aligned}$$

The Lagrangian for this problem is:

$$\mathcal{L}(b_t, \lambda) = \sum_{t=1}^{T} \left[r_t \cdot G_t(b_t) - \lambda H_t(b_t) + \lambda \cdot \frac{B}{T} \right]$$

The corresponding dual problem is:

$$\mathcal{L}(\lambda) = \max_{b_t \geq 0} \mathcal{L}(b_t, \lambda)$$

As derived in (3.8), the maximum is attained when:

$$b_t = b_t(\lambda) = \left(\frac{h_t}{g_t} \right)^{-1} \left(\frac{r_t}{\lambda} \right)$$

where h_t and g_t represent the derivative functions of H_t and G_t, respectively. Substituting this into the Lagrangian gives:

$$\mathcal{L}(\lambda) = \sum_{t=1}^{T} \left[r_t \cdot G_t(b_t(\lambda)) - \lambda H_t(b_t(\lambda)) + \lambda \cdot \frac{B}{T} \right] = \sum_{t=1}^{T} L_t(\lambda)$$

with:

$$L_t(\lambda) = r_t \cdot G_t(b_t(\lambda)) - \lambda H_t(b_t(\lambda)) + \lambda \cdot \frac{B}{T}$$

To update λ, we apply gradient descent in the dual space:

$$\lambda \leftarrow \lambda - \epsilon_t \cdot \frac{\partial}{\partial \lambda} \mathcal{L}(\lambda)$$

Since auction opportunities arrive in a streaming manner, we use a stochastic gradient descent approach based on each incoming auction:

$$\lambda \leftarrow \lambda - \epsilon_t \cdot \frac{\partial}{\partial \lambda} L_t(\lambda)$$

The derivative is computed as:

$$\frac{\partial}{\partial \lambda} L_t(\lambda) = b_t'(\lambda) \left[r_t \cdot g_t(b_t(\lambda)) - \lambda \cdot h_t(b_t(\lambda)) \right] + \frac{B}{T} - H_t(b_t(\lambda))$$

Recall that:

$$\frac{h_t(b_t)}{g_t(b_t)} = \frac{r_t}{\lambda}$$

implying:

$$r_t \cdot g_t(b_t(\lambda)) - \lambda \cdot h_t(b_t(\lambda)) = 0$$

Therefore:

$$\frac{\partial}{\partial \lambda} L_t(\lambda) = \frac{B}{T} - H_t(b_t)$$

The update rule becomes:

$$\lambda \leftarrow \lambda - \epsilon_t \cdot \left(\frac{B}{T} - H_t(b_t) \right) \tag{3.9}$$

In this update rule, $\frac{B}{T}$ represents the target spend per auction opportunity, while $H_t(b_t)$ denotes the actual spend per auction opportunity. The update step for λ adjusts according to the deviation between the actual spend and the target spend, mirroring the update rule derived in (5.1) for the maximum delivery problem under the standard second-price auction.

5.2 Monotonicity of the Ratio of PDF to CDF under Log-Concavity

We prove the following statement:

Let $F(x)$ be the CDF and $f(x)$ the PDF of a log-concave distribution. Then the ratio

$$R(x) = \frac{F(x)}{f(x)}$$

is non-decreasing.

Proof. We proceed in the following steps:
Step 1: Compute the derivative of $R(x)$. Define $R(x) = \frac{F(x)}{f(x)}$. By the quotient rule:

$$\begin{aligned} R'(x) &= \frac{d}{dx}\left(\frac{F(x)}{f(x)}\right) = \frac{F'(x)f(x) - F(x)f'(x)}{f(x)^2} \\ &= \frac{f(x)^2 - F(x)f'(x)}{f(x)^2} \quad \text{(since } F'(x) = f(x)\text{)} \\ &= 1 - \frac{F(x)f'(x)}{f(x)^2}. \end{aligned}$$

For $R(x)$ to be non-decreasing, we need to show:

$$f(x)^2 \geq F(x)f'(x) \quad \forall x. \tag{3.10}$$

Step 2: Log-Concavity and Its Consequence. Since $f(x)$ is log-concave, its logarithm $\log f(x)$ is concave, meaning:

$$\frac{d^2}{dx^2} \log f(x) \leq 0.$$

Define:

$$g(x) = \frac{f'(x)}{f(x)} = \frac{d}{dx} \log f(x).$$

Since $\log f(x)$ is concave, its derivative $g(x)$ is non-increasing, meaning:

$$g'(x) = \frac{d}{dx}\left(\frac{f'(x)}{f(x)}\right) \leq 0.$$

This key fact ensures that the relative change in $f(x)$ is decreasing.

Step 3: Define $h(x) = f(x) - F(x)g(x)$. To analyze the key inequality (3.10), define:

$$h(x) = f(x) - F(x)g(x).$$

Taking its derivative:

$$\begin{aligned} h'(x) &= f'(x) - [F'(x)g(x) + F(x)g'(x)] \\ &= f'(x) - [f(x)g(x) + F(x)g'(x)] \\ &= f'(x) - f'(x) - F(x)g'(x) \quad (\text{since } f'(x) = f(x)g(x)) \\ &= -F(x)g'(x). \end{aligned}$$

Since $g'(x) \leq 0$ (log-concavity implies $g(x)$ is non-increasing), we get:

$$h'(x) = -F(x)g'(x) \geq 0.$$

Thus, $h(x)$ is non-decreasing.

Step 4: Boundary Condition Analysis. To ensure $h(x) \geq 0$ for all x, consider the boundary conditions:

- If $x \to -\infty$, then $F(x) \to 0$ and $f(x) \to 0$, and it is known from Mill's ratio for log-concave distributions that:

$$\lim_{x\to-\infty} \frac{F(x)}{f(x)} = 0.$$

This implies that at the lower bound, $h(x) \geq 0$.

- If the distribution has bounded support, i.e., starting at $x = a$, then $F(a) = 0$ and thus:

$$h(a) = f(a) > 0.$$

Since $h(x)$ is non-decreasing, it follows that:

$$h(x) \geq 0 \quad \forall x.$$

Step 5: Final Conclusion. Since $h(x) = f(x) - F(x)g(x)$ is non-negative, this implies:

$$f(x)^2 \geq F(x)f'(x).$$

Thus:

$$R'(x) = \frac{f(x)^2 - F(x)f'(x)}{f(x)^2} \geq 0.$$

which proves that $R(x)$ is non-decreasing.

$$\frac{F(x)}{f(x)} \text{ is increasing under log-concavity.}$$

□

5.3 Log-concavity of Winning Probability Function under Sigmoid Assumption

We consider the winning probability function given by:

$$P_t(b) = \frac{1}{1 + e^{-(w_0 + \sum_{i=1}^k w_i x_i + \beta \log b)}}$$

which can be rewritten as:

$$P_t(b) = \frac{1}{1 + e^{-z}}, \quad \text{where} \quad z = w_0 + \sum_{i=1}^{k} w_i x_i + \beta \log b.$$

Our goal is to establish the log-concavity of $P_t(b)$, i.e., to show that:

$$\frac{d^2}{db^2} \log P_t(b) \leq 0.$$

Step 1: First Derivative of $P_t(b)$ Differentiating $P_t(b)$ with respect to b:

$$\frac{d}{db} P_t(b) = P_t(b)(1 - P_t(b)) \frac{d}{db} z.$$

Since:

$$\frac{d}{db} z = \frac{\beta}{b},$$

it follows that:

$$\frac{d}{db} P_t(b) = P_t(b)(1 - P_t(b)) \frac{\beta}{b}.$$

Step 2: Second Derivative of $P_t(b)$ Differentiating again:

$$\frac{d^2}{db^2} P_t(b) = \frac{d}{db} \left[P_t(b)(1 - P_t(b)) \frac{\beta}{b} \right].$$

Applying the product rule:

$$\frac{d^2}{db^2} P_t(b) = (1 - 2P_t(b)) P_t(b)(1 - P_t(b)) \frac{\beta^2}{b^2} + P_t(b)(1 - P_t(b)) \left(-\frac{\beta}{b^2} \right).$$

Factoring $P_t(b)(1 - P_t(b)) \frac{1}{b^2}$:

$$\frac{d^2}{db^2} P_t(b) = P_t(b)(1 - P_t(b)) \frac{1}{b^2} \left[(1 - 2P_t(b))\beta^2 - \beta \right].$$

Step 3: Second Derivative of $\log P_t(b)$ By the identity:

$$\frac{d^2}{db^2}\log P_t(b) = \frac{P_t''(b)}{P_t(b)} - \left(\frac{P_t'(b)}{P_t(b)}\right)^2,$$

we substitute $P_t''(b)$ and $P_t'(b)$ to obtain:

$$\frac{d^2}{db^2}\log P_t(b) = \frac{1-P_t(b)}{b^2}\left[(1-2P_t(b))\beta^2 - \beta - (1-P_t(b))\beta^2\right].$$

Simplifying:

$$\frac{d^2}{db^2}\log P_t(b) = \frac{1-P_t(b)}{b^2}\left[\beta^2 - 2P_t(b)\beta^2 - \beta - \beta^2 + P_t(b)\beta^2\right].$$

$$= \frac{1-P_t(b)}{b^2}\left[-P_t(b)\beta^2 - \beta\right].$$

Step 4: Sign Analysis and Conclusion Since $1-P_t(b)$ is positive and b^2 is positive, the sign of $\frac{d^2}{db^2}\log P_t(b)$ depends on:

$$-P_t(b)\beta^2 - \beta.$$

- If $\beta > 0$, then $-P_t(b)\beta^2 - \beta$ is negative for all $P_t(b)$, ensuring $\frac{d^2}{db^2}\log P_t(b) \le 0$.
- If $\beta < 0$, the expression could be positive, violating log-concavity.

Thus, we conclude that

$$P_t(b) \text{ is log-concave if and only if } \beta > 0.$$

CHAPTER 4

MULTI-CHANNEL DELIVERY

We discuss how bidding algorithms should be designed when a campaign can be delivered across multiple channels with different auction mechanisms. This is particularly important for platforms that extend their ad delivery to audience networks, external SSPs, and ad exchanges.

A number of digital advertising platforms (e.g., Meta and LinkedIn) offer advertisers the option to extend their sponsored content (campaigns) to third-party, offsite platforms, thereby enabling campaigns to reach a broader audience. One key motivation for this extension is that these platforms possess vast amounts of first-party, onsite data, which can be leveraged to more effectively target the desired audience and predict conversion rates compared to traditional Demand-Side Platforms (DSPs). By extending delivery to offsite traffic, the platform can increase its available supply while continuing to capitalize on the value of its onsite data.

In such cases, campaigns can be delivered across multiple placements. For onsite placements, the auction mechanism might use a second-price or generalized second-price auction, whereas for offsite placements (e.g., ad exchanges or Supply-Side Platforms (SSPs)), the auction is more likely to follow a first-price auction model (as discussed in the previous chapter). When designing a bid optimization strategy, we need to take these hybrid auction mechanisms into consideration.

1 Multi-Channel Max Delivery Problem

We use a max-delivery example to illustrate how to manage multi-channel delivery under hybrid auction mechanisms. As before, we aim to maximize total conversions under a given budget constraint. Suppose there are M channels, with the first one representing the onsite platform and the remaining $M-1$ channels representing offsite platforms. On the onsite platform, ad slots are auctioned using a standard second-price mechanism, and all auction data are fully available. In contrast, on offsite platforms, ad slots are auctioned using a first-price mechanism with only partial information visibility (i.e., the data are censored; the campaign only learns whether it won the auction or not). With these assumptions, the problem can be formulated as follows:

$$\begin{aligned} \max_{b_{i,t}\geq 0} \quad & \sum_{i=1}^{M}\sum_{t=1}^{T_i} r_{i,t}\cdot G_{i,t}(b_{i,t}) \\ \text{s.t.} \quad & \sum_{i=1}^{M}\sum_{t=1}^{T_i} H_{i,t}(b_{i,t}) \leq B. \end{aligned} \tag{4.1}$$

where:

- B is the total campaign budget.
- T_i is the number of auction opportunities for channel i.
- $i = 1, \cdots, M$ indicates different channels (e.g., $i = 1$ represents traffic from onsite sources, $i = 2$ represents traffic from an ad exchange, etc.).
- $t = 1, \cdots, T_i$ indicates the t-th auction opportunity from channel i.
- $r_{i,t}$ is the estimated conversion rate for the t-th auction opportunity from channel i.
- $b_{i,t}$ is the bid amount per impression.

- $G_{i,t}(b_{i,t})$ is the winning probability (or a binary indicator) given the bid price $b_{i,t}$.
- $H_{i,t}(b_{i,t})$ is the expected payment (or cost) given the bid price $b_{i,t}$.

We apply the standard primal-dual method to derive the bidding formula. First, we introduce the Lagrangian with dual parameter λ:

$$\begin{aligned}\mathcal{L}(b_{i,t},\lambda) &= \sum_{i=1}^{M}\sum_{t=1}^{T_i} r_{i,t}\cdot G_{i,t}(b_{i,t}) - \lambda\left(\sum_{i=1}^{M}\sum_{t=1}^{T_i} H_{i,t}(b_{i,t}) - B\right)\\ &= \sum_{i=1}^{M}\underbrace{\sum_{t=1}^{T_i}\left[r_{i,t}\cdot G_{i,t}(b_{i,t}) - \lambda H_{i,t}(b_{i,t})\right]}_{S_i(b_{i,t})} + \lambda B.\end{aligned}$$

The dual problem is then obtained by maximizing over $b_{i,t}$:

$$\mathcal{L}(\lambda) = \max_{b_{i,t}\ge 0}\mathcal{L}(b_{i,t},\lambda).$$

We handle this maximization problem separately for different channels:

- **Onsite traffic** $i = 1$: In this case, the auction follows a standard second-price mechanism. We have

$$G_{1,t}(b_{1,t}) = \mathbb{1}_{\{b_{1,t}>c_t\}} \quad \text{and} \quad H_{1,t}(b_{1,t}) = \mathbb{1}_{\{b_{1,t}>c_t\}}\cdot c_t,$$

where c_t is the supporting price (the highest bid among other bidders) for the t-th auction opportunity. Therefore, we get:

$$\begin{aligned}S_1(b_{1,t}) &= \sum_{t=1}^{T_1}\left[r_{1,t}\cdot G_{1,t}(b_{1,t}) - \lambda\cdot H_{1,t}(b_{1,t})\right]\\ &= \sum_{t=1}^{T_1}\left[r_{1,t}\cdot\mathbb{1}_{\{b_{1,t}>c_t\}} - \lambda\cdot\mathbb{1}_{\{b_{1,t}>c_t\}}\cdot c_t\right]\\ &= \sum_{t=1}^{T_1}\mathbb{1}_{\{b_{1,t}>c_t\}}\left(r_{1,t} - \lambda\cdot c_t\right).\end{aligned}$$

To maximize the sum S_1 over $b_{1,t}$, as we do for the max-delivery case in Part II section 1, we choose $b_{1,t} > c_t$ if $r_{1,t} > \lambda\cdot c_t$ and $b_{1,t}\le c_t$ otherwise. The maximal value is then:

$$\max_{b_{1,t}\ge 0} S_1(b_{1,t}) = \sum_{t=1}^{T_1}\left(r_{1,t} - \lambda\cdot c_t\right)_+. \tag{4.2}$$

For a fixed dual parameter λ, the optimal bid for onsite traffic is given by:

$$b^*_{1,t} = \frac{r_{1,t}}{\lambda}. \tag{4.3}$$

This bidding formula is identical to the one derived in Part II section 1.

- **Offsite traffic** $2 \le i \le M$: In this case, the auction follows a first-price mechanism. Assume that the marginal profit percentile m is collected for these offsite auctions. We then have:

$$G_{i,t}(b_{i,t}) = P_{i,t}(b_{i,t}) \quad \text{and} \quad H_{i,t}(b_{i,t}) = P_{i,t}(b_{i,t}) \cdot b_{i,t} \cdot (1+m),$$

where $P_{i,t}(b_{i,t})$ denotes the winning probability for the t-th auction opportunity from the i-th offsite channel given bid $b_{i,t}$. For the i-th channel, we have:

$$\begin{aligned} S_i(b_{i,t}) &= \sum_{t=1}^{T_i} \left[r_{i,t} \cdot G_{i,t}(b_{i,t}) - \lambda \cdot H_{i,t}(b_{i,t}) \right] \\ &= \sum_{t=1}^{T_i} \left[r_{i,t} \cdot P_{i,t}(b_{i,t}) - \lambda \cdot P_{i,t}(b_{i,t}) \cdot b_{i,t} \cdot (1+m) \right]. \end{aligned}$$

Similar to the derivation in (3.6), for a fixed dual parameter λ, S_i attains its maximum when $b_{i,t} = b^*_{i,t}$ such that:

$$b^*_{i,t} + \frac{P_{i,t}(b^*_{i,t})}{P'_{i,t}(b^*_{i,t})} = \frac{r_{i,t}}{\lambda(1+m)}. \tag{4.4}$$

The maximal value of S_i over $b_{i,t}$ is then:

$$\max_{b_{i,t}\ge 0} S_i(b_{i,t}) = \sum_{t=1}^{T_i} \left[r_{i,t} \cdot G_{i,t}(b^*_{i,t}(\lambda)) - \lambda \cdot H_{i,t}(b^*_{i,t}(\lambda)) \right], \tag{4.5}$$

where $b^*_{i,t}(\lambda)$ is the solution of $b^*_{i,t}$ derived from equation (4.4).

Combining (4.3) and (4.4), we have:

$$\begin{aligned} \mathcal{L}(\lambda) &= \max_{b_{i,t}\ge 0} \mathcal{L}(b_{i,t}, \lambda) \\ &= \max_{b_{1,t}\ge 0} S_1(b_{1,t}) + \sum_{i=2}^{M} \max_{b_{i,t}\ge 0} S_i(b_{i,t}) + \lambda B \\ &= \sum_{t=1}^{T_1} (r_{1,t} - \lambda \cdot c_t)_+ + \sum_{i=2}^{M} \sum_{t=1}^{T_i} \left[r_{i,t} \cdot G_{i,t}(b^*_{i,t}(\lambda)) - \lambda \cdot H_{i,t}(b^*_{i,t}(\lambda)) \right] + \lambda B \\ &= \sum_{t=1}^{T_1} \left[\underbrace{(r_{1,t} - \lambda \cdot c_t)_+ + \lambda \cdot \frac{B}{T}}_{L_{1,t}(\lambda)} \right] \\ &\quad + \sum_{i=2}^{M} \sum_{t=1}^{T_i} \left[\underbrace{r_{i,t} \cdot G_{i,t}(b^*_{i,t}(\lambda)) - \lambda \cdot H_{i,t}(b^*_{i,t}(\lambda)) + \lambda \cdot \frac{B}{T}}_{L_{i,t}(\lambda)} \right], \end{aligned} \tag{4.6}$$

where $T = \sum_{i=1}^{M} T_i$ is the total traffic. To find the optimal λ, we apply the dual gradient descent method:

$$\lambda \leftarrow \lambda - \epsilon \cdot \frac{\partial}{\partial \lambda}\mathcal{L}(\lambda).$$

For streaming data, we use stochastic gradient descent, and the update rule for λ is:

$$\lambda \leftarrow \lambda - \epsilon \cdot \frac{\partial}{\partial \lambda}L_{i,t}(\lambda).$$

There are two cases:

- **For onsite traffic** ($i = 1$):

  $$L_{1,t}(\lambda) = (r_{1,t} - \lambda \cdot c_t)_+ + \lambda \cdot \frac{B}{T},$$

  so we have:

  $$\frac{\partial}{\partial \lambda}L_{1,t}(\lambda) = \frac{B}{T} - \mathbb{1}_{\{r_{1,t} > \lambda \cdot c_t\}} \cdot c_t.$$

- **For offsite traffic** ($i = 2, \cdots, M$):

  $$L_{i,t}(\lambda) = r_{i,t} \cdot G_{i,t}(b^*_{i,t}(\lambda)) - \lambda \cdot H_{i,t}(b^*_{i,t}(\lambda)) + \lambda \cdot \frac{B}{T}.$$

  From the discussion in Part III in subsection 5.1, we know that:

  $$\frac{\partial}{\partial \lambda}L_{i,t}(\lambda) = \frac{B}{T} - H_{i,t}(b_{i,t}(\lambda)).$$

Note that both $\mathbb{1}_{\{r_{1,t} > \lambda \cdot c_t\}} \cdot c_t$ and $H_{i,t}(b_{i,t}(\lambda))$ represent the expected cost for the corresponding auction opportunity. Additionally, $\frac{B}{T}$ denotes the target spend per auction opportunity for this campaign. From this, we can conclude that regardless of the platform type (onsite or offsite), the gradient $\frac{\partial}{\partial \lambda}L_{i,t}(\lambda)$ indicates the deviation of the actual spend from the target spend.

The algorithm to solve the multi-channel max-delivery problem (4.1) is now clear:

- Initialize the dual parameter λ.
- Submit bids and observe the actual cost for each auction opportunity.
- Update λ by applying gradient descent, where the gradient is the difference between the target spend $\frac{B}{T}$ and the actual spend.
- Update the bid using either (4.3) or (4.4), depending on whether the traffic is onsite or offsite.

Essentially, for the hybrid auction mechanisms in the multi-channel problem, λ is the parameter responsible for the overall delivery control (including both onsite and offsite traffic). Once λ is updated, the method for determining the bid level follows the same principles as discussed previously, with the specific bid calculation depending on whether the traffic originates from an onsite (second-price) or offsite (first-price) channel.

As we mentioned before, in practice, we may implement the algorithm in a batch manner, the main idea of this approach can be summarized in Algorithm 28:

Algorithm 28 DOGD-Based Bid Strategy for Multi-Channel Max Delivery Optimization

```
Require: $B$: Total campaign budget; $T$: Total auction opportunities; $\lambda_0$: Initial value
    for the dual parameter; $\epsilon$: Learning rate; $m$: Marginal profit percentile; $P_{i,t}(b)$, $P'_{i,t}(b)$:
    Winning probability function and its derivative.
Ensure: $b^*_{i,t}$: Optimal bid at auction opportunity $t$.
 1: Step 1: Initialize Parameters
        • Initialize dual parameter: $\lambda \leftarrow \lambda_0$
        • Define target spend: $\frac{B}{T}$
 2: Step 2: Solve for Optimal $b^*_{i,t}$
 3: for each pacing interval do
 4:     Observe actual spend $S$ and update $\lambda$ using gradient descent:
            $$\lambda \leftarrow \lambda - \epsilon \left( \frac{B}{T} - S \right)$$
 5:     for each auction opportunity $t$ do
 6:         if traffic is onsite ($i = 1$) then
 7:             Compute bid:
                    $$b^*_{1,t} = \frac{r_{1,t}}{\lambda}$$
 8:         else
 9:             Perform Binary Search to Solve for $b^*_{i,t}$
10:             Initialize search range: $b_{\text{low}} \leftarrow b_l, b_{\text{high}} \leftarrow b_u$
11:             while $b_{\text{high}} - b_{\text{low}} > \epsilon$ do
12:                 Set $b_{\text{mid}} \leftarrow \frac{b_{\text{low}} + b_{\text{high}}}{2}$
13:                 Compute:
        $$F(b_{\text{mid}}) = P'_{i,t}(b_{\text{mid}}) - \lambda \cdot (1+m) \cdot b_{\text{mid}} P'_{i,t}(b_{\text{mid}}) - \lambda \cdot (1+m) \cdot P_{i,t}(b_{\text{mid}})$$
14:                 if $F(b_{\text{mid}}) = 0$ then
15:                     $b^*_{i,t} \leftarrow b_{\text{mid}}$
16:                     break
17:                 else if $F(b_{\text{mid}}) > 0$ then
18:                     $b_{\text{low}} \leftarrow b_{\text{mid}}$
19:                 else
20:                     $b_{\text{high}} \leftarrow b_{\text{mid}}$
21:                 end if
22:             end while
23:         end if
24:     end for
25: end for
26: Step 3: Return Optimal Bids
        • Return $b^*_{i,t}$.
```

CHAPTER 5

CAMPAIGN GROUP OPTIMIZATION

In this chapter, we introduce the problem of campaign group optimization, where multiple campaigns share a common budget. We present a mathematical formulation of this problem and explore principled approaches for bid optimization, including control-based methods and learning-based strategies, to adaptively adjust bids in response to changing market conditions. Additionally, we discuss key practical considerations to enhance the robustness of the bidding algorithm in real-world applications.

A number of platforms provide advertisers with the option to create a campaign group consisting of multiple campaigns with the same objective, sharing a total budget. Some platforms even use this as the default configuration.

From the advertiser's perspective, if the delivery algorithm is well-designed, the budget will be automatically allocated across different campaigns to achieve optimal performance. This simplifies the campaign creation process, as advertisers no longer need to manually split the budget across campaigns based on estimations.

From the advertising platform's perspective, allowing budget sharing across different campaigns provides greater flexibility in designing delivery algorithms. This enables the platform to allocate budget more intelligently, optimizing overall performance while mitigating the risk of under-delivery when certain campaigns have lower quality (i.e., lower conversion rates, leading to less competitive bids).

1 Max Delivery Problem for Campaign Group Optimization

We first discuss the max-delivery problem. The basic setting is very similar to the max-delivery problem for a single campaign. When advertisers create a campaign group, they specify the lifetime and input the budget. The only difference is that multiple campaigns with different creatives are created under this group, and these campaigns share the group budget.

Problem Formulation Suppose there are N campaigns within the campaign group sharing a total budget B. We aim to maximize the total conversions of all campaigns within the given budget. Under a standard second-price auction, the problem can be formulated as:

$$\begin{aligned} \max_{x_{i,t}\in\{0,1\}} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot r_{i,t} \\ \text{s.t.} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t} \le B. \end{aligned} \tag{5.1}$$

where $x_{i,t} = \mathbb{1}_{\{b_{i,t}>c_{i,t}\}}$, and:

- B is the total campaign group budget.
- T_i is the number of eligible auction opportunities for campaign i.
- $x_{i,t}$ is the binary decision variable indicating whether campaign i wins its t-th auction opportunity.
- $r_{i,t}$ is the estimated conversion rate for the t-th auction opportunity for campaign i.
- $c_{i,t}$ is the highest competing bid per impression for the t-th auction opportunity for campaign i.
- $b_{i,t}$ is the bid amount per impression for the t-th auction opportunity for campaign i.

Derivation of the Optimal Bid As always, we introduce the Lagrangian with the dual multiplier $\lambda \geq 0$:

$$\begin{aligned}\mathcal{L}(x_{i,t}, \lambda) &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t} \cdot r_{i,t} - \lambda \cdot \left(\sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t} \cdot c_{i,t} - B\right) \\ &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} (x_{i,t} \cdot r_{i,t} - \lambda \cdot x_{i,t} \cdot c_{i,t}) + \lambda \cdot B \\ &= \sum_{i=1}^{N}\sum_{t=1}^{T_i} \left[x_{i,t} \cdot (r_{i,t} - \lambda \cdot c_{i,t}) + \lambda \cdot \frac{B}{T}\right]\end{aligned}$$

where $T = \sum T_i$ is the total traffic of the campaign group. As reasoned in Part III section 1, to maximize $\mathcal{L}(x_{i,t}, \lambda)$, we set $x_{i,t} = 1$ whenever $r_{i,t} - \lambda c_{i,t} > 0$ and $x_{i,t} = 0$ otherwise. Then:

$$\mathcal{L}^*(\lambda) = \max_{x_{i,t} \in \{0,1\}} \mathcal{L}(x_{i,t}, \lambda) = \sum_{i=1}^{N}\sum_{t=1}^{T_i} \left[\underbrace{(r_{i,t} - \lambda \cdot c_{i,t})_+ + \lambda \cdot \frac{B}{T}}_{f_{i,t}(\lambda)}\right].$$

where $(\cdot)_+$ is the ReLU function. The dual problem is:

$$\min_{\lambda \geq 0} \mathcal{L}^*(\lambda) = \min_{\lambda \geq 0} \sum_{i=1}^{N}\sum_{t=1}^{T_i} \left[(r_{i,t} - \lambda \cdot c_{i,t})_+ + \lambda \cdot \frac{B}{T}\right].$$

Assuming the problem is feasible, we find the optimal dual parameter:

$$\lambda^* = \arg\min_{\lambda \geq 0} \mathcal{L}^*(\lambda).$$

By the same argument as for the single campaign max-delivery problem, we see that $\lambda^* > 0$ is the value that just depletes the budget such that:

$$\sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t} \cdot c_{i,t} = B.$$

The corresponding optimal bid per impression is:

$$b_{i,t}^* = \frac{r_{i,t}}{\lambda^*}.$$

The optimal bid per click (objective) is given by the following constant bid:

$$b_{\text{click}}^* = \frac{1}{\lambda^*}.$$

Determining $b^*_{\textbf{click}}$ Most of the algorithms discussed in Part II can be applied to determine the optimal bid level. We briefly discuss PID control and DOGD methods:

- **PID Controller:** λ^* is the value of λ that depletes the total budget B. Based on overall traffic patterns across all campaigns, we construct a target spend per pacing interval for the campaign group. The actual spend within each pacing interval is collected and compared to the target as the error signal $e(t)$. A PID controller, as in Algorithm 3, is applied to update b_{click} adaptively. The bid per impression for campaign i is then computed as:
$$b_{i,t} = b_{\text{click}} \cdot r_{i,t}.$$
- **DOGD:** We update λ in dual space using DOGD. The update rule is:
$$\lambda \leftarrow \lambda - \epsilon \cdot \frac{\partial}{\partial \lambda} \mathcal{L}^*(\lambda).$$
For streaming data, we apply stochastic gradient descent to $f_{i,t}$:
$$\lambda \leftarrow \lambda - \epsilon \cdot \left(\frac{B}{T} - \mathbb{1}_{\{r_{i,t} - \lambda \cdot c_{i,t} > 0\}} \cdot c_{i,t} \right).$$
Note that $\frac{B}{T} - \mathbb{1}_{\{r_{i,t} - \lambda \cdot c_{i,t}\}} \cdot c_{i,t}$ represents the gap between the target spend and the actual spend for each auction request. The update rule we derived here, to some extent, is identical to the update rule derived in Algorithm 15. This similarity suggests that we can implement either the standard stochastic gradient descent (SGD) or a mini-batch version of the DOGD algorithm to update the bids.

One important consideration is that by applying these methods, we implicitly assume that all campaigns within the group share the same distributions for predicted click-through rate (pCTR) and the second-highest price. However, this assumption does not always hold in practice, as campaign quality can vary significantly. Some high-quality campaigns within the group may have substantially higher pCTR compared to others.

We will discuss how to handle this heterogeneous distribution later in this chapter. For now, we summarize the implementation of the PID controller and Dual Online Gradient Descent (DOGD) in Algorithm 29 and Algorithm 30, respectively.

Interpretation of the Bidding Formula If we examine the optimal bid formula:
$$b^*_{i,t} = \frac{r_{i,t}}{\lambda^*}$$
and note that λ^* is updated based on the overall delivery of the campaign group, we can see that this formula is identical to the one used when treating the campaign group holistically. The only difference is that, for each request, the conversion rate is determined by the quality of the creative associated with the specific campaign within the group.

Since the denominator (λ^*) is the same across all campaigns, we can expect that the final budget allocation for each campaign will be roughly proportional to its average conversion rate. This is because the average bid per impression for each campaign is determined

Algorithm 29 PID Controller for Max Delivery in Campaign Group Optimization

```
Require:
 1: $\Delta t$: Time interval for bid updates (e.g., 1 minute), $\Delta\tau$: Time interval of target budget
    per bucket
 2: $B$: Total campaign group budget
 3: $\{s_k\}$: Target spend for each pacing interval $k$ for the campaign group
 4: $K_p, K_i, K_d$: PID controller gains
 5: $N$: Number of campaigns within the campaign group
 6: $M$: Number of bid updates in a day ($M = \text{MinutesInOneDay}/\Delta t$)
Ensure: $b^*_{i,t}$: Optimal bid for each campaign $i$ and auction opportunity $t$
 7: Initialize $u(t_0) \leftarrow 0$, cumulative_error $\leftarrow 0$, previous_error $\leftarrow 0$, $b_{click} \leftarrow$ initBidPerClick
 8: for $k = 1$ to $M$ do                                          ▷ Loop over pacing intervals
 9:     Measure observed total spend across all campaigns during $k$-th pacing interval:
            $$r_k \leftarrow \sum_{i=1}^{N} \text{observed\_spend}_i(k)$$
10:     Compute the error factor:
            $$e_k \leftarrow r_k - s_k$$
11:     Update the control signal using the PID formula:
            $$u_k \leftarrow K_p \cdot e_k + K_i \cdot \text{cumulative\_error} + K_d \cdot \frac{e_k - \text{previous\_error}}{\Delta t}$$
12:     Update the cumulative error:
            $$\text{cumulative\_error} \leftarrow \text{cumulative\_error} + e_k \cdot \Delta t$$
13:     Update the bid per click price:
            $$b_{click} \leftarrow b_{clikc} \cdot \exp(u(k))$$
14:     for each campaign $i = 1, \ldots, N$ do
15:         for each auction opportunity $t$ in campaign $i$ do
16:             Get conversion rate $r_{i,t}$ and submit the new bid:
                    $$b_{i,t} = b_{click} \cdot r_{i,t}$$
17:         end for
18:     end for
19:     Update the previous error:
            $$\text{previous\_error} \leftarrow e_k$$
20: end for
```

Algorithm 30 DOGD for Max Delivery in Campaign Group Optimization

```
Require:
 1: $B$: Total budget, $T$: Predicted total number of auction opportunities
 2: $\Delta t$: Mini-batch update interval, $\epsilon_t$: Step size
 3: $N$: Number of campaigns within the campaign group
 4: $M$: Number of bid updates in a day ($M = \text{MinutesInOneDay}/\Delta t$)
Ensure: Optimal bid for each campaign $i$ and auction opportunity $t$
 5: Initialize $\lambda \leftarrow \lambda_{\text{init}}$                ▷ Initial dual variable
 6: for $k = 1$ to $M$ do                              ▷ Loop over pacing intervals
 7:     Count the number of total auction requests $R_k$ across all campaigns
 8:     Observe the actual spend $S_k$ during the pacing interval $\mathcal{I}_k$
 9:     Compute the mini-batch gradient:
            $$\text{BatchGrad}_k = \sum_{s \in \mathcal{I}_k} \nabla_\lambda f_{i,t}(\lambda) = \frac{R_k}{T} \cdot B - S_k$$
10:     Update the dual variable using the mini-batch gradient:
            $$\lambda \leftarrow \lambda - \epsilon \cdot \text{BatchGrad}_k$$
11:     Compute the bid per click:
            $$b_{click} = \frac{1}{\lambda}$$
12:     for each campaign $i = 1, \ldots, N$ do
13:         for each auction opportunity $t \in \mathcal{I}_k$ in campaign $i$ do
14:             Get conversion rate $r_{i,t}$ and submit the new bid:
                    $$b_{i,t} = b_{click} \cdot r_{i,t}$$
15:         end for
16:     end for
17: end for
```

solely by its conversion rate. As a result, campaigns with higher-quality creatives (i.e., those with higher conversion rates) will consume a larger share of the budget and achieve more conversions.

Thus, the bidding algorithm inherently allocates more budget to higher-quality campaigns. From another perspective, this validates the motivation behind the campaign group product, as it automatically selects and prioritizes better-performing ads for delivery to the target audience.

2 Cost Cap Problem for Campaign Group Optimization

We now discuss the cost cap problem for campaign group optimization. The only difference between cost cap and max delivery is that, for each campaign, there is an additional cost control constraint. Specifically, at the end of the campaign's lifetime, the average cost per result should not exceed a pre-specified cap.

Problem Formulation Suppose there are N campaigns within the campaign group sharing a total budget B. For each campaign $i = 1, \ldots, N$, the advertiser imposes a cap C_i on the average cost per conversion. Under a standard second-price auction, the problem can be formulated as:

$$\begin{aligned}
\max_{x_{i,t} \in \{0,1\}} \quad & \sum_{i=1}^{N} \sum_{t=1}^{T_i} x_{i,t} \cdot r_{i,t} \\
\text{s.t.} \quad & \sum_{i=1}^{N} \sum_{t=1}^{T_i} x_{i,t} \cdot c_{i,t} \le B, \\
& \frac{\sum_{t=1}^{T_i} x_{i,t} \cdot c_{i,t}}{\sum_{t=1}^{T_i} x_{i,t} \cdot r_{i,t}} \le C_i, \quad i = 1, \ldots, N.
\end{aligned} \tag{5.2}$$

where:

- B is the total campaign group budget.
- T_i is the number of eligible auction opportunities for campaign i.
- $x_{i,t}$ is the binary decision variable indicating whether campaign i wins its t-th auction opportunity.
- $r_{i,t}$ is the estimated conversion rate for the t-th auction opportunity for campaign i.
- $c_{i,t}$ is the highest competing bid per impression for the t-th auction opportunity for campaign i.
- $b_{i,t}$ is the bid amount per impression for the t-th auction opportunity for campaign i.
- C_i is the cost cap for campaign i.

Derivation of the Optimal Bid We use the primal-dual method. First, we write the Lagrangian:

$$\begin{aligned}\mathcal{L}(x_{i,t},\lambda,\mu_i) =& \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot r_{i,t} - \lambda\left(\sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t} - B\right)\\ &- \sum_{i=1}^{N}\mu_i\cdot\left[\sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t} - C_i\cdot\left(\sum_{t=1}^{T_i} x_{i,t}\cdot r_{i,t}\right)\right]\\ =& \sum_{i=1}^{N}\sum_{t=1}^{T_i}\left(x_{i,t}\cdot r_{i,t} - \lambda\cdot x_{i,t}\cdot c_{i,t} + \lambda\cdot\frac{B}{T} - \mu_i\cdot x_{i,t}\cdot c_{i,t} + \mu_i\cdot C_i\cdot x_{i,t}\cdot r_{i,t}\right)\\ =& \sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[x_{i,t}\cdot(r_{i,t} - \lambda\cdot c_{i,t} - \mu_i\cdot c_{i,t} + \mu_i\cdot C_i\cdot r_{i,t}) + \lambda\cdot\frac{B}{T}\right]\end{aligned}$$

Similar to the max delivery problem, to maximize $\mathcal{L}(x_{i,t},\lambda,\mu_i)$, we set $x_{i,t}=1$ whenever:

$$r_{i,t} - \lambda\cdot c_{i,t} - \mu_i\cdot c_{i,t} + \mu_i\cdot C_i\cdot r_{i,t} > 0,$$

and $x_{i,t}=0$ otherwise. The dual objective $\mathcal{L}^*(\lambda,\mu_i)$ then becomes:

$$\mathcal{L}^*(\lambda,\mu_i) = \sum_{i=1}^{N}\sum_{t=1}^{T_i}\left[(r_{i,t} - \lambda\cdot c_{i,t} - \mu_i\cdot c_{i,t} + \mu_i\cdot C_i\cdot r_{i,t})_+ + \lambda\cdot\frac{B}{T}\right].$$

Suppose we find a feasible solution to the dual:

$$\lambda^*,\mu_i^* = \underset{\lambda\geq 0,\mu_i\geq 0}{\arg\min}\,\mathcal{L}^*(\lambda,\mu_i).$$

The optimal bid per impression is then given by:

$$b_{i,t}^* = \frac{1+\mu_i^* C_i}{\lambda^*+\mu_i^*}\cdot r_{i,t}.$$

Determining $b_{i,t}^*$ There are various approaches to updating bids. In this section, we use Model Predictive Control (MPC) as an example to solve the problem. Other methods, such as PID control and Dual Online Gradient Descent (DOGD), are also applicable. Readers are encouraged to explore and implement these alternative algorithms themselves.

For simplicity, we assume that all campaigns within the group follow the same distribution for predicted click-through rate (pCTR) and the second-highest price. As discussed earlier, at a specific time τ, the MPC method formulates a new optimization problem to solve over a receding time horizon H.

For each campaign i, we leverage prediction models to estimate the remaining budget and the number of remaining auction opportunities. These estimates are then used to compute the target spend and the target cost per result for each campaign. Once these targets are established, bid landscape models—such as bid-to-spend, bid-to-number-of-conversions, and bid-to-cost-per-conversion (CPX)—can be utilized to evaluate each bid candidate and determine whether it satisfies the constraints within the receding horizon. The optimal bid is then selected as the new update.

Interpretation of the Bidding Formula The optimal bid formula above is essentially identical to the one derived for the single-campaign cost cap problem. Here, λ is the dual parameter responsible for overall delivery control, while μ_i is the dual parameter responsible for cost control specific to campaign i.

- If the cost constraint is not active, then $\mu_i^* = 0$, and the bidding formula simplifies to:

$$b_{i,t}^* = \frac{r_{i,t}}{\lambda^*},$$

  which corresponds to the max delivery problem.

- If the budget constraint is not active, then $\lambda^* = 0$, and the bidding formula reduces to:

$$b_{i,t}^* = \left(\frac{1}{\mu^*} + C_i\right) \cdot r_{i,t},$$

  which only incorporates the cost cap constraint.

This formulation allows for a flexible allocation of budget while ensuring that each campaign meets its cost cap constraints.

We summarize the above discussion in the following algorithm:

3 Remarks

3.1 Minimum Delivery Constraints in Campaign Group Optimization

Some advertisers wish to ensure that each campaign in the group receives a certain amount of impressions to prevent the entire budget from being spent on a single campaign. To achieve this, they may specify a minimum portion of the total budget that must be allocated to each campaign.

In this case, the optimization problem can be formulated as follows:

Algorithm 31 MPC for Cost Cap in Campaign Group Optimization

```
Require:
 1: $B_{\tau,r}$: Remaining budget; $B$: Total budget; $C_i$: Cost cap for campaign $i$; $N$: number of
    campaigns
 2: $T_i$: Total predicted requests for campaign $i$
 3: $H$: Receding time horizon; $NC_{\tau,i}$: Observed conversions for campaign $i$; $TR_{\tau,i}$: Predicted
    total remaining requests for campaign $i$;
 4: $NR_{\tau,H,i}$: Predicted requests in $(\tau, \tau + H)$ for campaign $i$; $[b_l, b_u]$: Bid bounds; $\Delta b$: Search
    step size.
Ensure: $b_i^*$: Optimal bid for $(\tau, \tau + H)$ for campaign $i$.
 5: Step 1: Compute Budget and Cost Cap Constraints
 6: Compute the budget allocation for each campaign $i$:
        $$B_i \leftarrow \frac{T_i}{\sum_{j=1}^{N} T_j} \cdot B$$
 7: Compute the budget allocation for the receding horizon for each campaign $i$:
        $$B_{\tau,H,i} \leftarrow \frac{NR_{\tau,H,i}}{\sum_{i=1}^{N} TR_{\tau,i}} \cdot B_{\tau,r}$$
 8: Compute the cost per result upper bound for the remaining time:
        $$C_{\tau,r,i} \leftarrow \frac{B_{\tau,r,i}}{\frac{B_i}{C_i} - NC_\tau}$$
 9: Step 2: Construct Models $f_i(b)$ and $g_i(b)$
10: Collect the most recent $N$ bid-spend pairs $\{b_k, s_k\}$ and apply LIS or PAVA to construct
    $f_i(b)$ normalized to $H$.
11: Collect the most recent $N$ bid-conversion pairs $\{b_k, n_k\}$ and apply LIS or PAVA to con-
    struct $g_i(b)$ normalized to $H$.
12: Compute $h_i(b)$ as:
        $$h_i(b) \leftarrow \frac{f_i(b)}{g_i(b)}$$
13: Step 3: Search for Optimal Bid $b^*$
14: Initialize $b^* \leftarrow b_l$.
15: for $b$ from $b_l$ to $b_u$ with step size $\Delta b$ do
16:     if $\sum_{i=1}^{N} f_i(b) \leq B_{\tau,H}$ and $h_i(b) \leq C_{\tau,r,i}$ then
17:         Update $b^* \leftarrow b$.
18:     end if
19: end for
20: Step 4: Return Optimal Bid
21: return $b^*$
```

$$\begin{aligned}
\max_{x_{i,t}\in\{0,1\}} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot r_{i,t} \\
\text{s.t.} \quad & \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t} \le B, \\
& \sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t} \ge s_i \cdot B, \quad i = 1,\dots,N, \\
& \sum_{i=1}^{N} s_i \le 1.
\end{aligned} \tag{5.3}$$

where:

- B is the total campaign group budget.
- T_i is the number of eligible auction opportunities for campaign i.
- $x_{i,t}$ is the binary decision variable indicating whether campaign i wins its t-th auction opportunity.
- $r_{i,t}$ is the estimated conversion rate for the t-th auction opportunity for campaign i.
- $c_{i,t}$ is the highest competing bid per impression for the t-th auction opportunity for campaign i.
- $b_{i,t}$ is the bid amount per impression for the t-th auction opportunity for campaign i.
- s_i is the minimum fraction of the total budget that must be allocated to campaign i.

The additional constraint $\sum_{i=1}^{N} s_i \le 1$ ensures that the budget allocation remains feasible, preventing an over-constrained system where the sum of required allocations exceeds the available budget.

For completeness, we present a bid update algorithm using the Dual Online Gradient Descent (DOGD) method. Readers interested in alternative approaches may also explore control-based methods, such as Model Predictive Control (MPC).

Lagrangian Formulation To solve this problem, we introduce dual multipliers:

- $\lambda \ge 0$ for the overall budget constraint.
- $\gamma_i \ge 0$ for the minimum budget constraint of each campaign.

The Lagrangian function is:

$$\begin{aligned}
\mathcal{L}(x_{i,t},\lambda,\gamma_i) = & \sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot r_{i,t} - \lambda\left(\sum_{i=1}^{N}\sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t} - B\right) \\
& - \sum_{i=1}^{N}\gamma_i\left(s_i B - \sum_{t=1}^{T_i} x_{i,t}\cdot c_{i,t}\right).
\end{aligned}$$

Expanding and rearranging terms:

$$\mathcal{L}(x_{i,t}, \lambda, \gamma_i) = \sum_{i=1}^{N} \sum_{t=1}^{T_i} \left(x_{i,t} \cdot r_{i,t} - \lambda \cdot x_{i,t} \cdot c_{i,t} + \lambda \cdot \frac{B}{T} + \gamma_i \cdot x_{i,t} \cdot c_{i,t} - \gamma_i \cdot s_i B \right).$$

Optimal Selection Condition To maximize $\mathcal{L}(x_{i,t}, \lambda, \gamma_i)$, we select $x_{i,t} = 1$ whenever:

$$r_{i,t} - \lambda \cdot c_{i,t} + \gamma_i \cdot c_{i,t} > 0.$$

Rearranging:

$$r_{i,t} > (\lambda - \gamma_i) \cdot c_{i,t}.$$

Thus, campaign i wins the auction if:

$$\frac{r_{i,t}}{c_{i,t}} > (\lambda - \gamma_i).$$

Optimal Bid Formula From the standard bidding mechanism, the optimal bid is given by:

$$b^*_{i,t} = \frac{r_{i,t}}{\lambda - \gamma_i}. \tag{5.4}$$

where:

- λ controls the overall budget pacing.
- γ_i ensures that each campaign receives at least $s_i B$ of the budget.

Dual Gradient Descent Updates We update λ and γ_i using stochastic gradient descent (SGD):

$$\lambda \leftarrow \lambda - \epsilon_\lambda \left(\frac{B}{T} - x_{i,t} \cdot c_{i,t} \right).$$

$$\gamma_i \leftarrow \gamma_i - \epsilon_\gamma \left(x_{i,t} \cdot c_{i,t} - s_i \cdot \frac{B}{T_i} \right).$$

These updates ensure:

- λ adjusts the overall budget pacing.
- γ_i enforces the minimum budget constraint for each campaign.

Interpretation of the Formula

- If $\gamma_i = 0$ (i.e., the minimum budget constraint is inactive for campaign i), then:

$$b_{i,t}^* = \frac{r_{i,t}}{\lambda}.$$

  which is the standard max delivery bid formula.

- If $\gamma_i > 0$ (i.e., campaign i is underfunded and needs more budget), then:
  - The bid increases (since $\lambda - \gamma_i$ is reduced).
  - The campaign becomes more competitive and wins more auctions.
- If γ_i is too high, the campaign overcompensates, leading to potential inefficiencies.

3.2 Some Practical Considerations

1. heterogeneous distribution of campaigins in the group, e.g., pctr distributions are different

CHAPTER 6

DEEP RETENTION PROBLEM

In this chapter, we discuss the deep retention optimization problem, which arises in a special business scenario where post-conversion retention is critical for advertisers.

1 Deep Retention Problem

For some particular objective such as APP ads, user acquisition alone is not enough—retaining high-value users who engage deeply with the app is critical for long-term success. For example, traditional Cost-Per-Install (CPI) bidding focuses only on getting users to install an app, but many of these users never return or contribute to revenue.

To address this issue, Deep Retention bidding is introduced to optimize not just for installs, but for long-term user engagement and monetization.Deep retention focuses on in-app behaviors that signal lasting value, such as:

- Frequent app reopens (D3, D7, D14, D30 retention rates)
- In-app purchases (IAPs) and subscription activations
- Product views, add-to-cart, and purchases in e-commerce apps
- Level completion, engagement with premium content, or social interactions in gaming and entertainment apps

Problem Formulation Under standard second price auction, we can formulate the deep conversion problem as follows:

$$
\begin{aligned}
\max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot c_t \\
\text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \le B \\
& \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} \le C \\
& \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t \cdot d_t} \le D.
\end{aligned} \tag{6.1}
$$

where

- B is the total budget.
- T is the (predicted) number of auction opportunities.
- x_t is the binary decision varaible indicating whether campaign wins t-th auction opportunity.
- c_t is the highest competing bid per impression for the t-th auction opportunity.
- r_t is the impression to action(e.g., App install) conversion rate for t-th auction opportunity for the campaign.
- d_t is the deep conversion rate conditioned on action for t-th auciton opportunity for the campaign.

- C is the cap for average cost per action.
- D is the cap for average cost per deep conversion.

Derivation of the Optimal Bid We first introduce the Lagrangian with dual parameters:

$$\begin{aligned}\mathcal{L}(x_t, \lambda, \mu, \gamma) &= \sum_{t=1}^{T} x_t \cdot c_t - \lambda \left(\sum_{t=1}^{T} x_t \cdot c_t - B \right) \\ &\quad - \mu \left(\sum_{t=1}^{T} x_t \cdot c_t - C \cdot \sum_{t=1}^{T} x_t \cdot r_t \right) - \gamma \left(\sum_{t=1}^{T} x_t \cdot c_t - D \cdot \sum_{t=1}^{T} x_t \cdot r_t \cdot d_t \right) \\ &= \sum_{t=1}^{T} \left\{ x_t \cdot [c_t \cdot (1 - \lambda - \mu - \gamma) + \mu \cdot C \cdot r_t + \gamma \cdot D \cdot r_t \cdot d_t] + \lambda \cdot \frac{B}{T} \right\}.\end{aligned}$$

To maximize $\mathcal{L}(x_t, \lambda, \mu, \gamma)$, we set $x_t = 1$ whenever:

$$c_t \cdot (1 - \lambda - \mu - \gamma) + \mu \cdot C \cdot r_t + \gamma \cdot D \cdot r_t \cdot d_t > 0,$$

and $x_t = 0$ otherwise. The dual objective $\mathcal{L}^*(\lambda, \mu, \gamma)$ then becomes:

$$\mathcal{L}^*(\lambda, \mu, \gamma) = \sum_{t=1}^{T} \left\{ (c_t \cdot (1 - \lambda - \mu - \gamma) + \mu \cdot C \cdot r_t + \gamma \cdot D \cdot r_t \cdot d_t)_+ + \lambda \cdot \frac{B}{T} \right\}.$$

Suppose we find a feasible solution to the dual problem:

$$\lambda^*, \mu^*, \gamma^* = \underset{\lambda \geq 0, \mu \geq 0, \gamma \geq 0}{\arg\min} \ \mathcal{L}^*(\lambda, \mu, \gamma).$$

The optimal bid per impression is then given by:

$$b_t^* = \frac{\mu^* \cdot C + \gamma^* \cdot D \cdot d_t}{\lambda^* + \mu^* + \gamma^* - 1} \cdot r_t. \tag{6.2}$$

The bid per conversion can be expressed as:

$$b^*_{conversion,t} = \frac{\mu^* \cdot C + \gamma^* \cdot D \cdot d_t}{\lambda^* + \mu^* + \gamma^* - 1}. \tag{6.3}$$

Interpretation of the Bidding Formula Examining (6.3), we observe the roles of the dual parameters:

- λ governs controls budget pacing.
- μ controls the cost per conversion.
- γ controls the cost per deep conversion.

If the deep conversion rate is unbiased, we define $C' = D \cdot d_t$ as the normalized cost per conversion target based on the deep conversion target. The term $\mu \cdot C + \gamma \cdot C'$ represents a linear combination of the original conversion cost target and the normalized deep conversion cost target.

In theory, only the most restrictive constraint should be active, meaning either μ or γ should be positive while the other remains zero. A well-designed pacing algorithm should iteratively update μ and γ based on real-time observed data, allowing the active constraint to emerge dynamically.

Algorithm Design There are multiple approaches to designing algorithms for bid updates. In this section, we provide a detailed discussion on using a PID controller for this scenario. We will briefly mention alternative approaches based on Model Predictive Control (MPC) and Dual Online Gradient Descent (DOGD) in the section of Remarks.

For simplicity, we rewrite (6.2) as:

$$b_t^* = \alpha \cdot (\beta_1 \cdot C \cdot r_t + \beta_2 \cdot D \cdot d_t \cdot r_t),$$

where:

$$\alpha = \frac{1}{\lambda^*}, \quad \beta_1 = \frac{\lambda^*}{\lambda^* + \mu^* + \gamma^* - 1} \cdot \mu^*, \quad \beta_2 = \frac{\lambda^*}{\lambda^* + \mu^* + \gamma^* - 1} \cdot \gamma^*.$$

Here, α is responsible for budget delivery control, while β_1 and β_2 regulate conversion cost control and deep conversion cost control, respectively.[1]

With this formulation in mind, we can design a PID control mechanism as follows: At each pacing cycle, we observe the actual budget spend, cost per conversion, and cost per deep conversion. These values are then compared to their respective targets—namely, the target spend, target cost per conversion C, and target cost per deep conversion D. The differences between observed and target values are used as error signals for the PID controller, which subsequently updates the dual parameters accordingly.

We summarize the discussion above in the following Algorithm 32:

Practical Considerations We previously mentioned that the error signals for (deep) conversion cost control are derived from the differences between the target (deep) conversion cost and the actual (deep) conversion cost. However, in practice, each pacing cycle operates on a minute-scale interval, meaning that conversion events (e.g., app installs, in-app purchases) are unlikely to occur in most pacing intervals. If we rely solely on observed data as error signal inputs, the bid dynamics may become highly unstable.

A common workaround is to use predicted (deep) conversion rates as approximations for actual observed events. To obtain these approximations, for each winning auction opportunity t within a pacing cycle, we collect the tuple (c_t, r_t, d_t), where:

- c_t is the winning price,

[1] By adopting this approach, we simplify the nonlinear interactions between delivery and cost control. Note that both β_1 and β_2 depend on λ^*, μ^*, and γ^*. The rewritten formula can be regarded as a linear approximation of the actual optimal bid, which may be suboptimal in some cases.

Algorithm 32 PID-Based Bid Algorithm for Deep Retention Problem

```
Require:
 1: $B$: Total budget, $T$: Total (predicted) number of auction opportunities, $C$: Target cost
    per conversion, $D$: Target cost per deep conversion, PID gains $K_p, K_i, K_d$, positive initial
    values $\alpha_0, \beta_{1,0}, \beta_{2,0}$.
Ensure: $b_t^*$: Optimal bid per impression.
        Step 1: Initialize Parameters
 2: Initialize $\alpha, \beta_1, \beta_2 \leftarrow \alpha_0, \beta_{1,0}, \beta_{2,0}$
 3: Initialize $I_\alpha, I_{\beta_1}, I_{\beta_2} \leftarrow 0, 0, 0$
 4: Initialize previous errors $e_\alpha^{prev}, e_{\beta_1}^{prev}, e_{\beta_2}^{prev} \leftarrow 0, 0, 0$
        Step 2: Iterative Bid Updates
 5: for each pacing cycle do
 6:     Observe spend $S_{obs}$, cost per conversion $C_{obs}$, and cost per deep conversion $D_{obs}$
 7:     Count the number of auction opportunities $N$
 8:     Compute errors: $e_\alpha = \frac{B}{T} \cdot N - S_{obs}$, $e_{\beta_1} = C - C_{obs}$, $e_{\beta_2} = D - D_{obs}$
 9:     Update integral terms:
            $I_\alpha \leftarrow I_\alpha + e_\alpha, \quad I_{\beta_1} \leftarrow I_{\beta_1} + e_{\beta_1}, \quad I_{\beta_2} \leftarrow I_{\beta_2} + e_{\beta_2}$
10:     Compute derivative terms:
            $D_\alpha = e_\alpha - e_\alpha^{prev}, \quad D_{\beta_1} = e_{\beta_1} - e_{\beta_1}^{prev}, \quad D_{\beta_2} = e_{\beta_2} - e_{\beta_2}^{prev}$
11:     Update parameters using PID:
            $\alpha \leftarrow \alpha \cdot \exp(K_p e_\alpha + K_i I_\alpha + K_d D_\alpha)$
            $\beta_1 \leftarrow \beta_1 \cdot \exp(K_p e_{\beta_1} + K_i I_{\beta_1} + K_d D_{\beta_1})$
            $\beta_2 \leftarrow \beta_2 \cdot \exp(K_p e_{\beta_2} + K_i I_{\beta_2} + K_d D_{\beta_2})$
12:     for each auction opportunity $t$ do
13:         Get coversion rate $r_t$ and deep conversion rate $d_t$ from prediction models
14:         Compute optimal bid:
                $b_t^* = \alpha \cdot (\beta_1 \cdot C \cdot r_t + \beta_2 \cdot D \cdot d_t \cdot r_t)$
15:         Submit $b_t^*$
16:     end for
17:     Update previous errors:
            $e_\alpha^{prev} \leftarrow e_\alpha, \quad e_{\beta_1}^{prev} \leftarrow e_{\beta_1}, \quad e_{\beta_2}^{prev} \leftarrow e_{\beta_2}$
18: end for
```

- r_t is the predicted conversion rate,
- d_t is the predicted deep conversion rate.

Using these values, we approximate the actual cost per conversion and cost per deep conversion as:

$$\tilde{C}_{obs} = \frac{\sum_t c_t}{\sum_t r_t}, \quad \tilde{D}_{obs} = \frac{\sum_t c_t}{\sum_t r_t \cdot d_t}.$$

These approximations help smooth bid updates by reducing the impact of sparse conversion events, ensuring a more stable bidding dynamics.

2 Remarks

2.1 MPC and DOGD Approaches for Deep Retention Problem

TBA

2.2 A Variant Formulation of Deep Retention Problem

We present an alternative formulation of the deep retention problem.

Problem Formulation We formulate the problem as:

$$\begin{aligned} \max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot c_t \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \leq B, \\ & \frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t} \leq C, \\ & \frac{\sum_{t=1}^{T} x_t \cdot r_t \cdot C}{\sum_{t=1}^{T} x_t \cdot r_t \cdot d_t} \leq D. \end{aligned} \tag{6.4}$$

It is easy to see that if the constraints in (6.4) hold, then the constraints in the original formulation (6.1) also hold automatically. Comparing (6.4) to (6.1), the only difference lies in the deep conversion cost control constraint, where we replace:

$$\frac{\sum_{t=1}^{T} x_t \cdot c_t}{\sum_{t=1}^{T} x_t \cdot r_t \cdot d_t} \leq D \tag{6.5}$$

with:

$$\frac{\sum_{t=1}^{T} x_t \cdot r_t \cdot C}{\sum_{t=1}^{T} x_t \cdot r_t \cdot d_t} \leq D. \tag{6.6}$$

Rewriting (6.4), we obtain:

$$\frac{\sum_{t=1}^{T} x_t \cdot r_t \cdot d_t}{\sum_{t=1}^{T} x_t \cdot r_t} \geq \frac{C}{D}.$$

Here, $\frac{C}{D}$ represents the desired deep conversion rate given a shallow conversion, while the left-hand side corresponds to the (predicted) deep conversion rate. Therefore, constraint (6.6) enforces that the actual deep conversion rate is no less than the target deep conversion rate. This formulation focuses on optimizing deep conversions conditioned on conversion events occurring, whereas (6.5) focuses on the end-to-end optimization of deep conversion events.

The optimal bid formula for (6.4) is given by:

$$b_t^* = \frac{(\mu^* - \gamma^*) \cdot C \cdot r_t - \gamma^* \cdot D \cdot r_t \cdot d_t}{\lambda^* + \mu^* - 1},$$

where λ^*, μ^*, and γ^* are the dual parameters corresponding to the budget delivery constraint, conversion constraint, and deep conversion constraint, respectively. The derivation of this formula is left as an exercise for the reader.

PID Controller Design We now design a PID controller to update the bids. First, we rewrite the above formula as:

$$b_t^* = \underbrace{\alpha}_{\text{Delivery Control}} \cdot \underbrace{\beta_1 \cdot C \cdot r_t}_{\text{Conversion Cost Control}} \cdot \underbrace{\left[1 + \beta_2 \left(\frac{d_t}{\frac{C}{D}} - 1\right)\right]}_{\text{Deep Conversion Adjustment}} \tag{6.7}$$

where:

$$\alpha = \frac{1}{\lambda^*}, \quad \beta_1 = \frac{\lambda^* \cdot \mu^*}{\lambda^* + \mu^* - 1}, \quad \beta_2 = \frac{\gamma^*}{\mu^*}.$$

From (6.7), we can see that the bid consists of three components: Delivery control (α), Conversion cost control (β_1), and Deep conversion adjustment (β_2).

The PID controller is designed for each part as follows:

- **Delivery Control α:** The goal of this component is to ensure that the overall budget delivery remains on target. At each pacing interval, we collect the actual spend and compare it to the target spend. The difference serves as the error signal for α modulation in the PID controller.

- **Conversion Cost Control β_1:** The objective is to maintain the cost per conversion at the target level C. The error signal for β_1 is defined as the difference between the target cost C and the observed cost per conversion within the pacing interval.

- **Deep Conversion Adjustment β_2:** The goal is to ensure that the observed deep conversion rate meets or exceeds the target deep conversion rate $\frac{C}{D}$. Intuitively, if the deep conversion rate is already higher than the target, the deep conversion constraint is inactive, and no adjustment is needed. In this case, β_2 should approach zero, meaning no additional bid adjustment. If the deep conversion rate is below the target, we need to increase β_2. The term:

$$1 + \beta_2 \left(\frac{d_t}{\frac{C}{D}} - 1 \right)$$

ensures that when $d_t > \frac{C}{D}$, the bid price increases, whereas when $d_t < \frac{C}{D}$, the bid price is suppressed. This aligns with the intuition that when the deep conversion target is missed, we boost bids for auction requests with a higher deep conversion rate while lowering bids for those with a lower deep conversion rate. This analysis suggests that the error signal for β_2 should be the difference between the target deep conversion rate $\frac{C}{D}$ and the observed deep conversion rate in the past pacing interval.

PID Control Implementation Based on the above discussion, we design the PID control algorithm for solving (6.4). At each pacing interval, we retrieve: the actual spend S_{obs},the observed cost per conversion C_{obs} and the observed deep conversion rate d_{obs}. The error signals for α, β_1, and β_2 are computed as:

$$e_\alpha = S_{target} - S_{obs}, \quad e_{\beta_1} = C - C_{obs}, \quad e_{\beta_2} = \frac{C}{D} - d_{obs}.$$

The PID control equations are then applied to modulate these parameters accordingly. We summarize the discussion here in the following Algorithm 33. In practice, the observed conversions and cost per result can be replaced with predicted values from prediction models to mitigate data sparsity issues, as discussed in the previous section.

Algorithm 33 PID-Based Bid Algorithm for Varint Deep Conversion Problem

```
Require: $B$: Total budget, $T$: Total (predicted) number of auction opportunities, $C$: Target cost per conversion, $D$: Target cost per deep conversion, PID gains $K_p, K_i, K_d$, initial values $\alpha_0, \beta_{1,0}, \beta_{2,0}$.
Ensure: $b_t^*$: Optimal bid per impression.
        Step 1: Initialize Parameters
 1: Initialize $\alpha, \beta_1, \beta_2$ and integral terms $I_\alpha, I_{\beta_1}, I_{\beta_2}$
 2: Initialize previous errors $e_\alpha^{prev}, e_{\beta_1}^{prev}, e_{\beta_2}^{prev}$
        Step 2: Iterative Bid Updates
 3: for each pacing interval do
 4:     Count the number of auction opportunities $N$
 5:     Observe actual spend $S_{obs}$, cost per conversion $C_{obs}$, and deep conversion rate $d_{obs}$
 6:     Compute error signals:
            $$e_\alpha = \frac{B}{T} \cdot N - S_{obs}, \quad e_{\beta_1} = C - C_{obs}, \quad e_{\beta_2} = \frac{C}{D} - d_{obs}$$
 7:     Update integral terms:
            $$I_\alpha \leftarrow I_\alpha + e_\alpha, \quad I_{\beta_1} \leftarrow I_{\beta_1} + e_{\beta_1}, \quad I_{\beta_2} \leftarrow I_{\beta_2} + e_{\beta_2}$$
 8:     Compute derivative terms:
            $$D_\alpha = e_\alpha - e_\alpha^{prev}, \quad D_{\beta_1} = e_{\beta_1} - e_{\beta_1}^{prev}, \quad D_{\beta_2} = e_{\beta_2} - e_{\beta_2}^{prev}$$
 9:     Update PID-controlled parameters:
            $$\alpha \leftarrow \alpha \cdot \exp\left(K_p e_\alpha + K_i I_\alpha + K_d D_\alpha\right)$$
            $$\beta_1 \leftarrow \beta_1 \cdot \exp\left(K_p e_{\beta_1} + K_i I_{\beta_1} + K_d D_{\beta_1}\right)$$
            $$\beta_2 \leftarrow \beta_2 \cdot \exp\left(K_p e_{\beta_2} + K_i I_{\beta_2} + K_d D_{\beta_2}\right)$$
10:     for each auction opportunity $t$ do
11:         Get conversion rate $r_t$ and deep conversion rate $d_t$ from prediction models
12:         Compute optimal bid:
                $$b_t^* = \alpha \cdot (\beta_1 \cdot C \cdot r_t) \cdot \left[1 + \beta_2 \left(\frac{d_t}{\frac{C}{D}} - 1\right)\right]$$
13:         Submit $b_t^*$
14:     end for
15:     Update previous errors:
            $$e_\alpha^{prev} \leftarrow e_\alpha, \quad e_{\beta_1}^{prev} \leftarrow e_{\beta_1}, \quad e_{\beta_2}^{prev} \leftarrow e_{\beta_2}$$
16: end for
```

CHAPTER 7

REACH & FREQUENCY AND GUARANTEED DELIVERY

In this chapter, we discuss two brand bidding products: Reach & Frequency and Guaranteed Delivery. For these products, advertisers specify branding requirements related to reach and frequency metrics or demand guarantees on impression delivery. We demonstrate how these requirements can be formulated as optimization problems and how control methods can be leveraged to design corresponding pacing algorithms.

In this chapter, we discuss two popular brand awareness bidding products: Reach & Frequency and Guaranteed Delivery. These products provide advertisers with greater control over ad delivery, audience reach, and frequency management, helping them achieve their marketing objectives more effectively.

1 Reach and Frequency Problem

Problem Formulation Reach and Frequency(R&F) is a product designed to help advertisers optimize brand ad campaigns by focusing on two key metrics:

- Reach: The number of unique users exposed to the ad.
- Frequency: The number of times each user sees the ad within a specific timeframe.

This bidding product ensures that advertisers can plan and predict their campaign's outcomes more accurately, making it ideal for brands that need measurable and scalable visibility while managing overexposure and staying within a fixed budget.

R&F can be formulated in terms of the target frequency. Suppose there are M targeting users for this ad campaign. For each user m, the number of eligible auction opportunities is T_m. The desired frequency for each user within the campaign lifetime is between F_l and F_u. The reach and frequency can be formulated as the following optimization problem:

$$\begin{aligned} \max_{x_{m,t}\in\{0,1\}} \quad & \sum_{m=1}^{M}\sum_{t=1}^{T_m} x_{m,t} \\ \text{s.t.} \quad & \sum_{m=1}^{M}\sum_{t=1}^{T_m} x_{m,t}\cdot c_{m,t} \le B, \\ & \sum_{t=1}^{T_m} x_{m,t} \ge F_l, \quad \forall m = 1,2,\ldots,M, \\ & \sum_{t=1}^{T_m} x_{m,t} \le F_u, \quad \forall m = 1,2,\ldots,M, \end{aligned} \tag{7.1}$$

where $x_{m,t} \in \{0,1\}$ is an indicator of whether the campaign wins the t-th auction for user m, $c_{m,t}$ is the cost for the t-th auction for user m, and B is the total budget of the campaign.

Derivation of the Optimal Bid We use the DOGD method to solve this problem. The Lagrangian is defined as:

$$\begin{aligned}
&\mathcal{L}(\lambda, \mu_m, \gamma_m, x_{m,t}) \\
&= \sum_{m=1}^{M} \sum_{t=1}^{T_m} x_{m,t} + \lambda \cdot \left(B - \sum_{m=1}^{M} \sum_{t=1}^{T_m} x_{m,t} \cdot c_{m,t} \right) + \sum_{m=1}^{M} \mu_m \cdot \left(F_u - \sum_{t=1}^{T_m} x_{m,t} \right) \\
&\quad + \sum_{m=1}^{M} \gamma_m \cdot \left(\sum_{t=1}^{T_m} x_{m,t} - F_l \right) \\
&= \sum_{m=1}^{M} \sum_{t=1}^{T_m} [1 - \lambda c_{m,t} - \mu_m + \gamma_m] \cdot x_{m,t} + \lambda B + \sum_{m=1}^{M} \mu_m F_u - \sum_{m=1}^{M} \gamma_m F_l.
\end{aligned}$$

To maximize $\mathcal{L}(\lambda, \mu_m, \gamma_m, x_{m,t})$, we set $x_{m,t} = 1$ whenever $1 - \lambda c_{m,t} - \mu_m + \gamma_m > 0$, and $x_{m,t} = 0$ otherwise. This gives the dual problem's objective function:

$$\begin{aligned}
\mathcal{L}^*(\lambda, \mu_m, \gamma_m) &= \max_{x_{m,t} \in \{0,1\}} \mathcal{L}(\lambda, \mu_m, \gamma_m, x_{m,t}) \\
&= \sum_{m=1}^{M} \sum_{t=1}^{T_m} (1 - \lambda c_{m,t} - \mu_m + \gamma_m)_+ + \lambda B + \sum_{m=1}^{M} \mu_m F_u - \sum_{m=1}^{M} \gamma_m F_l,
\end{aligned}$$

where $(z)_+ = \max(0, z)$.

Under the Second Price Auction (SPA), the optimal bid for user m is given by:

$$b_m^* = \frac{1 - \mu_m^* + \gamma_m^*}{\lambda^*},$$

where λ^*, μ_m^*, and γ_m^* are the solutions to the dual problem.

We can apply stochastic gradient descent (SGD) to iteratively update these parameters:

$$\begin{aligned}
\lambda_t &\leftarrow \lambda_{t-1} - \epsilon_t \cdot \frac{\partial}{\partial \lambda} \mathcal{L}^*, \\
\mu_{m,t} &\leftarrow \mu_{m,t-1} - \epsilon_t \cdot \frac{\partial}{\partial \mu_m} \mathcal{L}^*, \\
\gamma_{m,t} &\leftarrow \gamma_{m,t-1} - \epsilon_t \cdot \frac{\partial}{\partial \gamma_m} \mathcal{L}^*.
\end{aligned}$$

The gradients are computed as:

$$\begin{aligned}
\frac{\partial}{\partial \lambda} \mathcal{L}^* &= \frac{B}{T} - c_{m,t} \cdot x_{m,t}, \\
\frac{\partial}{\partial \mu_m} \mathcal{L}^* &= \frac{F_u}{T_m} - x_{m,t}, \\
\frac{\partial}{\partial \gamma_m} \mathcal{L}^* &= x_{m,t} - \frac{F_l}{T_m}.
\end{aligned}$$

where $T = \sum T_m$ is the total number of auction opportunities for this campaign cross all targeting users. The dual parameter λ is responsible for the overall delivery control. The gradient of λ is simply the gap between the expected target cost per auction and the actual spend per auction. The other two sets of dual parameters, μ_m and γ_m, are responsible for controlling the impression frequency and cadence for each user m. The gradients of these parameters compare the actual impressions with the expected lower and upper bounds per auction. This comparison determines how to tweak the bid to achieve the desired frequency for the ad campaign.

Algorithm Design To implement this reach and frequency algorithm, we initialize λ_0 for overall delivery control, and a set of $\mu_{m,0}$ and $\gamma_{m,0}$ for frequency control for each user m. At each step, we observe the actual spend and impressions. The parameter λ is always updated based on the update rule described above, while μ_m and γ_m are updated only for the user m who triggers the auction request. The idea discussed above can be summarized as the following Algorithm 34:

Algorithm 34 Reach and Frequency Algorithm with Dual Parameters

```
Require: $B$: Total budget, $F_l$: Minimum frequency, $F_u$: Maximum frequency
Require: $T_m$: Expected number of auction opportunities from user $m$
Require: $\lambda_0$: Initial dual parameter for delivery control
Require: $\mu_{m,0}, \gamma_{m,0}$: Initial dual parameters for frequency control for each user $m$
Require: $\epsilon_t$: Learning rates for $\lambda$, $\mu$, and $\gamma$
Ensure: Bids $b_{m,t}$ for each user $m$
1: Compute the total auction opportunites
       $$T = \sum T_m$$
2: Initialize $\lambda_0 \leftarrow \lambda_0$, $\mu_{m,0} \leftarrow \mu_{m,0}$, $\gamma_{m,0} \leftarrow \gamma_{m,0}$ for all $m = 1, \ldots, M$
3: for each auction request at time $t$ do
4:     Observe the user $m$ triggering the auction and the auction cost $c_{m,t}$
5:     Observe the impression $x_{m,t}$
6:     Update dual parameters for delivery control
           $$\lambda_t \leftarrow \lambda_{t-1} - \epsilon_t \cdot \left(\frac{B}{T} - c_{m,t}\right)$$
7:     Update dual parameters of user $m$ for frequency control
           $$\mu_{m,t} \leftarrow \mu_{m,t-1} - \epsilon_t \cdot \left(\frac{F_u}{T_m} - x_{m,t}\right)$$
           $$\gamma_{m,t} \leftarrow \gamma_{m,t-1} - \epsilon_t \cdot \left(x_{m,t} - \frac{F_l}{T_m}\right)$$
8:     Compute the bid price:
           $$b_{m,t} = \frac{1 - \mu_{m,t} + \gamma_{m,t}}{\lambda_t}$$
9: end for
```

As we mentioned before, in practice, it's more common to implement the algorithm in a mini-batch manner, the bids stay unchanged within a time As we mentioned before, in practice, it is more common to implement the algorithm in a mini-batch manner, where the bids remain unchanged within a time duration Δt. We first compute the mini-batch gradients for all auction opportunities within $(t, t + \Delta t)$.

For λ, the gradient is:

$$\begin{aligned}\sum_{s\in(t,t+\Delta t)} \frac{\partial}{\partial \lambda}\mathcal{L}^* = &\sum_{s\in(t,t+\Delta t)} \left(\frac{B}{T} - c_{m,s} \cdot x_{m,s}\right) \\ =&\frac{R(t)}{T}B - S(t),\end{aligned}$$

where $R(t)$ is the number of observed auction requests, and $S(t)$ is the actual spend during Δt.

For μ_m, the gradient is:

$$\begin{aligned}\sum_{s\in(t,t+\Delta t)} \frac{\partial}{\partial \mu_m}\mathcal{L}^* = &\sum_{s\in(t,t+\Delta t)} \left(\frac{F_u}{T_m} - x_{m,s}\right) \\ =&\frac{R_m(t)}{T_m}F_u - I_m(t),\end{aligned}$$

where $R_m(t)$ is the number of observed auction requests from user m, and $I_m(t)$ is the number of impressions shown to this user.

For γ_m, the gradient is:

$$\begin{aligned}\sum_{s\in(t,t+\Delta t)} \frac{\partial}{\partial \gamma_m}\mathcal{L}^* = &\sum_{s\in(t,t+\Delta t)} \left(x_{m,s} - \frac{F_l}{T_m}\right) \\ =&I_m(t) - \frac{R_m(t)}{T_m}F_l.\end{aligned}$$

We summarize the mini-batch algorithm as follows:

Algorithm 35 Mini-Batch Reach and Frequency Algorithm

```
Require: $B$: Total budget, $F_l$: Minimum frequency, $F_u$: Maximum frequency
Require: $\lambda_0$, $\mu_{m,0}$, $\gamma_{m,0}$: Initial dual parameters
Require: $\Delta t$: Mini-batch interval, $\epsilon_\lambda$, $\epsilon_\mu$, $\epsilon_\gamma$: Learning rates
Ensure: Optimal bids $b_m^*$ for each user $m$
1: Initialize $\lambda \leftarrow \lambda_0$, $\mu_m \leftarrow \mu_{m,0}$, $\gamma_m \leftarrow \gamma_{m,0}$ for all $m = 1, \dots, M$
2: for each mini-batch interval $(t, t + \Delta t)$ do
3:     Observe $R(t)$, $S(t)$, $R_m(t)$, and $I_m(t)$ for all users
4:     Compute mini-batch gradients:
           $\lambda \leftarrow \lambda - \epsilon_\lambda \cdot \left( \frac{R(t)}{T} B - S(t) \right)$
           $\mu_m \leftarrow \mu_m - \epsilon_\mu \cdot \left( \frac{R_m(t)}{T_m} F_u - I_m(t) \right), \quad \forall m$
           $\gamma_m \leftarrow \gamma_m - \epsilon_\gamma \cdot \left( I_m(t) - \frac{R_m(t)}{T_m} F_l \right), \quad \forall m$
5:     Compute the bid for each user $m$:
           $b_{m,t} = \frac{1 - \mu_m + \gamma_m}{\lambda}$
6: end for
```

2 Guaranteed Delivery Problem

Guaranteed Delivery (GD) ads, also referred to as programmatic guaranteed ads or reserved media buys, are advertising deals in which advertisers purchase a predetermined volume of impressions (or another agreed-upon metric, such as video views) directly from a publisher or via a platform. The price is determined upfront, and a certain portion of the inventory is reserved for these ads.

While traditional GD ads are sold at a fixed price for reserved inventory without participating in real-time auctions, some ad platforms incorporate an internal auction mechanism to enhance efficiency and reduce costs. This approach allows the system to optimize inventory allocation by identifying lower-cost opportunities rather than always serving GD ads at a fixed high CPM.

Problem Formulation We present a simple formulation of the Guaranteed Delivery (GD) problem:

$$\begin{aligned} \min_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot c_t \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t \geq G. \end{aligned} \tag{7.2}$$

where:

- T represents the total available inventory for the campaign, i.e., the number of eligible auction opportunities.
- G is the guaranteed delivery goal of impressions specified in the deal. We assume $G \leq T$ to ensure sufficient inventory for meeting the delivery goal.
- c_t is the highest competing bid at the t-th auction opportunity. Under a second-price auction, this is also the winning price.
- x_t is a binary decision variable indicating whether the campaign wins the t-th auction opportunity. Under a second-price auction, $x_t = \mathbb{1}_{\{b_t > c_t\}}$, where b_t is the bid for the impression.

Derivation of the Optimal Bid We apply the primal-dual method to solve (7.2). The first step is to formulate the Lagrangian:

$$\begin{aligned} \mathcal{L}(x_t, \lambda) &= \sum_{t=1}^{T} x_t \cdot c_t + \lambda \cdot \left(G - \sum_{t=1}^{T} x_t \right) \\ &= \sum_{t=1}^{T} \left[x_t \cdot (c_t - \lambda) + \lambda \cdot \frac{G}{T} \right]. \end{aligned}$$

To minimize $\mathcal{L}(x_t, \lambda)$, we set $x_t = 0$ whenever $c_t - \lambda > 0$ and $x_t = 1$ otherwise. Then, we obtain:

$$\mathcal{L}^*(\lambda) = \min_{x_t \in \{0,1\}} \mathcal{L}(x_t, \lambda) = \sum_{t=1}^{T} \left[\underbrace{(c_t - \lambda) \cdot \mathbb{1}_{\{c_t < \lambda\}} + \lambda \cdot \frac{G}{T}}_{f_t(\lambda)} \right].$$

The corresponding dual problem is:

$$\max_{\lambda \geq 0} \mathcal{L}^*(\lambda) = \max_{\lambda \geq 0} \sum_{t=1}^{T} \left[(c_t - \lambda) \cdot \mathbb{1}_{\{c_t < \lambda\}} + \lambda \cdot \frac{G}{T} \right].$$

Assuming the problem is feasible, we determine the optimal dual parameter:

$$\lambda^* = \arg\max_{\lambda \geq 0} \mathcal{L}^*(\lambda).$$

By the KKT condition, we have:

$$\sum_{t=1}^{T} x_t = G.$$

Thus, the optimal bid per impression is:

$$b^* = \lambda^*.$$

Algorithm Design We design a bid update rule using the Dual Online Gradient Descent (DOGD) algorithm. Other approaches, such as PID control and Model Predictive Control (MPC), are left for interested readers.

Applying stochastic gradient ascent (since the dual problem is a maximization problem), the update rule for λ is:

$$\lambda \leftarrow \lambda + \epsilon \cdot \frac{\partial}{\partial \lambda} f_t(\lambda) = \lambda + \epsilon \cdot \left(\frac{G}{T} - \mathbb{1}_{\{c_t < \lambda\}} \right).$$

Since we bid using λ, the term $\mathbb{1}_{\{c_t<\lambda\}}$ corresponds to x_t, which indicates whether the t-th auction opportunity is won. The term $\frac{G}{T}$ represents the expected number of impressions the campaign should win per auction opportunity.

For a mini-batch update within each pacing interval, the rule is:

$$\lambda \leftarrow \lambda + \epsilon \cdot \left(\frac{G}{T} \cdot N - W \right),$$

where:

- W is the number of impressions won by the campaign within the pacing interval.
- N is the number of auction opportunities.
- $\frac{G}{T} \cdot N$ represents the target number of impressions.

This update rule ensures that if the actual impressions are below the target, we increase the bid, and if the impressions exceed the target, we decrease the bid. This guarantees smooth budget pacing and delivery alignment.

We summarize the idea discussed above in the following Algorithm 36:

Algorithm 36 DOGD-Based Bid Algorithm for Guaranteed Delivery

```
Require: $G$: Guaranteed impressions goal, $T$: Total available inventory, $\epsilon$: Learning rate,
    Initial bid parameter $\lambda_0$.
Ensure: $b^*$: Optimal bid per impression.
        Step 1: Initialize Parameters
 1: Initialize $\lambda \leftarrow \lambda_0$
        Step 2: Iterative Bid Updates
 2: for each pacing interval do
 3:     Observe actual impressions won $W$ and number of auction opportunities $N$
 4:     Compute target impressions per interval:
            $G_{target} = \frac{G}{T} \cdot N$
 5:     Compute update step:
            $\lambda \leftarrow \lambda + \epsilon \cdot (G_{target} - W)$
 6:     Update bid:
            $b^* = \lambda$
 7:     Submit $b^*$ for the auction opportunities in the next pacing interval
 8: end for
```

For more details of guaranteed delivery problems, one may refer to the related papers, such as [29], [13], [30], [18].

3 Remarks

- Marketplace level formulation of R&F problem
- Sliding window optimization for R&F.
- GD cost cap problem: [73]
- GD for unique reach
- Implementation tips(e.g., rescaling)
- prediction of expected number of auction opportunity per use
- Compare different frequency control setups: lower bound + upper bound vs single median target

3.1 Fixed Frequency Problem

Fixed frequency target problem: Sometimes advertisers want to target a specific frequency (e.g., F). In this case, we simply set $F_l = F_u = F$, and Equation 7.1 simplifies to:

$$\begin{aligned}
\max_{x_{m,t}\in\{0,1\}} \quad & \sum_{m=1}^{M}\sum_{t=1}^{T_m} x_{m,t} \\
\text{s.t.} \quad & \sum_{m=1}^{M}\sum_{t=1}^{T_m} x_{m,t}\cdot c_{m,t} \le B, \\
& \sum_{t=1}^{T_m} x_{m,t} = F, \quad \forall m = 1, 2, \dots, M.
\end{aligned}$$

The corresponding bid formula is:

$$b_{m,t} = \frac{1-\mu_{m,t}}{\lambda_t}.$$

The update rules for the dual parameters are:

$$\lambda_t \leftarrow \lambda_{t-1} - \epsilon_t \cdot \left(\frac{B}{T} - c_{m,t}\right),$$

$$\mu_{m,t} \leftarrow \mu_{m,t-1} - \epsilon_t \cdot \left(\frac{F}{T_m} - x_{m,t}\right).$$

The mini-batch algorithm for this specific target frequency can be derived in a similar way, where gradients are computed over all auction opportunities within each interval Δt, and the dual parameters are updated iteratively.

CHAPTER 8

EVEN BUDGET PACING

In this chapter, we demonstrate how to approach the even budget pacing problem. This problem arises when advertisers require their campaign budget to be distributed evenly across its lifetime. We show how to use the MPC controller to address this challenge.

1 Budget Pacing with Intra-Period Limits

Some advertisers prefer to pace their budget evenly over the campaign's duration rather than spending the majority within a short period. For example, in a daily pacing campaign, advertisers may set constraints to ensure that no more than 50% of the total daily budget is spent within a single hour. Similarly, in a lifetime pacing campaign, they may aim to prevent a significant portion of the budget from being spent within just one or two days.

Problem Formulation One approach to addressing this requirement is to impose a spending limit for each intra-period. Suppose we divide the campaign's lifetime I into N consecutive and mutually exclusive sub-intervals, denoted as $\{I_i\}_{i=1}^N$. The revised pacing problem can then be formulated as follows:

$$\begin{aligned} \max_{x_t \in \{0,1\}} \quad & \sum_{t=1}^{T} x_t \cdot r_t \\ \text{s.t.} \quad & \sum_{t=1}^{T} x_t \cdot c_t \le B, \\ & \sum_{t \in I_i} x_t \cdot c_t \le \sigma B, \quad i = 1, \ldots, N. \end{aligned} \tag{8.1}$$

where:

- B is the total budget of the campaign.
- T is the number of auction opportunities for this campaign.
- r_t is the estimated conversion rate for the t-th auction opportunity.
- c_t is the expected payment (cost) per impression for the t-th auction.
- x_t is a binary decision variable indicating whether the campaign wins the t-th auction opportunity.
- σ is the intra-period cap, ensuring that the spending within each period I_i does not exceed σB.

Under a standard second-price auction, x_t can be expressed as $\mathbb{1}_{\{b_t > c_t\}}$, where b_t is the bid amount per impression. The time intervals $\{I_i\}$ are often chosen to be of equal duration, satisfying $I_i \cap I_j = \emptyset$ for $i \ne j$ and $\bigcup_{i=1}^N I_i = I$.

Derivation of the Optimal Bid The Lagrangian is:

$$\begin{aligned} \mathcal{L}(x_t, \lambda, \lambda_i) &= \sum_{t=1}^{T} x_t \cdot r_t - \lambda \cdot \left(\sum_{t=1}^{T} x_t \cdot c_t - B \right) - \sum_{i=1}^{N} \left[\lambda_i \cdot \left(\sum_{t \in I_i} x_t \cdot c_t - \sigma \cdot B \right) \right] \\ &= \sum_{t=1}^{T} \left[x_t \cdot \left(r_t - \lambda \cdot c_t - \left(\sum_{i=1}^{N} \lambda_i \cdot \mathbb{1}_{\{t \in I_i\}} \right) \cdot c_t \right) + \lambda \cdot \frac{B}{T} + \left(\sum_{i=1}^{N} \lambda_i \right) \cdot \frac{\sigma B}{T} \right]. \end{aligned}$$

To maximize $\mathcal{L}(x_t, \lambda, \lambda_i)$, we set $x_t = 1$ whenever:

$$r_t - \lambda \cdot c_t - \left(\sum_{i=1}^{N} \lambda_i \cdot \mathbb{1}_{\{t \in I_i\}} \right) \cdot c_t > 0,$$

and $x_t = 0$ otherwise. The dual $\mathcal{L}^*(\lambda, \lambda_i)$ then becomes:

$$\mathcal{L}^*(\lambda, \lambda_i) = \sum_{t=1}^{T} \left[\underbrace{\left(r_t - \lambda \cdot c_t - \left(\sum_{i=1}^{N} \lambda_i \cdot \mathbb{1}_{\{t \in I_i\}} \right) \cdot c_t \right)_+ + \lambda \cdot \frac{B}{T} + \left(\sum_{i=1}^{N} \lambda_i \right) \cdot \frac{\sigma B}{T}}_{f_t(\lambda, \lambda_i)} \right].$$

Suppose we find a feasible solution to the dual:

$$\lambda, \lambda_i = \underset{\lambda \geq 0, \lambda_i \geq 0}{\arg\min}\, \mathcal{L}^*(\lambda, \lambda_i).$$

The optimal bid per impression is then given by:

$$b_t^* = \frac{r_t}{\lambda^* - \sum_{i=1}^{N} \lambda_i^* \cdot \mathbb{1}_{\{t \in I_i\}}}. \tag{8.2}$$

Interpretation of the Bidding Formula Since $I_i \cap I_j = \emptyset$ whenever $i \neq j$, each auction opportunity t belongs to exactly one interval, say I_i. The bid formula (8.2) is thus equivalent to:

$$b_t^* = \frac{r_t}{\lambda^* - \lambda_i^*}. \tag{8.3}$$

If the intra-period constraint for I_i is not active, meaning that the budget allocated based on the overall traffic pattern is less than $\sigma \cdot \frac{B}{T}$, then the corresponding dual parameter satisfies $\lambda_i^* = 0$. In this case, equation (8.3) simplifies to:

$$b_t^* = \frac{r_t}{\lambda^*},$$

which corresponds to the standard bid formula without an intra-period constraint.

On the other hand, if the intra-period constraint is active—meaning that $\sigma \cdot \frac{B}{T}$ imposes a stricter limit on I_i—then $\lambda_i^* > 0$, and the optimal bid b_t^* is lower than the bid level in the absence of the intra-period constraint, i.e., $\frac{r_t}{\lambda^*}$.

This pacing principle aligns with our intuition: when the new constraint for I_i is not active, the campaign follows the standard pacing strategy. However, if the constraint is active, the bid must be reduced to ensure compliance with the stricter intra-period spending limit.

Algorithm Design We can design the pacing algorithm using any of the approaches discussed in Part II. For example, if we adopt the DOGD framework, we can compute the gradient to derive the stochastic gradient descent update rule:

$$\lambda \leftarrow \lambda - \epsilon \cdot \frac{\partial}{\partial \lambda} f_t(\lambda, \lambda_i) = \lambda - \epsilon \cdot \left(\frac{B}{T} - c_t \cdot x_t \right),$$

$$\lambda_i \leftarrow \lambda_i - \epsilon \cdot \frac{\partial}{\partial \lambda_i} f_t(\lambda, \lambda_i) = \lambda_i - \epsilon \cdot \left(\sigma \cdot \frac{B}{T} - \mathbb{1}_{t \in I_i} \cdot c_t \cdot x_t \right).$$

Readers should already be familiar with the update rule for λ. For λ_i, the update occurs only within I_i, where the gradient is the difference between the target spend rate $\sigma \cdot \frac{B}{T}$ and the actual spend per impression $\mathbb{1}_{t \in I_i} \cdot c_t \cdot x_t$ in the i-th intra-period. This update rule provides the foundation for designing the corresponding DOGD-based pacing algorithm.

A more suitable framework for this problem is Model Predictive Control (MPC). Since each intra-period has its own constraint, MPC naturally fits as an adaptive receding horizon control approach. At the beginning of each pacing interval within I_i, say at time τ, the receding horizon control problem can be formulated as:

$$\begin{aligned} \max_{x_t \in \{0,1\}} \quad & \sum_{t \in I_\tau} x_t \cdot r_t \\ \text{s.t.} \quad & \sum_{t \in I_\tau} x_t \cdot c_t \leq \min\{B_\tau, B_{\tau,i}\}. \end{aligned} \tag{8.4}$$

where:

- I_τ represents the remaining time in I_i starting from τ.
- B_τ is the budget allocated for I_τ, derived from the remaining overall budget (normalized[1] for I_τ).
- $B_{\tau,i}$ is the remaining budget from the intra-period constraint, computed as $\sigma \cdot \frac{B}{T}$ minus the amount spent up to τ since the beginning of I_i.

The effective budget for I_τ is then adaptively set as the minimum of B_τ and $B_{\tau,i}$, ensuring compliance with both the overall and intra-period constraints for the campaign. This receding horizon control problem effectively reduces to a standard max delivery problem, allowing us to apply previously discussed techniques to determine the optimal bid for the next update.

We summarize the MPC approach in Algorithm 37.

2 Remarks

2.1 Throttling-based Approach

In practice, some traffic spikes may not be captured by prediction models (e.g., unusual traffic surges due to sudden events), and such fluctuations can occur within just a few seconds. Within such a short time frame, there is often no opportunity to adjust bids in response

[1]Here, we normalize the budget based on the duration of pacing intervals under the assumption that traffic is uniformly distributed. However, a more accurate approach would be to normalize the budget based on the actual traffic pattern.

Algorithm 37 MPC-Based Bidding Algorithm for Budget Pacing with Intra-Period Limits

```
Require:
 1: Total budget $B$, total time horizon $T$
 2: Intra-periods $\{I_i\}_{i=1}^{N}$, intra-period cap $\sigma$,
Ensure: Computes optimal bid per impression $b_t^*$
 3: for each pacing interval starting at $\tau$ do
 4:     Identify the current intra-period $I_i$ such that $\tau \in I_i$
 5:     Compute remaining time in $I_i$ from $\tau$ to the end of $I_i$, denoted as $I_\tau$
 6:     Observe budget spent from the beginning of $I_i$ up to $\tau$, denoted as $B_{\text{spent},i}$
 7:     Compute the remaining budget for the intra-period:
            $$B_{\tau,i} = \sigma B - B_{\text{spent},i}$$
 8:     Observe total remaining budget $B_{\text{remain}}$ and normalize it for $I_\tau$:
            $$B_\tau = \frac{|I_\tau|}{|I_{\text{remain}}|} B_{\text{remain}}$$
        where $|\cdot|$ denotes the duration of the interval, $I_{\text{remain}}$ is the remaining lifetime of this
        campaign
 9:     Compute the effective budget for $I_\tau$ as
            $$B_{\text{effective}} = \min\{B_{\tau,i}, B_\tau\}$$
10:     Solve the max-delivery problem (8.4) with effective budget $B_{\text{effective}}$ to get the optimal
        bid per conversion for $I_\tau$: $b_\tau^*$
11:     for each auction request $t$ in pacing interval $I_\tau$ do
12:         Get predicted conversion rate $r_t$ from the prediction model
13:         Compute bid per impression:
                $$b_t^* = b_\tau^* \cdot r_t$$
14:         Submit $b_t^*$ for auction request $t$
15:     end for
16: end for
```

to these traffic spikes—especially considering that the pacing interval is typically around 30 seconds to a few minutes. Consequently, the budget may be consumed much faster than expected or even be depleted within seconds.

In such cases, a bid-based pacing algorithm alone may not be sufficient to address the issue. One potential workaround is to employ the throttling techniques discussed earlier to temporarily prevent the campaign from participating in auctions.

The throttling-based method can serve as a safeguard for even pacing. For instance, consider the pacing formulation in (8.1). Every X seconds, we compute the total budget spent, B_s, since the beginning of I_i up to the current time. If B_s exceeds a predefined threshold (e.g., $0.8 \cdot \sigma \cdot B$), the throttling mechanism is triggered. In this case, the throttling probability increases as B_s approaches the budget limit $\sigma \cdot B$, thereby reducing participation in auctions to prevent overspending.

Such a throttling mechanism requires only the accumulated budget data of the campaign. For oCPM campaigns, these signals can be collected with negligible delays, making it highly suitable for second-level granularity control.

2.2 Comparison of Different Budget Allocation Patterns

We compare three different budget allocation strategies:

- **Even Pacing**: This strategy is similar to the one discussed in this chapter, where the budget is evenly distributed throughout the day. As shown in the leftmost plot of Figure 8.1, the budget spending targets remain constant within each hour.

- **Traffic-Based Pacing**: In this strategy, the budget is allocated based on traffic volume. This approach has been discussed throughout the book and is optimal under the assumption that conversion rates and supporting prices follow some i.i.d. distributions. The middle plot in Figure 8.1 illustrates this pattern. Since traffic volume is relatively low at night and higher during the day, the budget allocation forms a bell-shaped curve, peaking around noon.

- **Performance-Based Pacing**: Unlike the previous strategy, this approach does not assume an i.i.d. distribution. Instead, it considers variations in online traffic quality throughout the day when allocating the budget. As shown in the rightmost plot of Figure 8.1, the allocation curve now has two peaks—one around 6 AM and another around 6 PM—corresponding to periods when traffic exhibits higher CTRs compared to other times of the day.

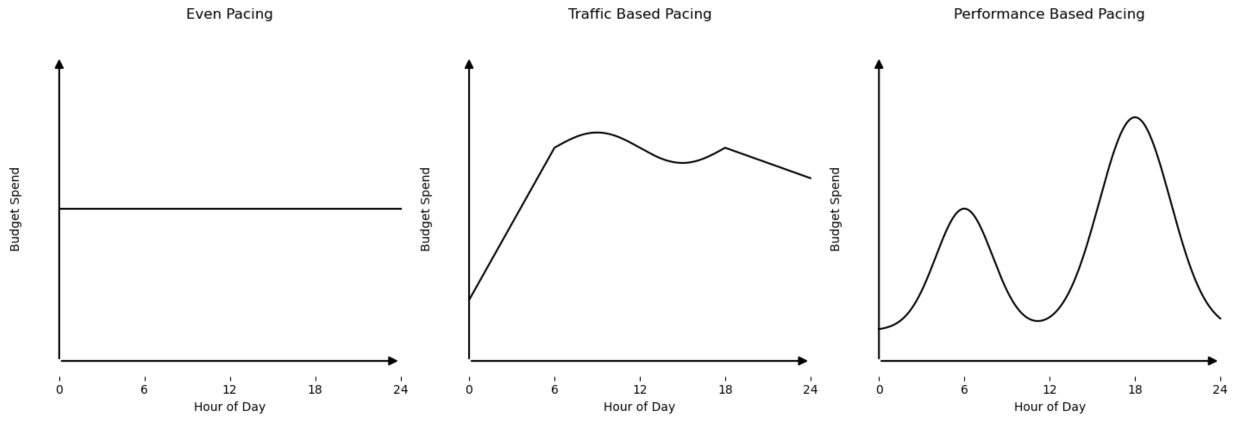


Figure 8.1: Different Budget Pacing Patterns

A detailed discussion of these budget allocation patterns, along with additional strategies, can be found in [43]. Readers interested in this topic may refer to this source for further technical details.

BIBLIOGRAPHY

[1] Deepak Agarwal, Souvik Ghosh, Kai Wei, and Siyu You. "Budget pacing for targeted online advertisements at LinkedIn". In: KDD '14 (2014), pp. 1613–1619.

[2] Gagan Aggarwal, Ashwinkumar Badanidiyuru, Santiago R Balseiro, Kshipra Bhawalkar, Yuan Deng, Zhe Feng, Gagan Goel, Christopher Liaw, Haihao Lu, Mohammad Mahdian, et al. "Auto-bidding and auctions in online advertising: A survey". In: *ACM SIGecom Exchanges* 22.1 (2024), pp. 159–183.

[3] Jonathan Amar and Nicholas Renegar. "The Second-Price Knapsack Problem: Near-Optimal Real Time Bidding in Internet Advertisement". In: *arXiv preprint arXiv:1810.10661* (2018).

[4] Lin An, Andrew A Li, Benjamin Moseley, and Gabriel Visotsky. "Best of Many in Both Worlds: Online Resource Allocation with Predictions under Unknown Arrival Model". In: *arXiv preprint arXiv:2402.13530* (2024).

[5] Miriam Ayer, H Daniel Brunk, George M Ewing, William T Reid, and Edward Silverman. "An empirical distribution function for sampling with incomplete information". In: *The annals of mathematical statistics* (1955), pp. 641–647.

[6] Santiago Balseiro and Yonatan Gur. "Learning in repeated auctions with budgets: Regret minimization and equilibrium". In: *Management Science* 65.9 (2019), pp. 3952–3968.

[7] Santiago Balseiro, Christian Kroer, and Rachitesh Kumar. "Online resource allocation under horizon uncertainty". In: *Abstract Proceedings of the 2023 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*. 2023, pp. 63–64.

[8] Santiago Balseiro, Haihao Lu, and Vahab Mirrokni. "Dual mirror descent for online allocation problems". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 613–628.

[9] Santiago R Balseiro, Haihao Lu, Vahab Mirrokni, and Balasubramanian Sivan. "Analysis of Dual-Based PID Controllers through Convolutional Mirror Descent". In: *arXiv e-prints* (2022), arXiv–2202.

[10] Guillaume W Basse, Hossein Azari Soufiani, and Diane Lambert. "Randomization and the pernicious effects of limited budgets on auction experiments". In: *Artificial Intelligence and Statistics*. PMLR. 2016, pp. 1412–1420.

[11] DP Bertsekas. "Neuro-dynamic programming". In: *Athena Scientific* (1996).

[12] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA, 1997.

[13] Vijay Bharadwaj, Peiji Chen, Wenjing Ma, Chandrashekhar Nagarajan, John Tomlin, Sergei Vassilvitskii, Erik Vee, and Jian Yang. "Shale: an efficient algorithm for allocation of guaranteed display advertising". In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 1195–1203.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[15] Yang Cai, Constantinos Daskalakis, and S Matthew Weinberg. "On optimal multidimensional mechanism design". In: *ACM SIGecom Exchanges* 10.2 (2011), pp. 29–33.

[16] Eduardo F Camacho and Carlos Bordons Alba. *Model Predictive Control*. en. 2nd ed. Advanced Textbooks in Control and Signal Processing. London, England: Springer, May 2004.

[17] George Casella and Roger Berger. *Statistical inference*. CRC press, 2024.

[18] Bowei Chen, Shuai Yuan, and Jun Wang. "A dynamic pricing model for unifying programmatic guarantee and real-time bidding in display advertising". In: *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. 2014, pp. 1–9.

[19] Yuanlong Chen, Bowen Zhu, Bing Xia, Yichuan Wang, Reza Madad, Xudong Zhang, and He Xiao. "Method, Device, and Medium for Placing Information". US20240403125. Accessed: 2025-01-12. Dec. 2024. URL: https://patents.justia.com/patent/20240403125.

[20] Zhaohua Chen, Chang Wang, Qian Wang, Yuqi Pan, Zhuming Shi, Zheng Cai, Yukun Ren, Zhihua Zhu, and Xiaotie Deng. "Dynamic budget throttling in repeated second-price auctions". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 9. 2024, pp. 9598–9606.

[21] Edward H Clarke. "Multipart pricing of public goods". In: *Public choice* (1971), pp. 17–33.

[22] Paul Covington, Jay Adams, and Emre Sargin. "Deep neural networks for youtube recommendations". In: *Proceedings of the 10th ACM conference on recommender systems*. 2016, pp. 191–198.

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

[24] Joyce Delnoij and Kris De Jaegher. "Competing first-price and second-price auctions". In: *Economic Theory* 69.1 (2020), pp. 183–216.

[25] Nikhil R Devanur, Kamal Jain, Balasubramanian Sivan, and Christopher A Wilkens. "Near optimal online algorithms and fast approximation algorithms for resource allocation problems". In: *Proceedings of the 12th ACM conference on Electronic commerce*. 2011, pp. 29–38.

[26] W Erwin Diewert. "Applications of duality theory". In: (1974).

[27] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. "Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords". In: *American economic review* 97.1 (2007), pp. 242–259.

[28] Hossein Esfandiari, Nitish Korula, and Vahab Mirrokni. "Online allocation with traffic spikes: Mixing adversarial and stochastic models". In: *Proceedings of the Sixteenth ACM Conference on Economics and Computation*. 2015, pp. 169–186.

[29] Zhen Fang, Yang Li, Chuanren Liu, Wenxiang Zhu, Yu Zheng, and Wenjun Zhou. "Large-scale personalized delivery for guaranteed display advertising with real-time pacing". In: *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2019, pp. 190–199.

[30] Jon Feldman, Nitish Korula, Vahab Mirrokni, Shanmugavelayutham Muthukrishnan, and Martin Pál. "Online ad assignment with free disposal". In: *International workshop on internet and network economics*. Springer. 2009, pp. 374–385.

[31] Joaquin Fernandez-Tapia, Olivier Guéant, and Jean-Michel Lasry. "Optimal real-time bidding strategies". In: *Applied mathematics research express* 2017.1 (2017), pp. 142–183.

[32] Yuan Gao, Kaiyu Yang, Yuanlong Chen, Min Liu, and Noureddine El Karoui. "Bidding agent design in the linkedin ad marketplace". In: *arXiv preprint arXiv:2202.12472* (2022).

[33] Aritra Ghosh, Saayan Mitra, Somdeb Sarkhel, Jason Xie, Gang Wu, and Viswanathan Swaminathan. "Scalable bid landscape forecasting in real-time bidding". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer. 2019, pp. 451–466.

[34] Nathalie Gimenes and Emmanuel Guerre. "Quantile regression methods for first-price auctions". In: *Journal of Econometrics* 226.2 (2022), pp. 224–247.

[35] Djordje Gligorijevic, Tian Zhou, Bharatbhushan Shetty, Brendan Kitts, Shengjun Pan, Junwei Pan, and Aaron Flores. "Bid shading in the brave new world of first-price auctions". In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 2453–2460.

[36] Theodore Groves. "Incentives in teams". In: *Econometrica: Journal of the Econometric Society* (1973), pp. 617–631.

[37] Jason D Hartline. "Mechanism design and approximation". In: *Book draft. October* 122.1 (2013).

[38] Elad Hazan et al. "Introduction to online convex optimization". In: *Foundations and Trends® in Optimization* 2.3-4 (2016), pp. 157–325.

[39] Edward L Kaplan and Paul Meier. "Nonparametric estimation from incomplete observations". In: *Journal of the American statistical association* 53.282 (1958), pp. 457–481.

[40] Brendan Kitts, Michael Krishnan, Ishadutta Yadav, Yongbo Zeng, Garrett Badeau, Andrew Potter, Sergey Tolkachov, Ethan Thornburg, and Satyanarayana Reddy Janga. "Ad serving with multiple KPIs". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 1853–1861.

[41] Roger Koenker and Kevin F Hallock. "Quantile regression". In: *Journal of economic perspectives* 15.4 (2001), pp. 143–156.

[42] Kevin J Lang, Benjamin Moseley, and Sergei Vassilvitskii. "Handling forecast errors while bidding for display advertising". In: *Proceedings of the 21st international conference on World Wide Web*. 2012, pp. 371–380.

[43] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. "Real time bid optimization with smooth budget delivery in online advertising". In: *Proceedings of the seventh international workshop on data mining for online advertising*. 2013, pp. 1–9.

[44] Jan de Leeuw, Kurt Hornik, and Patrick Mair. "Isotone optimization in R: pool-adjacent-violators algorithm (PAVA) and active set methods". In: (2009).

[45] Min Liu, Jialiang Mao, and Kang Kang. "Trustworthy online marketplace experimentation with budget-split design". In: *arXiv preprint arXiv:2012.08724* (2020).

[46] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. "Cascade ranking for operational e-commerce search". In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017, pp. 1557–1565.

[47] Alejandro M Manelli and Daniel R Vincent. "Multidimensional mechanism design: Revenue maximization and the multiple-good monopoly". In: *Journal of Economic theory* 137.1 (2007), pp. 153–185.

[48] Alberto Marchetti-Spaccamela and Carlo Vercellis. "Stochastic on-line knapsack problems". In: *Mathematical Programming* 68.1 (1995), pp. 73–104.

[49] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[50] Steven A Matthews. *A technical primer on auction theory I: Independent private values*. Tech. rep. Discussion paper, 1995.

[51] Jacob Mattingley, Yang Wang, and Stephen Boyd. "Code generation for receding horizon control". In: *2010 IEEE international symposium on computer-aided control system design*. IEEE. 2010, pp. 985–992.

[52] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. "Adwords and generalized online matching". In: *Journal of the ACM (JACM)* 54.5 (2007), 22–es.

[53] Roger B Myerson. "Optimal auction design". In: *Mathematics of operations research* 6.1 (1981), pp. 58–73.

[54] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani, eds. *Algorithmic game theory*. Cambridge, England: Cambridge University Press, Sept. 2007.

[55] Shengjun Pan, Brendan Kitts, Tian Zhou, Hao He, Bharatbhushan Shetty, Aaron Flores, Djordje Gligorijevic, Junwei Pan, Tingyu Mao, San Gultekin, et al. "Bid shading by win-rate estimation and surplus maximization". In: *arXiv preprint arXiv:2009.09259* (2020).

[56] Kan Ren, Jiarui Qin, Lei Zheng, Zhengyu Yang, Weinan Zhang, and Yong Yu. "Deep landscape forecasting for real-time bidding advertising". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 363–372.

[57] John A Rice. *Mathematical statistics and data analysis*. Vol. 371. Thomson/Brooks/-Cole Belmont, CA, 2007.

[58] Herbert Robbins and Sutton Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[59] Tim Roughgarden. *Twenty lectures on algorithmic game theory*. Cambridge University Press, 2016.

[60] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[61] Hal R Varian. "Position auctions". In: *international Journal of industrial Organization* 25.6 (2007), pp. 1163–1178.

[62] Hal R Varian and Christopher Harris. "The VCG auction in theory and practice". In: *American Economic Review* 104.5 (2014), pp. 442–445.

[63] William Vickrey. "Counterspeculation, auctions, and competitive sealed tenders". In: *The Journal of finance* 16.1 (1961), pp. 8–37.

[64] Jun Wang, Weinan Zhang, Shuai Yuan, et al. "Display advertising with real-time bidding (RTB) and behavioural targeting". In: *Foundations and Trends® in Information Retrieval* 11.4-5 (2017), pp. 297–435.

[65] Yuan Wang, Peifeng Yin, Zhiqiang Tao, Hari Venkatesan, Jin Lai, Yi Fang, and PJ Xiao. "An empirical study of selection bias in pinterest ads retrieval". In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 5174–5183.

[66] Yuchen Wang, Kan Ren, Weinan Zhang, Jun Wang, and Yong Yu. "Functional bid landscape forecasting for display advertising". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part I 16*. Springer. 2016, pp. 115–131.

[67] Laurence A Wolsey and George L Nemhauser. *Integer and combinatorial optimization*. John Wiley & Sons, 1999.

[68] Wush Wu, Mi-Yen Yeh, and Ming-Syan Chen. "Deep censored learning of the winning price in the real time bidding". In: *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*. 2018, pp. 2526–2535.

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

[70] Yadong Xu, Bonan Ni, Weiran Shen, Xun Wang, Zichen Wang, Yinsong Xue, and Pingzhong Tang. "Simultaneous Optimization of Bid Shading and Internal Auction for Demand-Side Platforms". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. 9. 2024, pp. 9935–9943.

[71] Xun Yang, Yasong Li, Hao Wang, Di Wu, Qing Tan, Jian Xu, and Kun Gai. "Bid optimization by multivariable control in display advertising". In: *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2019, pp. 1966–1974.

[72] Smirnov Yury, Lu Quan, and Lee Kuang-chih. "Online Ad Campaign Tuning with PID Controllers". US20160110755A1. Apr. 21, 2016. URL: https://patents.google.com/patent/US20160110755A1/en.

[73] Haoqi Zhang, Junqi Jin, Zhenzhe Zheng, Fan Wu, Haiyang Xu, and Jian Xu. "Control-based Bidding for Mobile Livestreaming Ads with Exposure Guarantee". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 2539–2548.

[74] Wei Zhang, Brendan Kitts, Yanjun Han, Zhengyuan Zhou, Tingyu Mao, Hao He, Shengjun Pan, Aaron Flores, San Gultekin, and Tsachy Weissman. "Meow: A space-efficient nonparametric bid shading algorithm". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 3928–3936.

[75] Weinan Zhang, Yifei Rong, Jun Wang, Tianchi Zhu, and Xiaofan Wang. "Feedback control of real-time display advertising". In: *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*. 2016, pp. 407–416.

[76] Tian Zhou, Hao He, Shengjun Pan, Niklas Karlsson, Bharatbhushan Shetty, Brendan Kitts, Djordje Gligorijevic, San Gultekin, Tingyu Mao, Junwei Pan, et al. "An efficient deep distribution network for bid shading in first-price auctions". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 3996–4004.

[77] Yunhong Zhou, Deeparnab Chakrabarty, and Rajan Lukose. "Budget constrained bidding in keyword auctions and online knapsack problems". In: *Proceedings of the 17th international conference on world wide web*. 2008, pp. 1243–1244.

[78] Wen-Yuan Zhu, Wen-Yueh Shih, Ying-Hsuan Lee, Wen-Chih Peng, and Jiun-Long Huang. "A gamma-based regression for winning price estimation in real-time bidding advertising". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE. 2017, pp. 1610–1619.

[79] Martin Zinkevich. "Online convex programming and generalized infinitesimal gradient ascent". In: *Proceedings of the 20th international conference on machine learning (icml-03)*. 2003, pp. 928–936.