A Pratical Guide to Budget Pacing Algorithms in Digital Advertising

For Engineers

Yuanlong Chen

PREFACE

A typical real-time ad-serving funnel comprises ad targeting, conversion (e.g. click through rate) modeling, 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—remains underexplored in existing literature. This book aims to provide engineers with a practical yet 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, CPC, 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 YouTube or Instagram), 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, online adaptive optimal control, and deep reinforcement learning. For each approach, we explain the motivation, introduce the basic background, and describe how it can be applied to bidding 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.

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

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Part I

Some Basics

CHAPTER 1

A BRIEF HISTORY OF DIGITAL ADS

Brief History Intro

Introduction

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 like Google Ads and Facebook Ads Manager. This chapter explores the history of digital advertising, with a particular focus on the rise of RTB and its integration into a broader programmatic ecosystem.

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 start of online monetization. During this period, advertisers purchased ad space directly from publishers, with little 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 began replacing 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 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 Advent 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:** Launch of DoubleClick Ad Exchange by Google, introducing the first large-scale RTB platform.
- **2011:** Facebook introduced its ad 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

Major platforms like Google and Facebook have developed their own proprietary bidding systems to capitalize on their vast user bases and data resources:

Google Ads

Google Ads integrates RTB capabilities into its ecosystem, allowing advertisers to bid on search, display, and video ads. Its advanced targeting and optimization features make it a dominant force in digital advertising.

Facebook Ads Manager

Facebook introduced FBX to integrate RTB into its platform. Although FBX was later retired, Facebook Ads Manager continues to provide advertisers with precise audience targeting and real-time campaign optimization.

Challenges and Innovations in Digital Advertising

While digital advertising has achieved remarkable success, it faces ongoing challenges such as data privacy concerns, ad fraud, and transparency issues. Regulations like GDPR and CCPA have reshaped the industry, emphasizing the need for responsible data usage. Simultaneously, emerging technologies such as artificial intelligence and blockchain are driving innovation, addressing these challenges, and enhancing efficiency.

Conclusion

The history of digital advertising reflects a journey of continuous innovation, from static banners to dynamic, data-driven systems like RTB. Understanding this evolution highlights the transformative impact of programmatic technologies and the importance of platforms like DSPs, SSPs, and in-house solutions from tech giants like Google and Facebook. As digital advertising continues to evolve, RTB remains a cornerstone, shaping the future of how brands connect with audiences.

CHAPTER 2

ADVERTISING BASICS

Basic concepts in digital advertising

CHAPTER 3

BIDDING PRODUCTS

Introduction to some popular bidding products

CHAPTER 4

OPTIMIZATION TECHNIQUES

Some basics of optimization techniques

[5] [4]

CHAPTER 5

AUCTION MECHANISMS

Introduction to some popular auction mechanisms

[**roughgarden2010algorithmic**]

CHAPTER 6

EXPERIMENT FRAMEWORK

Introduction to A/B testing framework in advertisering business

1 Campaign Level A/B Test

normal A/B test framework

2 Budget Split A/B Test

paper ref: [14]

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.

1 Max Delivery

In the Max Delivery setting, advertisers set up a campaign with a specified budget B. The objective is to optimize the performance of the ad campaign while adhering to this budget constraint. Assuming this is a CPC (Cost-Per-Click) daily 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:

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

Here, T represents the total number of auction opportunities within a day. For the t-th auction:

- r_t is the predicted click-through rate (CTR).
- c_t denotes the cost (in a second-price auction, this corresponds to the highest effective CPM [eCPM]).
- x_t is a decision variable indicating whether we win the 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:

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

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.

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:

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

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 $\mathcal{L}(x, \lambda)$ as:

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

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

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

For more technical details, one may refer to [13] and [18].

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 B, the advertiser defines a cost cap C, 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 a CPC daily campaign can be formulated as follows:

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

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

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:

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

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 $\alpha \to 1$ (i.e., $\mu^* \to 0$), the cost constraint becomes invalid, and $b^*_{\text{click}} \to 1/\lambda^*$, reducing the problem to the max delivery problem. Conversely, when $\alpha \to 0$ (i.e., $\mu^* \to \infty$), the budget constraint becomes invalid, and $b^* \to C$, meaning the campaign will bid at the maximum level allowed under the cost constraint, i.e., the cost cap.

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

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.

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

Initialize $p(0) \leftarrow 1.0$ and $S(0) \leftarrow 0.0$
for each auction at t-th auction **do**
    Calculate target spend: $target\_spend\alpha(t) \leftarrow \frac{t}{T} \times B$
    if $S(t) \leq \alpha(t)$ **then**
        Increase throttling probability: $p(t) \leftarrow \min\{1.0, p(t) \cdot (1 + \lambda_t\}$
    else
        Decrease throttling probability: $p(t) \leftarrow \max\{0.0, p(t) \cdot (1 - \lambda_t\}$
    end if
    Generate a random number $r \in [0, 1]$
    if $r \leq p(t)$ **then**
        Participate in the auction and get the spend in current auction c_t
        Update spend: $S(t) \leftarrow S(t-1) + c_t$
    else
        Skip the auction
    end if
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 [6], which also includes a comparison between throttle-based pacing and bid-based pacing, both of which we will introduce in the subsequent sections.

Throttling for Cost Cap Campaign

More technical details on cost cap throttling [21]

CHAPTER 3

PID CONTROLLER

PID controller

1 Introduction to PID Controllers

A Proportional-Integral-Derivative (PID) controller is a widely used feedback control mechanism in industrial and engineering systems. 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 primary motivation for using a PID controller is its ability to regulate dynamic systems efficiently by balancing fast response, minimal steady-state error, and robustness to disturbances. PID controllers are versatile and can be tuned to meet specific performance requirements in diverse applications, such as temperature control, motor speed regulation, and process automation.

Mathematical Details

The output of a PID controller, $u(t)$, is given by:

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

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.

Components of a PID Controller

1. **Proportional Term**: $K_p e(t)$ provides an immediate response proportional to the current error. However, it may not fully eliminate the steady-state error.
2. **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.
3. **Derivative Term**: $K_d \frac{de(t)}{dt}$ predicts future errors by responding to the rate of change of the error, improving stability and damping oscillations.

PID controllers are simple to implement, robust, and effective for a wide range of systems. With proper tuning of K_p, K_i, and K_d, they can balance speed, stability, and accuracy in dynamic environments.

2 PID Contoller in Max Delivery

Main Algorithm

In section 1, 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.

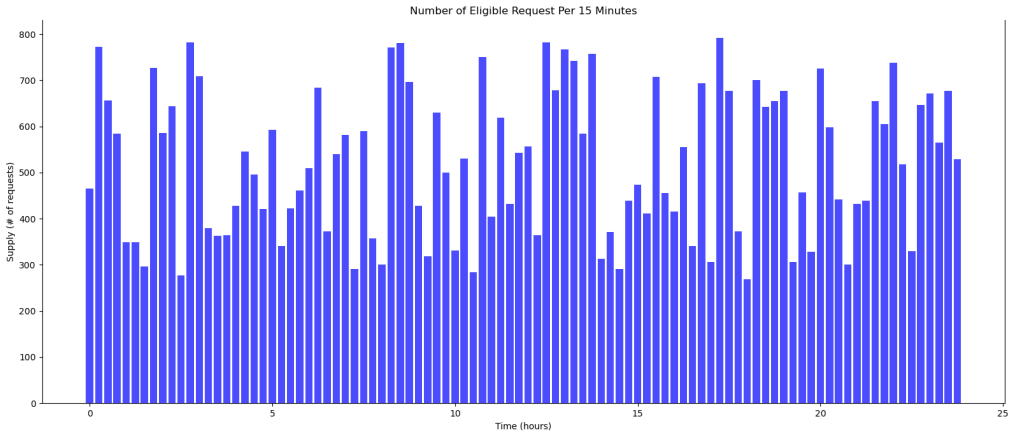

Figure 3.1: Supply Pattern (Number of Eligible Requests Per 15 Minutes)

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:

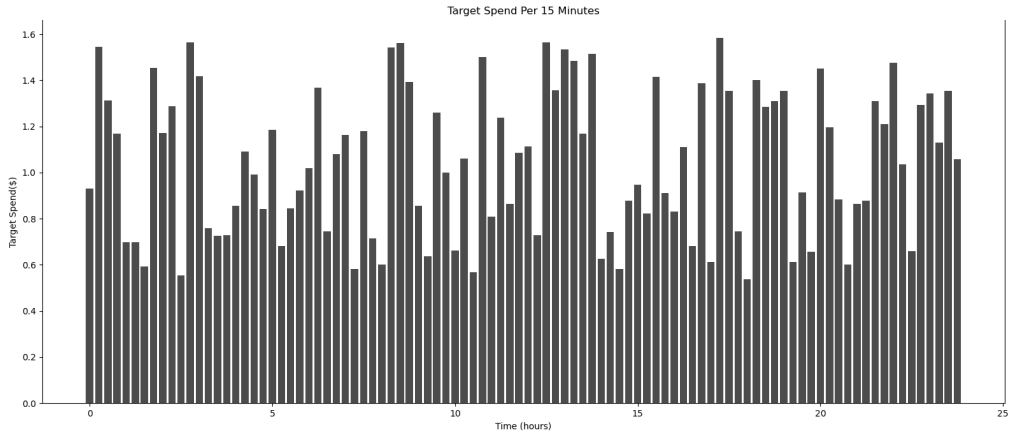

Figure 3.2: Target Spend Per 15 Minutes

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 $t = 1, \ldots, T$
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 $t = 1, \ldots, T$

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 [18] and [25].

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.

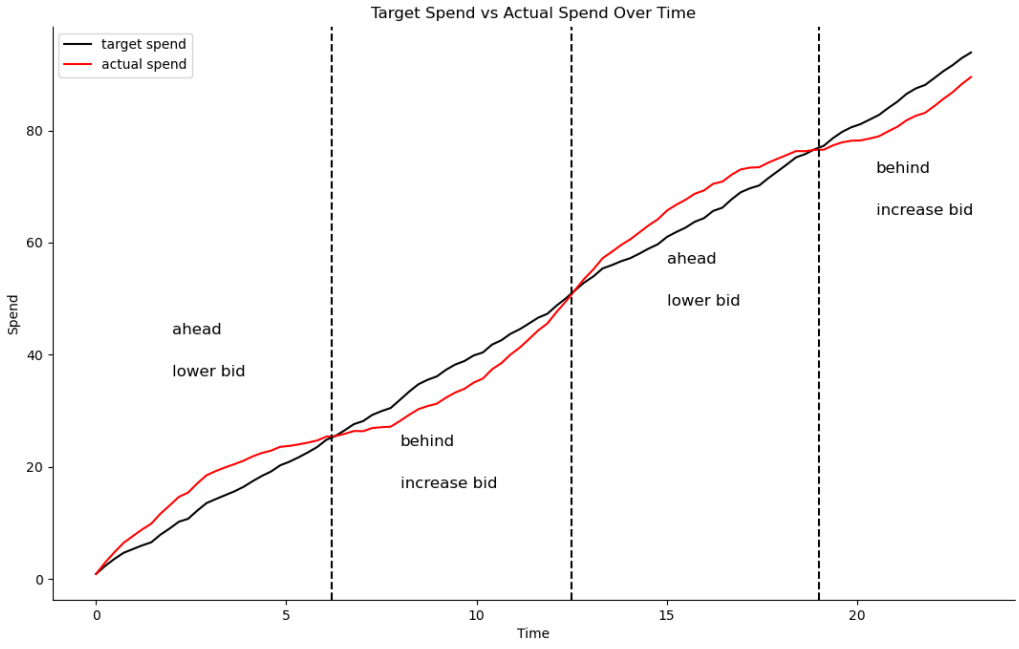

Figure 3.3: Target Spend vs Actual Spend Over Time

Algorithm 3 PID Controller for Bid Price Modulation in Max Delivery

Require: Δt: 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

Initialize $b(t_0) \leftarrow$ initial_bid
Initialize $u(t_0) \leftarrow 0$
Initialize cumulative_error $\leftarrow 0$
Initialize previous_error $\leftarrow 0$
for $k = 1$ to N **do** ▷ $N = \text{MinutesInOneDay}/\Delta t$
    Find the t-th target budget interval that contains t_k update interval, compute:

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

    Measure observed spend during the interval:

$$r(t_k) \leftarrow \text{observed\_spend}(t_k)$$

    Compute the error factor:

$$e(t_k) \leftarrow r(t_k) - s(t_k)$$

    Update the control signal using the PID formula:

$$u(t_k) \leftarrow K_p \cdot e(t_k) + K_i \cdot \text{cumulative\_error} + K_d \cdot \frac{e(t_k) - \text{previous\_error}}{\Delta t}$$

    Update the cumulative error:

$$\text{cumulative\_error} \leftarrow \text{cumulative\_error} + e(t_k) \cdot \Delta t$$

    Update the bid price:

$$b(t_k) \leftarrow b(t_{k-1}) \cdot \exp(u(t_k))$$

    Update the previous error:

$$\text{previous\_error} \leftarrow e(t_k)$$

    Actuate the new bid price:

$$\text{submit\_bid}(b(t_k))$$

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.

Practical Considerations

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

- **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.

- **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.

3 PID Contoller in Cost Cap

cost-min paper: [11]

CHAPTER 4

MPC CONTROLLER

MPC controller

1 MPC Controller for Max Delivery

2 MPC Controller for Cost Cap

paper: [23]

CHAPTER 5

DUAL ONLINE GRADIENT DESCENT

Online Gradient Descent in dual space

1 Max Delivery

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:

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

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 4:

Algorithm 4 **D**ual **O**nline **G**radient **D**escent(**DOGD**) for Max Delivery

Require: B: Total budget, T: Predicted total number of auction opportunities, ϵ_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 λ_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 5:

Algorithm 5 Mini-Batch DOGD for Max Delivery Problem

Require: B: Total budget, T: Predicted total number of auction opportunities, Δt: Mini-batch update interval, ϵ_t: Step size schedule

Ensure: Optimal dual variable λ_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 Δt **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 [2], [3], and [7].

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

2 Cost Cap

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 6 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, ϵ_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 λ_t and μ_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:

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

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 7 Mini-Batch DOGD for Cost Cap Problem

Require: B: Total budget, C: Target cost per click (CPC), T: Predicted total number of auction opportunities, Δt: Mini-batch update interval, ϵ_t: Step size schedule

Ensure: Optimal dual variables λ_t, μ_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 Δt **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 [7].

Practical Considerations

CHAPTER 6

DEEP REINFORCEMENT LEARNING

Deep Reinforcement Learning in auto-bidding

Part III

Miscellaneous

CHAPTER 1

INITIALIZATION OF CAMPAIGN BID

Initial Bid

cold start problem: [12] [17]

1 Parametric Approach

2 Non-Parametric Approach

CHAPTER 2

BID SHADING

1 Bid Shading via Surplus Maximization

point estimation: [9]

parametric estimations:

- sigmoid: [15]
- gamma: [27]
- gaussion: [20]
- gumbel: [19]
- lognormal: [26]
- mixture gaussin: [8]

non-parametric estimations:

- [10]
- [24]

deep learning method: [16] other: [22]

CHAPTER 3

MULTI-CHANNEL DELIVERY

Multi-Channel Delivery problem.

CHAPTER 4

CAMPAIGN GROUP OPTIMIZATION

Campaign Group Optimization

CHAPTER 5

DEEP CONVERSION

Deep conversion(retention) optimization.

CHAPTER 6

REACH AND FREQUENCY

Brand ads, reach and frequency

CHAPTER 7

OVER-DELIVERY PROBLEM

Over Delivery

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] 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.

[3] Santiago R Balseiro and Yonatan Gur. "Learning in repeated auctions with budgets: Regret minimization and equilibrium". In: *Management Science* 65.9 (2019), pp. 3952–3968.

[4] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to linear optimization*. Vol. 6. Athena Scientific Belmont, MA, 1997.

[5] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[6] 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.

[7] 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).

[8] 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.

[9] 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.

[10] Edward L Kaplan and Paul Meier. "Nonparametric estimation from incomplete observations". In: *Journal of the American statistical association* 53.282 (1958), pp. 457–481.

[11] 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.

[12] Tor Lattimore and Csaba Szepesvári. *Bandit algorithms*. Cambridge University Press, 2020.

[13] 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.

[14] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani, eds. *Algorithmic game theory*. Cambridge, England: Cambridge University Press, Sept. 2007.

[15] 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).

[16] 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.

[17] Daniel J Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. "A tutorial on thompson sampling". In: *Foundations and Trends® in Machine Learning* 11.1 (2018), pp. 1–96.

[18] 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.

[19] 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.

[20] 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.

[21] Jian Xu, Kuang-chih Lee, Wentong Li, Hang Qi, and Quan Lu. "Smart pacing for effective online ad campaign optimization". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 2015, pp. 2217–2226.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.