Analysis of a Learning Based Algorithm for Budget Pacing

MohammadTaghi Hajiaghayi
University of Maryland
College Park, USA
hajiagha@umd.edu

Max Springer
University of Maryland
College Park, USA
mss423@umd.edu

ABSTRACT

In this paper, we analyze a natural learning algorithm for uniform pacing of advertising budgets, equipped to adapt to varying ad sale platform conditions. On the demand side, advertisers face a fundamental technical challenge in automating bidding in a way that spreads their allotted budget across a given campaign subject to hidden, and potentially dynamic, cost functions. This automation and calculation must be done in runtime, implying a necessarily low computational cost for the high frequency auction rate. Advertisers are additionally expected to exhaust nearly all of their sub-interval (by the hour or minute) budgets to maintain budgeting quotas in the long run. To resolve this challenge, our study analyzes a simple learning algorithm that adapts to the latent cost function of the market and learns the optimal *average* bidding value for a period of auctions in a small fraction of the total campaign time, allowing for smooth budget pacing in real-time. We prove our algorithm is robust to changes in the auction mechanism, and exhibits a fast convergence to a stable average bidding strategy. The algorithm not only guarantees that budgets are nearly spent in their entirety, but also smoothly paces bidding to prevent early exit from the campaign and a loss of the opportunity to bid on potentially lucrative impressions later in the period.

In addition to the theoretical guarantees, we validate our algorithm with experimental results from open source data on real advertising campaigns to further demonstrate the effectiveness of our proposed approach.

KEYWORDS

budget pacing, auction systems, mechanism design, learning algorithm

1 INTRODUCTION

Ad impressions sold through real-time bidding (RTB) auctions are responsible for an ever-increasing portion of company expenditures as well as the revenue of large ad providers like Google, Facebook, Amazon, Microsoft, and Yahoo!. Google alone is responsible for

upwards of 50 billion ad impressions on average per day [21], with a corresponding revenue on the order of $100 million. While companies bidding in these advertising campaigns do not participate in every auction throughout the day, they are often required to participate enough to spend their allotted budget in this time. Thus, great interest is placed on adequately pacing budgets throughout the day so as to not spend too hastily at the beginning of a day and miss out on better impressions, or spend too frugally until the close of day.

We here take the perspective of a demand-side platform (DSP), an intermediary serving as the interface connecting advertisers with ad-exchanges and offering customized bid strategies in online auctions. Our model of a DSP is given a users budget and target spent amount, and is asked to uniformly pace spending so as to exhaust this amount.

In optimizing bids within an RTB, an advertiser hopes to maximize key performance indicators, such as cost per conversion (CPC) or click through rate (CTR), for an ad campaign subject to the budgeting constraints for each time period [7, 13, 16, 24]. By smoothly spreading a budget across the duration of a campaign, an advertiser can ensure sustained influence across the broad spectrum of ad impressions with which they interact. We can summarize the objective of an advertiser using a DSP as follows:

- **Achieve performance goals:** in an advertising campaign, performance goals are typically meant to ensure a broad audience is reached while maintaining a low CPC, ie. the investment required to acquire new customers is relatively small.
- **Optimally pace the budget:** an advertiser hopes to have a sustained influence in an ad campaign, smoothly rolling out their advertisements to ensure impact on users. This budget pacing can be uniform (equal representation throughout a campaign) or potentially more sophisticated so as to maximize the number of users reached during certain times of the day, while placing less priority on other periods.

In spite of the ever increasing prevalence of online advertising in modern society and the clear value of optimizing the above objectives in practice, the research literature on the problem instance of optimally spending budgets in their entirety is limited. The bulk of the work either excludes comprehensive analysis of the stability of a proper budget pacing mechanism in favor of simulating these results [7, 15, 23], or does not impose the realistic constraints that are met in the advertising economy [11, 16]. As such, we bring an analytic approach in combination with simulated results on real datasets of our pacing mechanism to bridge this gap in the literature.

1.1 Main Ideas

In this paper, we present an online approach to smoothly and uniformly pacing an advertiser's budget over a fixed-length advertising campaign. Our approach invokes an iterative control feedback mechanism to estimate the proper *average* bid to submit over each time period, which is later manipulated by a mechanism (potentially hidden to the bidder) to compute an "actual spent amount" for that period so that the total budget is approximately spent in a uniform fashion throughout the campaign. The algorithm relies on simplistic scaling of bids in response to learning of this latent mechanism in a naturalistic way, and is currently in implementation at two major companies with which the authors were previously affiliated. Specifically, we analytically derive the stability and convergence time of this natural learning algorithm that has analogously appeared in other online bidding literature [15, 24]. Though simple, this algorithm exhibits low convergence times for the problem of budget pacing and possesses interesting dynamics in accordance with other one-dimensional mappings like the logistic map. We summarize our main results informally:

- Develop an iterative control feedback algorithm for smart pacing of an advertisers budget through an ad campaign.
- Analytically derive the efficiency of this algorithm in quickly converging to an optimal bid strategy to ensure budgeting quotas are met.
- Implement our algorithm with extensive simulations on real ad campaigns from Yahoo! to demonstrate its feasibility in practice.

The remainder of the paper is organized as follows. In Section 2, we discuss the budgeting problem on a high level and previous work in the field. In Section 3, we describe our model and give a comprehensive analysis of the convergence dynamics exhibited. We discuss extensions of our algorithm to account for other realistic budgeting schemes to be implemented by a DSP in Section 4, and finally present simulations of our algorithm on real datasets in Section 5 to demonstrate its utility in practice. We conclude with a discussion of our methodology, its limitations, and where future work will be directed.

2 BACKGROUND AND RELATED WORK

In this section, we discuss the problem of budget pacing with the joint problem of bid optimization, and subsequently discuss the related work, noting where these results fall short of the reality of RTB schemes.

2.1 Problem Statement

We consider *online bid optimization* in the following framework: there are T auction periods ordered by an index $t \in \{1, ..., T\}$. At the start of an auction period, an individual advertiser must make a decision as to how much to bid for desired ads within this period, denoted by b_t. Advertisers further have a total daily budget $B \in \mathbb{R}$ that limits the amount that can be spent within the day. Typically, advertisers would like to have smooth budget delivery [1, 3–5, 9, 15, 23], expressed as not buying more than a set fraction of the impressions before a given time in order to ensure that:

(1) budgets are not prematurely expended, thus resulting in missed opportunities later in the day

(2) spending does not fluctuate substantially for ease of analysis.

As a simple example, assume that each t denotes a 2 minute ad period on a streaming website. Within this time frame, advertiser A may choose to bid with value b_t in auctions against all advertisers interested in showing an ad to a given subset of users. Within a split-second, the highest bidder for each user is chosen and A must pay for the auctions they won, incurring some cost

$$c_t = b_t \times \text{(number of auctions won)}.$$

Following observation of this incurred cost, the advertiser A invokes an algorithm to adjust their bidding in an effort to ensure a smooth depletion of their budget throughout the day.

We note that for each impression, the base bid b_t can be rescaled in a manner corresponding to the value of the impression for that advertiser. For example: as formulated in [18], the bid value of the i-th impression is rescaled by the probability of its conversion p_i, divided by the average conversion (denoted $\overline{p}$). This is often done for individual impressions of great interest to an advertiser, however, in our formulation, b_t is submitted *on average* in each auction period.

The added complication of the smooth delivery problem we focus on in the present study is that oftentimes, when an advertiser submits a bid in an auction, this may not be the true value they pay for that impression. As demonstrated in the simple example above, although A submits a bid b_t, they actually incur a cost that is potentially much larger (proportional to the number of auctions won).

In response, the pacing of an advertiser's budget relies on learning the actual cost of submitted bids to adequately scale bid values to meet desired spending goals in each period.

We can now formally frame the budget pacing problems as:

$$\textbf{minimize } B - \sum_{t=1}^{T} c_t \tag{1}$$

$$\textbf{subject to } \left| \frac{B}{T} - c_t \right| \leq \epsilon \tag{2}$$

for some $\epsilon \ll 1$ and where the optimization is framed using the information for all the auctions, however, the problem itself is online. As a result, it is clear we need an algorithm that quickly learns the latent cost function, $f(b_t) = c_t$, in a small portion of the total number of auctions and subsequently uniformly paces their bidding for the duration of the campaign. Of crucial importance, we note that the cost function in each period implicitly contains information on the number of impressions won and overall surplus generated based on the value of impressions won, subject to the payment rules of whichever auction mechanism is implemented for those impressions.

Lastly, it is important to note that the cost function, f, is *subject to change* at different times in a campaign and can be adjusted by the advertiser when manipulating their target spend amount for different times. For example, if the algorithm is being utilized to exhaust a daily budget, the advertiser may emphasize impressions in the morning and input a higher target spend amount for that portion of the day, while reducing this target for later times in the day. With slight modifications of our algorithm we can further capture these instances (see Section 4).

2.2 Related Work

Real-time bidding strategies have an expansive literature [8, 10, 12, 14, 17, 22], however the field of optimal budget pacing is relatively new and, as such, the literature thus far is often limited in scope [1, 4–6]. We here examine the methodologies of a select number of other studies whose work closely aligns with our own, the aspects of real-time bidding they encompass, and where they fall short.

Most closely related to our work is that of Lee et al. [15], which devises a comparable algorithm for pacing the budget while also maximizing the performance indicators of impressions. In particular, their algorithm scales bids between successive rounds to ensure smooth delivery of the budget without early exit. However, this paper maximizes impression value by estimating a threshold for advertisers to begin bidding. As a result, the algorithm bids more highly on the valuable impressions, but risks overspending and early exit from the campaign in these instances. Our algorithm instead ensures that agents are present throughout the entire campaign so that no impression is missed out on and guarantees sustained presence in an ad space. Additionally, the work of Lee et al. does not account for the real-world quotas that advertisers must meet, forcing them to spend their budgets in close to their entirety. Our algorithm accounts for this constraint and utilizes it in the scaling of bid values.

The work done by Xu et al. [23] also aligns well with our study, emphasizing smooth delivery of budget and minimizing the probability of early exit from a campaign to ensure a sustained presence for advertisers. When omitting performance goals, their algorithm is comparable to ours as it optimizes bidding by scaling bids based on the discrepancy between desired spending amount and the actuality. Additionally, they optimize over a fixed penalty function, similar to our latent cost function. However, their algorithm is not robust to changes in this penalty over time and fails to provide analytic results on the convergence and stability of this algorithm. Our study attempts to bridge the gap between the analytic and computational studies on optimal budget pacing. Lastly, this paper fails to consider the real-world constraint of budgeting quotas.

Fernandez-Tapia [11] examines the pacing problem from different perspective to the above, deriving analytical results through variational calculus techniques. This work compares the optimal bidding strategy when RTBs occur at a linear rate (fixed spacing between bids) and nonlinear. As most papers, including our own, make the simplifying assumption that auctions occur at such a high rate that we need not consider unequal arrival of the impressions to bid on, this more technical study highlights a more complex and realistic setting of auctions. However, while impressions may arrive in this more complicated fashion, advertisers do not in practice base their strategies around this assumption. As such, our algorithm is better suited for real-world implementation.

The problem of finding an optimal bidding strategy for an advertiser who does not know their own valuation, subject to budgeting constraints, has also been studied as an extension of the multi-armed bandit problem as in [3]. Avadhanula et al. provide bidding schemes over discrete and continuous bid spaces for m platforms simultaneously with regret lower bounds of $\Omega\left(\sqrt{mOPT}\right)$ and $\Omega\left(m^{1/3}B^{2/3}\right)$ – where B is the budget and OPT the performance

of the optimal bidding strategy. This model however is limited in that the auction format is restricted to a second-price auction payout mechanism, whereas our approach provides a more general framework which can be applied to any auction mechanism.

3 OPTIMAL BIDDING

In this section, we detail the algorithm for learning the latent cost function and uniformly pacing an advertiser's budget throughout an advertising campaign. We first present the algorithm with the intuition behind its definition and proceed by analyzing the convergent dynamics and efficiency. We emphasize uniformly pacing an advertiser's budget, or target spend amount, so that all auction periods are weighted equally. The primary concern is to spend the entirety of their budget while staying active in the campaign for as long as possible.

We define the metrics by which a budget pacing algorithm should be measured in order to assess its overall effectiveness in online real-time bidding systems.

1. Fast convergence to a stable bidding strategy relative to the overall campaign duration.
2. Adaptive capacity for changes in the display auction mechanism (manipulated on the supplier end).

Ensuring good performance metrics is complex in the online market due to the speed and frequency that impressions are sold, yet we prove in this section that our natural learning algorithm performs well due in large part to its simplicity and efficient computation.

3.1 Pacing Algorithm

We begin by assuming an advertiser has two pieces of information: its budget, B, and the number of auction periods in the campaign, T (ie. auction periods per day). Intuitively, the advertiser who is trying to uniformly pace their budget will initially bid its average budget for the corresponding number of auctions, $b_0 = \frac{B}{nT}$ (where n is the number of auctions per period). However, in general, the actual cost is much larger than the input bid and the convergence time remains low regardless of this initial bid selection, so any choice suffices. Once an initial average bid is set, it is utilized for the first period ($t = 0$), once again being scaled for each impression in accordance with their value, and the agent incurs cost c_0 throughout this time.

Following the initial bid, an advertiser now has the information of how much was spent in the first period and can assess the discrepancy between the *desired* amount to be spent and the *actual* amount. Let B_r^t denote the remaining budget after period t, and c_t be the incurred cost in this period, then we can define the scaling factor $\frac{B_r^t}{T-t} \cdot \frac{1}{c_t}$ as the ratio of the amount the advertiser wants to spend on average in each remaining auction period to how much it spent on the previous. Using this ratio as a budget pacing factor, we have the following iterative scheme:

$$b_{t+1} = \frac{B_r^t}{c_t(T-t)} \cdot b_t$$

ALGORITHM 1 takes as input the budget (B), number of periods in a campaign (T), current period (t), the most recent bid value (b_t), and the history of previously spent amounts ($\{c_i\}_{i=0}^t$), outputting a new average bid value scaled based on this information. Note that

---

Algorithm 1 Budget Smoothing

1: **Input**: $B, T, t, b_t, (c_0, ..., c_t)$
2: **Output**: b_{t+1}
3: $B_r^t = B - \sum_{i=0}^t c_i$ {remaining budget after time t}
4: $c_{\text{opt}} = \frac{B_r^t}{T-t}$ {optimal spend amount}
5: $c_{\text{act}} = c_t$ {actual spend amount}
6: $\alpha = \frac{c_{\text{opt}}}{c_{\text{act}}}$
7: **return** $\alpha \cdot b_t$

---

each c_i is equal to $f(b_i)$, however the algorithm is trying to *learn* f, so we use the former notation.

As we will see in Section 3, this intuitive update algorithm converges to a fixed point that uniformly paces an advertisers budget within a small fraction of the total auction time, as well as undergoing interesting bifurcations when parameters of the system are tuned. A brief summary of the results for latent cost functions in the family $f(b_t) = C \cdot b_t^k$, where $k \in \mathbb{R}$ and $C \in \mathbb{R}_+$, are provided in Table 1.

In analyzing the efficiency of this learning algorithm, we consider various reasonable actual cost functions $f(b_t) = c_t$ and their divergent or convergent behavior to an optimal strategy. In the following analysis we consider only *continuous* cost functions, and as such simplify our analysis by examining polynomial functions which can approximate any such function – a consequence of the Stone-Weierstrass theorem [19].

3.2 Linear and Semi-Linear Cost Function

As a warm-up, we first consider the most simplistic form of the latent cost function: a *linear* cost. We assume that $f(b_t) = C \cdot b_t$ for some $C > 0$: the situation in which the agent running the ad sale simply scales incoming bids by a fixed constant factor. We reiterate that such a simple scaling can effectively capture the dynamics of a repeated first auction system wherein the actual amount spend by a bidder is proportional to the number of auctions won in a given period.

In this elementary case, we see that, for any initial bid placed an advertiser adapts to the latent cost function in *exactly* one iteration, bidding the optimal uniform amount for the remainder of the campaign.

THEOREM 3.1. *For a linear cost function, $f(b_t) = C \cdot b_t$ where $C > 0$, and initial bid b_0, ALGORITHM 1 converges to a fixed point bid value in exactly one iteration.*

In many practical instances, it is also important to consider a *semi-linear* cost function. We define the semi-linear cost function as $f(b_t) = \min\{C \cdot b_t, M\}^1$ where both $C, M > 0$. This is analogous to when the administration of the ad sales both scale an advertisers bids by a fixed constant factor and also instate "guard rails" on the spending: requiring the amount spent to be no greater than M. This is meant to protect against over excessive spending by an advertiser. Despite this limitation on bids, we see that the semi-linear function converges in one iteration when $M > \frac{B}{cT}$.

---

[1]Additionally, if we use the semi-linear function with a maximum rather than minimum, we see identical convergence rates.

THEOREM 3.2. *For a semi-linear cost function, $f(b_t) = \min\{C \cdot b, M\}$ where $C > 0$, initial bid $b_0 < M$, and $M > \frac{B}{CT}$ ALGORITHM 1 converges to a fixed point average bid value in exactly one iteration.*

Additionally, for $M \gg 1$, the semi-linear function is identical to the linear case. As a result, proving Theorem 3.2 encompasses the result of Theorem 3.1.

THEOREM 3.2. Assume $b_0 \neq b^*$:
$$
\begin{aligned}
b_1 &= \text{ALGORITHM } 1(B, T, 1, b_0, c_0) \\
&= \frac{B - \min\{Cb_0, M\}}{\min\{Cb_0, M\}T} \cdot b_0 \\
&= \frac{B - Cb_0}{cT} \\
b_2 &= \text{ALGORITHM } 1(B, T, 2, b_1, (c_0, c_1)) \\
&= \frac{B - \min\{Cb_0, M\} - \min\{cb_1, M\}}{\min\{cb_1, M\}(T-1)} \cdot b_1 \\
&= \frac{B - Cb_0 - \min\{C \cdot \frac{B-Cb_0}{CT}, M\}}{\min\{C \cdot \frac{B-Cb_0}{cT}, M\}(T-1)} \cdot \frac{B - Cb_0}{CT} \\
&= \frac{B - Cb_0}{CT} \\
&= b_1
\end{aligned}
$$
□

THEOREM 3.1. Using the result of Theorem 3.2 we take the limit as $M \to \infty$, thus giving $\min\{cb, M\} = cb$ with convergence in one iteration.
□

Before proceeding to give theoretical bounds on the convergence times of our algorithm for the cases where we do not see one iteration convergence, we must first define the "general cost function" and assess the fixed points of our system.

3.3 General Cost Function

In the most complex case, we consider a general cost function of the form $f(b_t) = C \cdot b_t^k$ where $k \neq 1$. In this case, we here identify the fixed point of the system and assess its stability.

THEOREM 3.3. *For $k \in (0, 2)$, Algorithm 1 has exactly one stable fixed point for cost functions of the form $f(b_t) = C \cdot b_t^k$ where $C > 0$.*

PROOF. The fixed point of our system occurs when the output of Algorithm 1 is equal to the input b_t. Solving this equation yields:

$$
b^* = \left(\frac{B - \sum_{i=0}^{t-1} Cb_i^k}{c(T-t+1)} \right)^{1/k} \tag{3.3.1}
$$

Noting that in general, if b_0 is not a fixed point, then the fixed point for period t is defined as the above. If instead b_0 is a fixed point, the system will thus bid this value on average for the entire campaign and the point has the more simplistic formula derived by the following:

$$
b^* = \left(\frac{B}{C(T+1)} \right)^{1/k} \tag{3.3.2}
$$

which is the special case of (3.3.1) where $t = 0$. As such it suffices to examine the more general formula to encompass all results.

Parameter Range	Dynamics
$k \leq 0$	Unstable fixed point
$k \in (0, 1)$	Stable fixed point with convergence time $\propto \frac{\ln(k)}{\ln(1-k)}$
$k = 1$	Stable fixed point with convergence in one iterations
$k \in (1, 2)$	Stable fixed point with convergence time $\propto \frac{\ln(2-k)}{\ln(k-1)}$
$k \geq 2$	Instability requiring guard rails on spending

Table 1: Summary of convergence results for cost functions of the form $f(b_t) = C \cdot b_t^k$.

In order to prove stability of the above fixed points, we linearize around the point b^* and assess whether the point is attracting or repelling by examining the derivative at that point [20]. Informally, if we let $|f'(b^*)| = |\lambda|$, then we have linear stability when $|\lambda| < 1$ and instability when $|\lambda| > 1$. Now we can analyze our system's fixed point. First we differentiate the function in Algorithm 1 to get the following:

$$\frac{d}{db_t} \left(\frac{B - \sum_{i=1}^{t} Cb_i^k}{Cb_t^k(T-t)} \cdot b_t \right)$$

$$= \frac{-C + (1-k)b_t^{-k}(B - \sum_{i=1}^{t-1} Cb_i^k)}{C(T-t)} \quad (3.3.3)$$

Plugging in the fixed point from (3.3.1) gives the following stability estimate respectively

$$|\lambda| = \left| 1 - k\left(\frac{T - t + 1}{T - t} \right) \right| \approx |1 - k| \text{ for } t \ll T \quad (3.3.4)$$

Thus for $0 < k < 2$, the system is in general attracted to the fixed point defined in (3.3.1), and the point is unstable outside of this parameter range. □

3.4 Convergence Time Bound

The crucial result of this paper comes from the minimal convergence time for Algorithm 1. In contrast to recent papers examining optimal online budget pacing schemes, none have analytically derived an estimate for convergence times to these optimal strategies [15].

Theorem 3.4. *For $|1-k| < 1$, Algorithm 1 has a bounded distance from the stable bid value at time t defined by:*

$$\epsilon := |b_t - b^*| \leq \gamma^{-1/k} \cdot \frac{|1-k|^{t-1+\frac{1}{k}}}{1 - |1 - k|}$$

where $\gamma = \frac{CT}{B}$. Subsequently, we have a convergence time, t^*, to the stable bid value bounded as:

$$t^* \leq \frac{k-1}{k} + \frac{\ln \left| \epsilon \gamma^{1/k}(1 - |1-k|) \right|}{\ln|1-k|} \quad (3.4.1)$$

We can see that the convergence time is dependent upon the parameters of our RTB campaign, namely the budget, length, and degree of the latent cost function. In order to prove the theorem, we first need a crucial lemma from the theory of metric spaces, proven in [2].

Lemma 3.5 (Banach Fixed Point Theorem). *Assume that $\Omega \subset \mathbb{R}^n$ is closed, and that $G : \Omega \mapsto \Omega$ is a contraction, that is, there exists $0 \leq L < 1$ such that*

$$\|G(x) - G(y)\| \leq L\|x - y\|,$$

for all $x, y \in \Omega$, where $\|\cdot\|$ is any norm on $\mathbb{R}^n$. Then the function G has a unique fixed point $x^* \in \Omega$. Additionally, let $x_0 \in \Omega$ be arbitrary and define the sequence $\{x_t\}_{t=1}^{\infty} \subset \Omega$ by $x_t = G(x_{t-1})$. Then we have the estimate

$$\|x_t - x^*\| \leq \frac{L^t}{1 - L}\|x_1 - x_0\|$$

In particular the sequence $\{x_t\}_{t=1}^{\infty}$ converges to x^*

Invoking this well studied result from Banach requires only that the mapping is contractive. We henceforth proceed in proving the theorem by demonstrating this crucial property of Algorithm 1, and applying Lemma 3.5.

Theorem 3.4. To invoke Lemma 3.5 we need to estimate an adequate Lipschitz constant L and put a proper bound on the distance between two points in the sequence defined by our algorithm. We first note that

$$|G(x) - G(y)| \leq L|x - y|$$

$$\Rightarrow \frac{|G(x) - G(y)|}{|x - y|} \leq L \leq |G'(x^*)|$$

where $x^* \in (x, y)$ gives us an easily computable bound the sequence when $|G'(x)| < 1$. Note that we now use the absolute value rather than other norms since our algorithm uses scalar valued functions.

Stemming from our analysis in Section 3.3, a Lipschitz bound for the dynamics when $0 < k < 2$ is $L = |1 - k|$. The remaining component needed to apply Lemma 3.5 is a bound on the distance $|b_1 - b_0|$. For any b_0 that is not the fixed point, we can look for the maximal distance to b_0 by taking the derivative of $b_1 - b_0$ and setting it equal to 0. Assume without loss of generality that $b_1 - b_0 > 0$.

$$\frac{d}{db_0}|b_1 - b_0| = \frac{d}{db_0} \left| \frac{B - Cb_0^k}{Cb_0^k(T-1)} \cdot b_0 - b_0 \right|$$

$$= \frac{-C + |1 - k|b_0^{-k}B}{C(T-1)} - 1$$

Setting the derivative equal to 0 and finding the maximal b_0

$$b_0 = \left(\frac{B|1-k|}{CT} \right)^{1/k}$$

which gives the distance value

$$|b_1 - b_0| = \left| \frac{T - |1-k|}{|1-k| \cdot (T-1)} \cdot \left(\frac{B|1-k|}{CT} \right)^{1/k} \right| \qquad (3.4.2)$$

Now plugging in our Lipschitz constant and maximal distance from (3.4.2) to the estimate from Lemma 3.5 yields

$$|b_t - b^*| \le \frac{|1-k|^{t-1}}{1 - |1-k|} \cdot \left| \frac{T - |1-k|}{T-1} \cdot \left(\frac{B|1-k|}{CT} \right)^{1/k} \right|$$

$$\approx \frac{|1-k|^{t-1}}{1 - |1-k|} \cdot \left| \left(\frac{B|1-k|}{CT} \right)^{1/k} \right|$$

Lastly, we can define $\gamma := \frac{CT}{B}$ to get the simplified

$$|b_t - b^*| \le \gamma^{-1/k} \cdot \frac{|1-k|^{t-1+\frac{1}{k}}}{1 - |1-k|}$$

and finally, we can rearrange this inequality to give us a bound on the convergence time:

$$t \le \frac{k-1}{k} + \frac{\ln \left| \gamma^{-1/k}(1 - |1-k|)|b_t - b^*| \right|}{\ln |1-k|}$$

$$= \frac{k-1}{k} + \frac{\ln \left| \epsilon \gamma^{1/k}(1 - |1-k|) \right|}{\ln |1-k|}$$

thus, we have the result. □

Lastly, it is important to note that up to this point, we have only considered latent cost functions which are monomial (consisting of one term). This is a simplifying assumption which is justified by bounding polynomial cost functions with monomials, detailed in the following lemma (the proof can be found in Appendix 6).

LEMMA 3.6. *For a cost function defined using the general form $c_1 b^{k_1} + c_2 b^{k_2} + ... c_m b^{k_m}$ where $k_1 > k_2 > ... > k_m$ and $b > 0$, the convergence time of ALGORITHM 1 is bounded by that of monomial functions.*

3.5 Loss of Stability

As noted in Section 3.3, when $k > 2$ the fixed point for average bidding becomes unstable. As a result, ALGORITHM 1 does not converge for a cost function of quadratic or higher degree. In this instance however, we can once again utilize guard rails on spending to obtain convergence to cyclic bidding behavior. The following theorem, proven in Appendix 6, demonstrates the importance of guard rails on spending to ensure adequate budget pacing for advertisers adapting to the RTB auction mechanism. The proof invokes the same strategy as Theorem 3.1, where we now find b_0 which are fixed points of ALGORITHM 1 applied twice, and can be found in the appendix.

THEOREM 3.7. *For a cost function of the form $min\{C \cdot b^k, M\}$ where $C > 0$ and $\frac{B}{CT} < M < B$, ALGORITHM 1 undergoes a period doubling bifurcation at $k = 2$, leading to oscillatory behavior between two average bid values for each iteration t.*

This period doubling is further indicative of a canonical path to chaos [20] and as such a loss of stability for our system.

Algorithm 2 Non-Uniform Pacing
1: **Input**: $B, T, t, b_t, (c_0, ..., c_t), \kappa_t$
2: $B_v \leftarrow \kappa_t \cdot B$
3: $b_{t+1} = $ ALGORITHM 1$(B_v, T, t, b_t, (c_0, ..., c_t))$
4: **Output**: b_{t+1}

Algorithm 3 Subthreshold Budget
1: **Input**: $B, T, b_t, (s_0, ..., s_t)$
2: **if** $B < \tau$ for some threshold τ **then**
3: $\quad B_v \leftarrow \sigma \cdot B$ for some $\sigma > 1$ fixed by the provider
4: $\quad b_{t+1} = $ ALGORITHM 1$(B_v, T, b_t, (s_0, ..., s_t))$
5: **else**
6: $\quad b_{t+1} = $ ALGORITHM 1$(B, T, b_t, (s_0, ..., s_t))$
7: **end if**
8: **Output**: b_{t+1}

---

4 REAL-WORLD IMPLEMENTATION

The algorithm presented is easy to implement and can be specifically tailored to a wide range of applications. As noted previously, slight variants on the presented algorithm are used at several companies which tailor the bidding to tertiary interests that we discuss in this section. [2] We focus on two specific variations: fluctuating target spend amounts and subthreshold budgets, however the simplicity of our algorithm allows for a plethora of adjustments for specific needs.

4.1 Non-Uniform Pacing

While our main analysis demonstrates the stability of average bidding for the desired goal of uniform pacing, the algorithm can be simply adapted to meet changing target spend amounts throughout the campaign. For instance, if an advertiser wants to decrease spending during the morning since the bulk of their target market is not online, they may decrease their target spend amount until later in the day. A variant of this form is easily achieved by adjusting the submitted budget for any given time period(s). This is achieved by adding another input parameter $\kappa_t \in (0, 1]$ which scales the budget by the multiplier at any given time point. Pseudocode for such a variation on our algorithm is presented in Algorithm 2. We note that in spite of this tailoring to a specific need, the convergence properties that were theoretically justified remain, and will be further validated in Section 5.

4.2 Subthreshold Budgets

In the instance where users submit a budget to the DSP that are too small for meaningful uniform pacing, the provider can implement a variant on our algorithm for forcibly exhausting a budget early in the campaign period, thus resulting in the user needing to update with a larger budget for the next campaign. [3] This is achieved by implementing a "virtual budget" where the DSP simply scales

---

[2]Variations of the bid pacing algorithm presented in this paper are currently implemented at two major companies with which an author was previously involved. Due to proprietary data restrictions at these companies, we focus mainly on important theoretical aspects of the algorithm along with simulations results in this paper.
[3]We once again note that this is common practice at certain companies, in an effort to force users to meet a minimum budget threshold.

up the input budget and thus the algorithm establishes a higher average bid than is feasible for the user's actual budget, leading to early exit from the campaign. While this outcome is counter to the presented practicality of the algorithm, it is an effective means by which a provider can ensure that advertisers are submitting a high enough budget to be competitive in the market space. This variant's pseudocode is presented in Algorithm 3. We reiterate that although the *intent* of this adapted algorithm is not in line with the primary justification for our presented results, it nonetheless retains the convergence properties of Section 3 and will be validated through experimentation in Section 5.

5 SIMULATION RESULTS

In this section, we proceed to validate our theoretical results through comprehensive simulations on both real and simulated data. We demonstrate the discussed convergence dynamics of the bidding scheme as well as implement our DSP within the context of a Yahoo! RTB auction system from publicly available data. All codes for the presented results are available as supplementary material to our paper.

5.1 Bid Dynamics

We here validate our model against simulated results for various different parameter values. These experiments are meant to illustrate the empirical behavior of our simplified model. We present the dynamics of bid values as a function of time in Figure 1 for the three algorithms discussed above. Namely, the first column plots the dynamics of Algorithm 1, with columns two and three depicting Algorithms 2 and 3 respectively.

For the presented simulations, the advertisers is given an arbitrary initial average bid for the first period, and subsequently bids based on the corresponding algorithms. We henceforth set the budget for all simulations at $B = 50000$, $T = 1000$, and an amount spent function defined as $f(b) = \min\{b_t^k, 100\}$. Additionally, we once again make the simplifying assumption that the spent functions are monomials, allowing us to study the dynamics exclusively as a function of the degree k.

We first consider $k = 0.5$, evaluating (3.4.1) with the given parameter set gives a bound on the iterations to convergence of $t^* \leq 31$. In simulating the model (row 1 of Figure 1) we see convergence within an error tolerance $\epsilon = 10^{-6}$ in 23 iterations for Algorithm 1, approximately 3% of the total auction period with 99.90% of the budget spent. Additionally, in the non-uniform instance where we force the system to increase bidding as the day progresses via Algorithm 2, we see that after each pacing increase the algorithm quickly converges to a new stable state until the pacing is further increased according to the advertiser's request. Lastly, setting a budget threshold of 75,000, we can use Algorithm 3 to force an early exit of the advertiser after convergence to a stable bid that is not sustainable.

Increasing the cost function parameter to $k = 1.4$, we predict the time to convergence to be at most $t^* = 19$ and in simulation (row 2 of Figure 1) we see convergence in 18 iterations with 99.87% of the budget spent. Thus, our theoretical results for the stable region are further validated in simulation. Moreover, we once again exhibit convergence to increasing bid values in the nonuniform

budget pacing context, as well as successful campaign termination by Algorithm 3.

We lastly show the interesting dynamics that arise when $k \geq 2$. At $k = 2$ we have the birth of 2-cycles in the dynamics, never converging on a singular fixed point but remaining oscillatory (as discussed in Section 3.5). As we further increase the parameter, higher order cycles emerge until the system devolves into aperiodic behavior, more formally known as chaos [20]. The results of this parameter tuning align with the bifurcations exhibited by the logistic map ($f(x) = rx(1 - x)$) [20]. The last row of Figure 1 plots the bid dynamics for $k = 2.3$ using the three algorithms. We see that in all instances, the bid value oscillates about a bid strategy until the campaign's conclusion or budget exhaustion. However, even in this unstable parameter region, our algorithm effectively mitigates this oscillatory behavior and ensures that the advertisers stays in the ad campaign for its total duration.

5.2 First Price Auction

To further demonstrate the efficacy of our budget pacing algorithm, we invoke an agent running Algorithm 1 within a first-price auction mechanism as pitted against advertisers from the Yahoo! real-time bidding dataset. This dataset contains the bids over time of all advertisers participating in Yahoo! Search Marketing auctions for the top 1000 search queries during the period from June 15, 2002, to June 14, 2003 (segregated into 15 minute time periods for each day). We simulate the efficiency of our procedure by selecting a day at random from the dataset and selecting bid values for arriving impressions within each 15 minute window via Algorithm 1. On each impression in this period, we implement a first-price auction to decide on a winner and incurred cost (ie. the highest bidder wins the impression and pays exactly what they bid). Thus, for the submitted bid value by the algorithm, we observe cost c_t (proportional to the number of auctions won) for a given 15 minute period and update the bid value for the next period in an effort to uniformly pace the budget expenditure. Figure 2 plots the proportion of budget spent throughout the day as compared to a perfectly uniform pacing.

We see that – even though the spent amount function is changing in each time period by nature of the first-price auction, number of bidders, and impression values – our algorithm successfully adjusts its bidding throughout the day and maintains a near uniform pacing of the allotted daily budget.

Intuitively, the algorithm is here increasing its bid value for a 15 minute window when it observes a cost for the prior period that indicates too few auctions were won. In a symmetric manner, when bidding too high and more spending occurs than desired, the algorithm lowers its bid value to win less of the total auctions for a period. As a result, our very simplistic algorithm manages to successfully pace its budget uniformly in the given realistic framework.

We further implement Algorithm 2 within the repeated first price auction setting, where we invoke multipliers that force minimal spending at the start of the day which slowly increases until the end of day. Such a pacing may be used for advertisers who are hoping to capture their market after the workday or those who heavily use a given platform in the evening. Figure 3 plots the proportion of
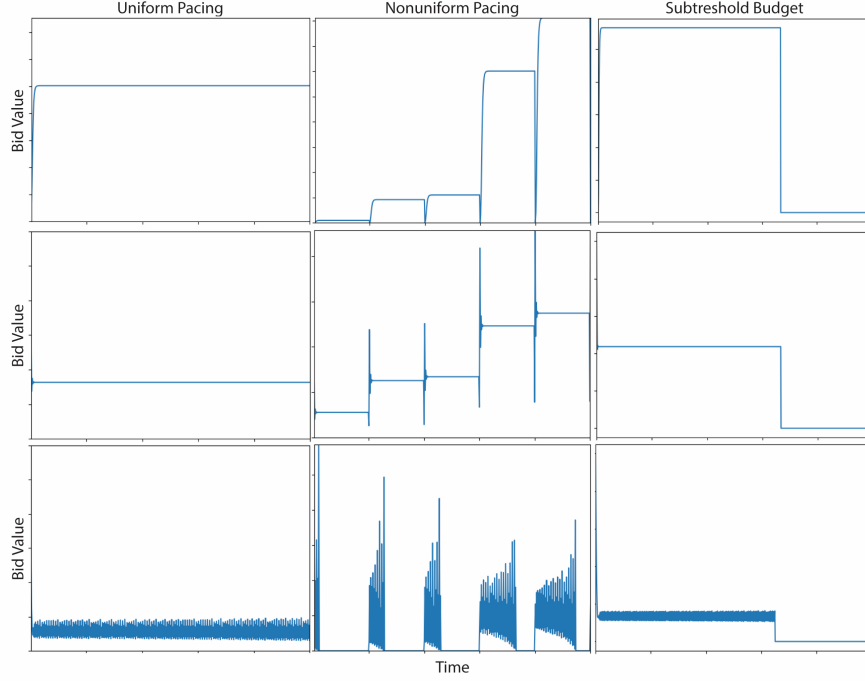
Figure 1: Bid values for different k and algorithms. The first row plots the bid dynamics of Algorithms 1, 2, 3 for $k = 0.5$. The second and third rows plot the dynamics for $k = 1.4$ and 2.3 respectively.
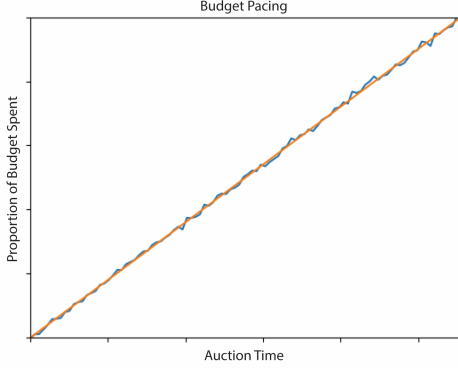

Figure 2: Proportion of the total budget spent as a function of time for an advertiser running Algorithm 1 in a repeated first-price auction format. The orange curve indicates uniformly paced budget and blue indicates the algorithms performance.
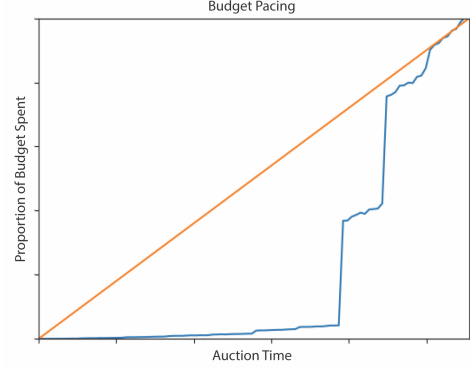

Figure 3: Proportion of the total budget spent as a function of time for an advertiser running Algorithm 2 in a repeated first-price auction format. The orange curve indicates nonuniformly paced budget and blue indicates the algorithms *nonuniform* performance.

budget spent throughout the day for this strategy as compared to a perfectly uniform pacing.

6 CONCLUSION

Bid optimization has become the central issue of budget pacing in online RTB problems. Our work has presented and analyzed a natural learning algorithm that ensures fast convergence times to

optimal bids for uniform pacing of an advertiser's budget. The algorithm is additionally robust to fluctuations in the underlying mechanism of the display ad sales, in the form of a latent cost function. The efficiency of our algorithm is demonstrated both analytically and numerically through simulations on real data, demonstrating the feasibility of its implementation in real world systems. Additionally, we highlight that the algorithm does not rely on a synthesis of

neural learning systems or more complex parameter estimation: it simply learns optimal behavior by comparing performance in the current iteration to the previous [18].

REFERENCES

[1] Deepak Agarwal, Souvik Ghosh, Kai Wei, and Siyu You. 2014. Budget Pacing for Targeted Online Advertisements at LinkedIn. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 1613–1619. https://doi.org/10.1145/2623330.2623366

[2] James Dugundji Andrzej Grana. 2003. *Fixed Point Theory*. Springer, New York.

[3] Vashist Avadhanula, Riccardo Colini-Baldeschi, S. Leonardi, Karthik Abinav Sankararaman, and Okke Schrijvers. 2021. Stochastic bandits for multi-platform budget optimization in online advertising. *Proceedings of the Web Conference 2021* (2021).

[4] S. Balseiro and Y. Gur. 2017. Learning in Repeated Auctions with Budgets: Regret Minimization and Equilibrium. *Proceedings of the 2017 ACM Conference on Economics and Computation* (2017).

[5] S. Balseiro, A. Kim, Mohammad Mahdian, and V. Mirrokni. 2021. Budget-Management Strategies in Repeated Auctions. *Oper. Res.* 69 (2021), 859–876.

[6] C. Borgs, J. Chayes, Nicole Immorlica, K. Jain, O. Etesami, and Mohammad Mahdian. 2007. Dynamics of bid optimization in online advertisement auctions. In *WWW '07*.

[7] Han Cai, Kan Ren, Weinan Zhang, Kleanthis Malialis, Jun Wang, Yong Yu, and Defeng Guo. 2017. Real-Time Bidding by Reinforcement Learning in Display Advertising. *CoRR* abs/1701.02490 (2017). arXiv:1701.02490 http://arxiv.org/abs/1701.02490

[8] A. Celli, Riccardo Colini-Baldeschi, Christian Kroer, and Eric Sodomka. 2021. The Parity Ray Regularizer for Pacing in Auction Markets. *ArXiv* abs/2106.09503 (2021).

[9] Vincent Conitzer, Christian Kroer, Eric Sodomka, and N. Stier-Moses. 2018. Multiplicative Pacing Equilibria in Auction Markets. In *WINE*.

[10] Zhe Feng, Chara Podimata, and Vasilis Syrgkanis. 2018. Learning to Bid Without Knowing your Value. *Proceedings of the 2018 ACM Conference on Economics and Computation* (2018).

[11] Joaquin Fernandez-Tapia. 2015. An analytical solution to the budget-pacing problem in programmatic advertising. *Journal of Information and Optimization Sciences* 40 (November 2015). https://doi.org/10.1080/02522667.2017.1303946

[12] Sahin Cem Geyik, Abhishek Saxena, and Ali Dasdan. 2015. Multi-Touch Attribution Based Budget Allocation in Online Advertising. arXiv:1502.06657 [cs.AI]

[13] Grégoire Jauvion and Nicolas Grislain. 2020. Optimal Allocation of Real-Time-Bidding and Direct Campaigns. *CoRR* abs/2006.07070 (2020). arXiv:2006.07070 https://arxiv.org/abs/2006.07070

[14] Orcun Karaca, Pier Giuseppe Sessa, A. Leidi, and M. Kamgarpour. 2019. No-Regret Learning from Partially Observed Data in Repeated Auctions. *ArXiv* abs/1912.09905 (2019).

[15] Kuang-Chih Lee, Ali Jalali, and Ali Dasdan. 2013. Real Time Bid Optimization with Smooth Budget Delivery in Online Advertising. *CoRR* abs/1305.3011 (2013). arXiv:1305.3011 http://arxiv.org/abs/1305.3011

[16] Alfonso Lobos, Paul Grigas, Zheng Wen, and Kuang chih Lee. 2018. Optimal Bidding, Allocation and Budget Spending for a Demand Side Platform Under Many Auction Types. arXiv:1805.11645 [math.OC]

[17] Thomas Nedelec, Noureddine El Karoui, and Vianney Perchet. 2019. Learning to bid in revenue-maximizing auctions. *ArXiv* abs/1902.10427 (2019).

[18] Claudia Perlich, Brian Dalessandro, Rod Hook, Ori Stitelman, Troy Raeder, and Foster Provost. 2012. Bid Optimizing and Inventory Scoring in Targeted Online Advertising. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Beijing, China) *(KDD '12)*. Association for Computing Machinery, New York, NY, USA, 804–812. https://doi.org/10.1145/2339530.2339655

[19] Walter Rudin. 1976. *Principles of mathematical analysis / Walter Rudin* (3d ed. ed.). McGraw-Hill New York. http://www.loc.gov/catdir/toc/mh031/75017903.html

[20] Steven H. Strogatz. 2000. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press.

[21] Srinivas Tunuguntla and Paul R. Hoban. 2021. A Near-Optimal Bidding Strategy for Real-Time Display Advertising Auctions. *Journal of Marketing Research* 58 (2021), p. 1–21. https://doi.org/10.1177/0022243720968547 arXiv:https://doi.org/10.1177/0022243720968547

[22] J. Weed, Vianney Perchet, and P. Rigollet. 2016. Online learning in repeated auctions. In *COLT*.

[23] Jian Xu, Kuang-chih Lee, Wentong Li, Hang Qi, and Quan Lu. 2015. Smart Pacing for Effective Online Ad Campaign Optimization. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Aug 2015). https://doi.org/10.1145/2783258.2788615

[24] Weinan Zhang, Shuai Yuan, and Jun Wang. 2014. Optimal Real-Time Bidding for Display Advertising. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, New York, USA) *(KDD '14)*. Association for Computing Machinery, New York, NY, USA, 1077–1086. https://doi.org/10.1145/2623330.2623633

A OMITTED PROOFS

A.1 Proof of Lemma 3.6

We proceed by bounding the cost function. For $0 < b < 1$, we bound the function using the minimal exponent

$$c_1 b^{k_m} < c_1 b^{k_1} + c_2 b^{k_2} + \ldots + c_m b^{k_m} < \tilde{C} b^{k_m}$$

where $\tilde{C} := \sum_i^m |c_i|$ and identically for $b \geq 1$, we bound the function with the maximal exponent

$$c_1 b^{k_1} < c_1 b^{k_1} + c_2 b^{k_2} + \ldots + c_m b^{k_m} < \tilde{C} b^{k_1}$$

For convenience, let $s(b) := c_1 b^{k_1} + c_2 b^{k_2} + \ldots + c_m b^{k_m}$ and without loss of generality assume $b \geq 1$. Using the above we demonstrate the following bounds on Algorithm 1 for the general cost function

$$c_1 b^{k_m} < s(b) < \tilde{C} b^{k_m}$$

$$\frac{1}{c_1 b^{k_m}} > \frac{1}{s(b)} > \frac{1}{\tilde{C} b^{k_m}}$$

$$b \cdot \left(\frac{B}{T c_1 b^{k_m}} - \frac{1}{T} \right) > b \cdot \left(\frac{B}{T s(b)} - \frac{1}{T} \right)$$

$$> b \cdot \left(\frac{B}{T \tilde{C} b^{k_m}} - \frac{1}{T} \right)$$

which is equivalent to the output of Algorithm 1 with $t = 0$. Thus, we have the algorithm bound

$$\text{Algorithm } 1(B, T, 0, b_0, c_1 b_0^{k_m})$$
$$> \text{Algorithm } 1(B, T, 0, b_0, s(b_0))$$
$$> \text{Algorithm } 1(B, T, 0, b_0, \tilde{C} b_0^{k_m})$$

Lastly, manipulating the result from Banach in Lemma 3.5 we have the convergence time bound

$$t \geq \ln \left(\frac{(1 - L)|b_t - b^*|}{|b_1 - b_0|} \right) / \ln(L)$$

substituting the output of Algorithm 1 for b_1, we thus define the convergence time bound for a cost function c as

$$\tau_{s(b)} := \ln \left(\frac{(1 - L)|b_t - b^*|}{|\text{Algorithm } 1(B, T, 0, b_0, s(b_0)) - b_0|} \right) / \ln(L) \tag{A1.1}$$

Now combining with the algorithm bounds above gives

$$\frac{(1 - L)|b_t - b^*|}{|\text{Algorithm } 1(B, T, 0, b_0, c_1 b_0^{k_m}) - b_0|}$$
$$< \frac{(1 - L)|b_t - b^*|}{|\text{Algorithm } 1(B, T, 0, b_0, s(b_0)) - b_0|}$$
$$< \frac{(1 - L)|b_t - b^*|}{|\text{Algorithm } 1(B, T, 0, b_0, \tilde{C} b_0^{k_m}) - b_0|}$$

which using definition (A1.1) equates to

$$L^{\tau_{c_1 b^{k_m}}} < L^{\tau_s} < L^{\tau_{\tilde{C} b^{k_m}}} \implies \tau_{c_1 b^{k_m}} < \tau_{s(b)} < \tau_{\tilde{C} b^{k_m}}$$

bounding the general cost function's convergence time with monomials.

A.2 Proof of Theorem 3.7

A 2-cycle for a function $f(x)$ exists if and only if there are two points x_1, x_2 such that $f(x_1) = x_2$ and $f(x_2) = x_1$. Equivalently, these points must satisfy $f(f(x_1)) = x_1$ and $f(f(x_2)) = x_2$. We proceed by finding the points that satisfy the latter condition, in our context being the

points which are fixed points when applying Algorithm 1 twice. Solving this relation amounts to the following problem

$$b_1 = \text{Algorithm } 1(B, T, 0, b_0, \min\{cb_0^k, M\})$$

$$= \frac{B - \min\{cb_0^k, M\}}{\min\{cb_0^k, M\}T} \cdot b_0$$

$$b_2 = \text{Algorithm } 1(B, T, 1, b_1, (\min\{cb_0^k, M\}, \min\{cb_1^k, M\}))$$

$$= \frac{B - \min\{cb_0^k, M\} - \min\{cb_1^k, M\}}{\min\{cb_1^k, M\}(T - 1)} \cdot b_1$$

$$= \frac{B - \min\{cb_0^k, M\} - \min\{c\left(\frac{B-\min\{cb_0^k,M\}}{\min\{cb_0^k,M\}T} \cdot b_0\right)^k, M\}}{\min\{c\left(\frac{B-\min\{cb_0^k,M\}}{\min\{cb_0^k,M\}T} \cdot b_0\right)^k, M\}(T - 1)}$$

$$\cdot \frac{B - \min\{cb_0^k, M\}}{\min\{cb_0^k, M\}T} \cdot b_0 \qquad\qquad\qquad\qquad\qquad\qquad\qquad\text{(A2.1)}$$

We now consider three unique cases for the value of the minimizations:

<u>Case 1:</u> $\min\{cb_0^k, M\} = cb_0^k, \min\{cb_1^k, M\} = cb_1^k$
Setting (A2.1) $= b_0$ and solving gives us

$$b_0 = \left(\frac{B}{c(T + 1)}\right)^{1/k}$$

which is the original fixed point)3.3.1) from Section 3.3 that was proved to now be unstable for $k > 2$. As a result, the value of the bid will be repelled from this point to the two points of our cycle. It is important to note that any fixed point of a function f satisfies $f(x) = x$ so we automatically have $f(f(x)) = f(x) = x$. We thus need not consider this trivial point in the 2 cycle, as we are interested in the situation where $f(f(x)) = x$ and $f(x) \neq x$.

<u>Case 2:</u> $\min\{cb_0^k, M\} = cb_0^k, \min\{cb_1^k, M\} = M$
Once again solving (A2.1) $= b_0$ produces the following point

$$b_0 = 2^{-1/k}(\sqrt{(-2B - MT^2 + MT + M)^2 + 4B(M - B)}$$

$$+2B + MT^2 - MT - M)^{1/k}$$

for which we need to verify $b_0 \in \mathbb{R}$. Real solutions exist for

$$\sqrt{(-2B - MT^2 + MT + M)^2 + 4B(M - B)} + 2B + \ldots$$

$$MT^2 - MT - M > 0$$

$$\Rightarrow (-2B - MT^2 + MT + M)^2 + 4B(M - B)$$

$$> (M + MT - 2B - MT^2)^2$$

$$M > B$$

however this last inequality on M does not hold. Therefore $b_0 \in \mathbb{C}$ in this case and thus this is not one of our period two points since our bid values are constrained to the real numbers.

<u>Case 3:</u> $\min\{cb_0^k, M\} = M, \min\{cb_1^k, M\} = cb_1^k$
Solving (A2.1) $= b_0$ once again yields

$$b_0 = \left(\frac{(B - M)^2}{c(B - M - MT + MT^2)}\right)^{1/k}$$

for which we once again need to verify $b_0 \in \mathbb{R}$. It is clear that $(B - M)^2 > 0$, so we need only check

$$c(B - M - MT + MT^2) > 0$$

$$cM(T^2 - T - 1) > B$$

$$M > \frac{B}{c(T^2 - T - 1)}$$

For $T > 1$, $\frac{B}{cT} > \frac{B}{c(T^2-T-1)}$. Combining this with fact that $M > \frac{B}{cT}$, verifies the above inequality and thus $b_0 \in \mathbb{R}$. Therefore, this is one of the points in our 2-period cycle. Lastly, to find the second point of this cycle, we simply apply the algorithm again. Let b_- denote the first point. We once again have two subcases which depend on the value of c. For $\min\{cb_-^k, M\} = cb_-^k$

$$
\begin{aligned}
b_+ &= \text{Algorithm } 1(B, T, 0, b_-, \min\{b_-^k, M\}) \\
&= \frac{B - \min\{b_-^k, M\}}{\min\{b_-^k, M\}T} \cdot b_- \\
&= \frac{B - b_-^k}{b_-^k T} \cdot b_- \\
&= \frac{MT}{B-M}\left(\frac{(B-M)^2}{c(B-M-MT+MT^2)}\right)^{1/k} \\
&= \frac{MT}{B-M} \cdot b_-
\end{aligned}
$$

and for $\min\{cb_-^k, M\} = M$

$$
b_+ = \frac{B-M}{MT} \cdot b_-
$$

Therefore we have our two cycle producing oscillatory behavior between b_+ and b_-.