Optimizing Ad Refresh In Mobile App Advertising

Florin Constantin, Christopher Harris, Samuel Ieong, Aranyak Mehta, Xi Tan
Google
Mountain View, CA, USA
florin,ckharris,sieong,aranyak,xit@google.com

ABSTRACT

In-app advertising is a complex market worth billions of dollars per year, yet it has been studied significantly less than traditional web display ads. In this paper we study an important but often overlooked feature of ads in mobile apps (mostly absent in traditional web ads), that of *ad refreshes*: A user is shown a stream of banner ads during the app session, in which each ad is displayed in the ad slot for a certain amount of time (the *refresh rate*) before the ad-slot is refreshed to the next ad. Data analysis on our large-scale experiments that vary refresh rates reveals a surprising result, that cannot be explained by existing user click models: *Varying ads' refresh almost preserves total number of clicks.*

We propose a new, natural, "two-phase" click model for this setting that explains this independence, as well as our measurements of the click-through rate as a function of the impression's time-on-screen and of ad-repeat counts. The new click model leads to a clean formulation of the problem of auctioning the entire user-session: i.e., determining online, both the sequence of winning ads as well as the amount of time to display each one. We complement the theoretical auction design with results from a live-traffic experiment with its implementation. Our experiments and analysis provide the theoretical foundation for AdMob's "Google-optimized refresh rate" feature, used by many mobile apps for better monetization of ads shown to millions of users.

ACM Reference Format:
Florin Constantin, Christopher Harris, Samuel Ieong, Aranyak Mehta, Xi Tan. 2018. Optimizing Ad Refresh In Mobile App Advertising. In *The 2018 Web Conference, April 23–27, 2018, Lyons, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186045

1 INTRODUCTION

The Mobile App ecosystem has grown to a substantial size over the last several years, and hence app monetization has become increasingly important. While some apps monetize by charging for installs or in-app purchases, a large population of apps use advertising as their main monetization strategy.

WWW 2018, April 23–27, 2018, Lyons, France

ACM ISBN 978-1-4503-5639-8/18/04..
https://doi.org/10.1145/3178876.3186045

1.1 Background on Ads in Mobile Apps

Ads in apps appear similar to traditional display ads on the web, but there are a few key differences. In this paper we address such an important dimension, mostly unique to the app domain: *the time duration of ads on screen.* An app typically obtains ads via an ad SDK (e.g., Google's AdMob SDK[1]), that queries an ad server (with an http request) and parses the response in order to display the ad. Our focus is on *banner ads*, which occupy a small portion of the screen, e.g., a band of size 320x50 pixels at the bottom, without interrupting the main flow of the app[2]. Once the SDK gets the response to a banner ad request, it displays the ad on screen for a specific interval of time, called the *refresh rate* (specified by the mobile app publisher, or set to a default, often 60 seconds), at the end of which it sends the next ad-request to change to a new banner ad. Sometimes, several consecutive impressions may show the same ad, causing the user to perceive an ad duration larger than the refresh rate.

Advertisers submit a $BidCpC$ (BidCostPerClick), which is an estimate of their value per click[3] (Internet advertisers typically pay only for clicks). The advertiser's estimated value of the impression is then $BidCpC \times pCTR$, where $pCTR$ is a prediction of the click-through-rate (CTR), i.e. the probability of a click given an ad impression.

1.2 Summary of Results

The Main Insight: A traditional (web-advertising based) auction would respond to an ad request by picking the ad with the highest value and pricing it by the ad with the second-highest value. However, this approach (a fixed refresh rate and the traditional auction) misses something new in apps: When a user is on an app, they see a *session* of ads, i.e. a sequence of ads, each on screen for some duration. Our main insight is that the correct good to sell in this setting is the effective time-on-screen. In this paper, we measure and model how ad time-on-screen and ad sequencing affect the user response to ads, and how this affects the optimal auction for the entire session. The paper is structured as follows:

Sec. 2: Temporal session analysis. In Secs. 2.2 and 2.3, we measure (from live traffic) a typical ad's CTR as a function of the ad's time on screen and of the number of its previous impressions. Sec. 2.4 describes a quite surprising result from our set of experiments that vary the refresh rate significantly: The total number of clicks in the session remains

[1] http://firebase.google.com/docs/admob/ios/download
[2] This is the most common ad type shown in apps (by total traffic). Other in-app ad types include interstitial (full-screen) and native ads.
[3] Our results extend to advertisers that optimize and bid for post-click events like installs or in-app purchases, a rapidly growing population.

mostly unchanged across the experiment arms. This cannot be explained by a traditional impression-based click model.
Sec. 3: A new click model. To support our empirical data, we introduce a new user click model that separates attention versus interest i.e. the user noticing the ad versus clicking it.
Sec. 4: Session auction. We then ask how the observations and the new model should inform auction design. We provide a clean formulation of the "Session Auction" problem which is to find a welfare optimal allocation and pricing for the entire user session, in an online manner.
Sec. 5: Online algorithm with tight competitive ratio. We describe an online allocation algorithm, bound its competitive ratio, and provide a tight example.
Sec. 6: Experiment results. We conclude with live-traffic data from an implementation of the new time-aware auction.

Our results provide the theoretical foundation for AdMob's "Google-optimized refresh rate" feature[4], used by many mobile apps for better monetization of ads shown to millions of users.

1.3 Related Work

Refresh rate in industry. From published industry practices, the refresh rate is clearly an important knob, and setting it optimally is not a trivial task: AdMob (Google) suggests 60-plus seconds[5], Facebook Audience Network suggests 30-plus seconds[6], whereas MoPub allows apps to experiment on refresh rate[7]. The Interactive Advertising Bureau, an industry standards organization, set guidelines[8] for Mobile In-App advertising, calling out the importance of auto-refresh ads, the app Session (as an alternate window of measurement), and the average ad exposure time (for reporting).
App Refresh rate literature. A few recent papers study refresh rates in apps. The paper closest to ours is [10], which argues that the correct metric for ad performance in apps is Clicks-per-Hour, and not the typical Clicks-per-Impression (CTR). It further showed that for a specific set of apps, total clicks increased as ad durations shortened, which in fact, is contrary to our findings (more in Sec. 3.2—our model identifies the features which influence the change in clicks). Further, [11] and [5] studied the savings in battery/resource usage if ad refresh is slowed; neither paper considered click behavior.
Time on Screen. A second closely related paper, [4], studies the effect of exposure time (i.e., time on screen) of display ads on the web; it is, to our knowledge, the first paper to study this feature. The paper describes experiments that increase the exposure time (although for one of two ads only), or sequence ads differently. The main metric is brand-awareness / ad recall, which is natural for some web display ads, while we focus on click-based ad value because we consider performance advertising. We also further study the question of optimal auction design in this setting, based on the empirical observations and modeling. Finally, the experiments in [4] are done with 1000 Amazon Mechanical Turk users, but we perform large-scale live-traffic AdMob experiments.
User Click Models. User click models have been closely studied. The basic model for position auctions was defined or implicit in [2, 3, 12]. The model postulated a separate and independent effect of ad positions on user behavior. Richer user models, such as the Markovian or Cascade models, were proposed in [1, 8]. These and other related models are per-impression whereas our work's novelty is in considering the effect of time, a dimension unique to the setting of streaming ads. Finally, [9] reviews modeling techniques for user click behavior.
Auction Design. Our click model simplifies the auction design question to a fairly standard one, that of allocating a divisible good to a set of agents with concave utilities. This abstract setting has been well studied in the literature, notably in the work on network resource allocation ([6], and subsequent results). Much of the work studies the setting of a single price per unit, which leads to the *Proportional Allocation* auction design. Our goal is to run an efficiency-maximizing auction; this too has been studied before [7]. However, to our knowledge, the problem of allocation of divisible goods with a *lower bound on any non-zero allocation* has not been considered before. While the lower bound appears specific to our motivating problem, it leads to an intriguing gap between the online and offline allocation values.

2 CLICK BEHAVIOR ALONG SESSION

We begin with data analyses and live experiment analyses on user click behavior as a function of temporal features such as refresh times and ad sequencing. We make three observations that motivate a new click behavior model.

2.1 Data Preliminaries

Our source data is sampled from live traffic obtained from a mobile ad network, covering apps like games, utilities, news, and retail. The ads served include ads that promote other apps and ads that direct users to mobile-optimized websites.

We define an *ad session* to be a consecutive sequence of ads shown to a given user within a single app without a break of more than 15 minutes[9]. As we are specifically interested in the effect of refresh rates, we include only banner ads in the session—banner ad slots are (typically) automatically refreshed at fixed time periods. All results are after the ad impressions and clicks have been cleaned of spam. Our study used over 100M sessions, spanning several months.

Due to confidentiality, we are not able to make the data publicly available. The exact values of the CTR have also been masked. Our analysis presents the *relative* difference in CTR, which is preserved without the exact values. We acknowledge that the confidentiality limits our experiments' reproducibility. Nonetheless, we have found consistent results across the entire time span. Also, we believe results can be verified by an analogous study using data from a different mobile ad network or by independent app publishers.

[4] See an AdMob account with this feature at http://ibb.co/gzeEkw
[5] http://support.google.com/admob/answer/3245199
[6] http://developers.facebook.com/docs/audience-network
[7] See "Optimizing Refresh Rates for Banner Ads" at www.mopub.com/wp-content/uploads/2015/02/Pubs-MaximizeRevenuePlaybook.pdf
[8] http://tinyurl.com/IAB-app-session. Links visited Oct. 2017.
[9] Other definitions e.g., usage across apps, yield similar results

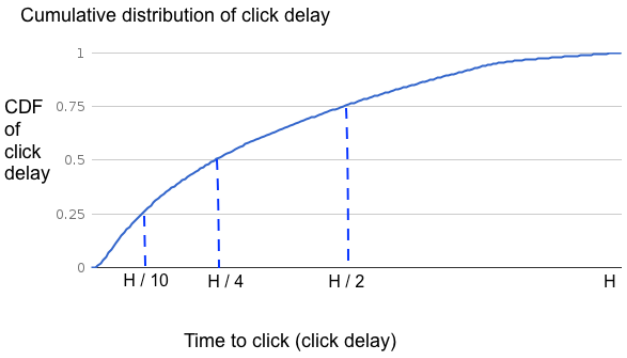

Figure 1: Cumulative distribution of clicks as a function of click delay. Time (click delay) is on the x-axis, capped at H (whose exact value is masked for confidentiality). The quartiles of the distribution are at about $H/10$, $H/4$, and $H/2$.

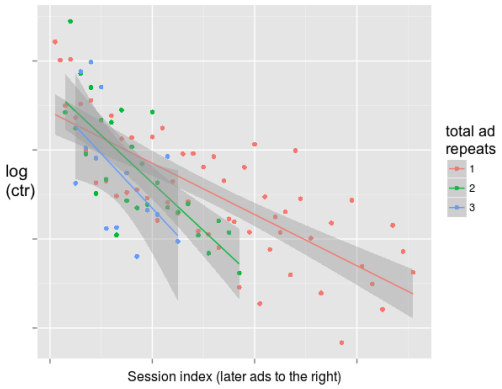

Figure 2: CTR drops along the session index, but falls faster for higher ad repeat count (number of times shown in the session thus far). Each point on the plot aggregates many ad impressions (at least 1000, usually much more) with same session index and same ad repeat: e.g. session index = 5 and ad repeat = 2 for the second a_3 in the ad sequence a_1, a_2, a_3, a_1, a_3. The conclusion follows from the best-fit lines, also shown with confidence intervals.

2.2 Time to Click

Time to click, also known as *click delay*, measures how long it takes before a user clicks on an ad after it was first shown on screen. More formally, consider a banner ad shown on screen starting at some time t_0 and intended to be on screen for a time duration of R (there can be other ads on screen prior to and after this ad). The first question we ask is: During an ad's screen lifetime, how soon does a click occur if it occurs at all? Suppose a click occurs at time t_c (before $t_0 + R$). Fig. 1 shows the cumulative distribution of the click delay, $(t_c - t_0)$, averaged over all clicks, where H is an upper bound on R.

The observed CTR of an ad impression is a function of its time on screen: As we persist the ad on screen for longer, CTR increases, but with a decreasing marginal rate. The underlying CTR distribution is well approximated by an *exponential cdf* after excluding the first few seconds to account for user reaction time. The model coefficient is about -0.015 with an R^2 of around 0.93. This holds for both the first and the subsequent impressions of the same ad.

2.3 Effect of Previous Impressions on CTR

Next, we consider the effect of ad sequencing on CTR, mainly how CTR changes when an ad is repeated in a session (not necessarily in consecutive impressions, but possibly interleaved by other ads). At a first cut, we observe that later impressions have lower CTR: In instances where we show a given ad for multiple impressions in the session, the CTR of each repeat impression is lower than the one preceding it.

Looking at the data more carefully, we notice a confounding factor in the analysis—the index of the ad in the session. We notice that CTR generally decreases as one goes deeper into the session, as a long session is usually indicative of the user being highly engaged with the app. Since a repeated ad occurs later in the session (often much later), we need to attribute the drop in CTR between the two effects. Fig. 2 disentangles the effects on CTR of session index and ad repeat count. It plots the (natural logarithm of) CTR on the y-axis as a function of session index (x-axis), grouped by ad repeat count. We observe that the drop off in CTR is more significant (the slope is steeper) for repeated ads.

To further understand the effects of these factors, we ran a logistic regression (a standard model for CTR prediction [9]) of clicks against them. The model coefficients for session index and ad repeat count are -0.05 and -0.25 respectively, both significant at p-values of 0.001. Roughly speaking, we see a *geometric drop-off* along repeat impressions: each subsequent impression is about 20% less effective than the previous one.

2.4 Effect of Changing Refresh Rates

In Sec. 2.2, we saw how the CTR of an impression is a concave function of its time-on-screen. Here we ask the same question, but at the session level: In what way does the *total number of clicks in the session* depend on the refresh rate of the impressions. While the previous questions were investigated by data analysis of the results of the current setting of refresh rate, here we need to run live-traffic experiments.

Experiment Setup: We sample from ad slots (apps) whose refresh rates are at some fixed $R > 0$. We then sample from the set of mobile devices. In the treatment groups, we vary the refresh rates (with arms $0.5R, 1.5R, 2R, 2.5R, \infty$) and measure how the volume of clicks changes. Note that this has an impact on both the time-on-screen per ad, as well as the ad sequence itself: for example, doubling the refresh time (to $2R$) halves the number of ad impressions in the sequence, since the session time is fixed.

We can attempt to predict the experiment results using the **standard click model** used widely in the sponsored search literature (e.g. [2, 3, 12]), which assumes:

$$\text{Total Clicks} = \sum_{a \in \text{shown ads}} \text{UINorm}_a \times pCTR_a$$

where the UINorm_a (User Interface Normalizer) factor captures the prominence of ad a. In our case, the UINorm should capture the time-on-screen effect (since all positions are identical), which was shown in Sec. 2.2 to be well approximated by an exponential CDF. So, in the $2R$ experiment arm we would expect a sum over half the number of impressions, with the UINorm per impression increasing, though nowhere close to doubling. Thus, the classic model would predict a large drop in clicks in this arm (and similarly a large increase in clicks in the $0.5R$ arm).

To our surprise, we found that the number of clicks between the control group and the treatment group were in fact very close to one another. The table below describes the average change in clicks observed upon changing the refresh times in a range, where a refresh rate of ∞ means we never refresh. Each group had at least 10000 clicks. ΔClicks denotes the ratio of treatment clicks to control clicks, minus 1, i.e. $\Delta\text{Clicks} = 0\%$ if both groups had the same total clicks.

Refresh Rate	95% Confidence Interval of ΔClicks
$0.5R$	$[-2.7\%, +0.7\%]$
$1.0R$	NA (control)
$1.5R$	$[+0.2\%, +3.6\%]$
$2.0R$	$[-1.0\%, +2.4\%]$
$2.5R$	$[0.0\%, +3.4\%]$
∞	$[-2.8\%, +0.5\%]$

Since the click change is almost nil, the CTR change turns out to be almost directly proportional to the change in refresh rate. These results motivate us to reconsider the standard click model, and we instead present a new model for click prediction, which not only explains all three observations in this section, but also helps us better understand the factors that influence click volume as refresh rate changes.

3 A NEW CLICK BEHAVIOR MODEL

Our new in-session click model consists of two random processes, for (1) user attention and (2) user interest:

(1) **Read Process:** A Poisson process with parameter λ (depends on app and user, but not on the choice of ads and ad durations). This determines the times in the session when the user reads the banner ad block.

(2) **Click Process:** If the user reads ad a then
− If this is the first read on a in the session, then the user clicks with probability $pCTR_a$ (which can depend on ad a, app and user).
− Else (if there was a read on a previously in the session) there is no click.

The intuition here is that over a session, a user looks at the ad block (a "read") at random times, according to a Poisson process[10]. If this is the first time they read some ad a (possibly a had some previous impressions that the user did not read), then they will click on the ad with probability $pCTR_a$. If instead, the user has already read an earlier impression of this ad in the session, then they will not click on the current impression. This is independent of whether the user actually *clicked* on this ad's previous impression. We call this the *non-cumulative effect* assumption, to emphasize that our model does not incorporate the so-called brand-effect (in which multiple impressions together lead to an action).

While $pCTR$ is usually defined as the probability of a click given an ad impression, here we define it the probability of a click given the ad's *first read* in the session. Fig. 3 discusses an example user session with four reads and a click.

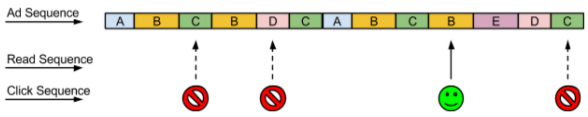

Figure 3: Example ad stream in a user session. The top row depicts the sequence of ads on screen over time. The Poisson Read process triggers four times (at the arrows) on ads C, D, B and a second C. The Click process is enabled upon each read and probabilistically triggers a click on ad B (at the green happy face). Note that this is the first time in the session that the user read the ad B. Ad C's first impression was read, but not clicked on, so the second read of C deterministically does not get a click.

Whereas a positive outcome in the click process (user clicks on the ad) is observed by the auctioneer, the entire read process is hidden from the auctioneer, who does not know if the user has read the ad. Thus, a lack of a click on the ad could either be due to the user not having read the ad, or the user choosing not to click the ad after reading it.

We note that previous related work on user models (Sec. 1.3) also, in effect, divide the click model into a user interface effect (or Read process), and a Bernoulli Click process. However, the Read processes in those models are impression based (e.g., position normalizers), while the novel aspect here is that the Read-Process is a time-indexed random process.

Let us show how the model explains all three findings above.

3.1 Time to Click and Effect of Ad Repeats

These two observations are explained simultaneously. Fix an ad, and suppose we show impressions for this ad k times in the user's session, for time durations $t_1, t_2, \ldots t_k$ (interleaved by other ads). Let V_i be the event that the ith impression of the ad got a read (V for view), and let C_i be the event that the ith impression of the ad got a click. In our model $pCTR$

[10]The model can further be extended to include a time-varying $\lambda(t)$, to capture the change in user attention over the session timeline.

is defined as the probability of a click given the first read of the ad. Then, for the first impression of the ad:

$$\Pr(V_1) = 1 - e^{-\lambda t_1} \quad \text{(Read-Process)}$$
$$\Pr(C_1|V_1) = pCTR \quad \text{(Click Process, } V_1 \text{ is the first read)}$$
$$\Rightarrow \Pr(C_1) = \Pr(C_1|V_1) \cdot \Pr(V_1) = pCTR \cdot (1 - e^{-\lambda t_1})$$

The ad's second impression may be clicked only if the first impression had not been read. Since $\Pr(V_2) = 1 - e^{-\lambda t_2}$

$$\Pr(C_2|V_2) = pCTR \cdot (1 - \Pr(V_1)) + 0 \cdot \Pr(V_1) = pCTR e^{-\lambda t_1}$$
$$\Rightarrow \Pr(C_2) \;=\; pCTR \cdot e^{-\lambda t_1} \cdot \left(1 - e^{-\lambda t_2}\right)$$

One interpretation is that the $pCTR$ of the second impression is reduced as a function of the time for which the *first* impression was shown. The time lengths of the other interleaved ads do not affect the second impression's $pCTR$. Similarly, we can show that, for general $i \leq k$:

$$\Pr(C_i) \;=\; pCTR \cdot e^{-\lambda \sum_{j=1}^{i-1} t_j} \cdot \left(1 - e^{-\lambda t_i}\right) \qquad (1)$$

Eq. (1) shows that the click-through rate, and equivalently, the time-to-click follow an exponential cdf, explaining the observations in Sec. 2.2. Further, if all impressions in the session are shown for the same duration R then $\Pr(C_i) = [pCTR \cdot e^{-\lambda(i-1)R}] \cdot (1 - e^{-\lambda R})$. Thus, CTR drops off geometrically in the number i of repeat impressions of this ad along the session. This explains the analysis in Sec. 2.3.

Finally, since the C_i are disjoint events, we get

$$\Pr(\text{Click on some impression of the ad}) =$$
$$= \sum_{i=1}^{k} \Pr(C_i) \;=\; pCTR(1 - e^{-\lambda \sum_{i=1}^{k} t_i}) \qquad (2)$$

i.e., the click probability on any one of the interleaved impressions is equal to the click probability on a single long impression with time duration equal to the sum of the individual times. This is an important consequence of our click model, which greatly simplifies further analysis, in particular our auction design (Sec. 4).

3.2 Conservation of Clicks

The standard click model can potentially also explain the first two observations (with appropriate assumptions on time-based UINorms and "hard-coding" the repeat effects). But as we have noted earlier, it cannot explain the counter-intuitive result that the total number of clicks in the session turns out to be independent of the refresh rate. We now show how the Read-Process can help us explain this observation.

3.2.1 **Intuition**. The high level intuition is that the number of clicks depends primarily on the number of reads in the session. We have assumed that the read parameter λ is independent of the ads served and their times on screen. Therefore, if we change the refresh times in an experiment, the number of reads in the session does not change. This by itself is not a proof that clicks will not change (even considering that the click process for any given ad is independent of the refresh rate), since the reads can now fall on a different set of ads. If the reads fall on *the same mix of ads* in the experiment as in the control, then total clicks will not change.

The mix of ads that the reads fall on in turn depends on:

(1) The magnitude of read parameter λ to refresh rate R
(2) The diversity of ads shown in the session, in particular the number of distinct ads shown (in experiment versus control) and the distribution of their $pCTR$s.

3.2.2 **Formalization**. To formalize the intuition above, we start with some notation. Let $a_1, a_2, \ldots, a_k$ be the unique candidate ads, and let $pCTR_i$ denote the predicted CTR of a_i (i.e., the probability of a click on the first read of a_i). Let Control denote the treatment with a refresh time of R, and experiment denote that with an refresh time of $2R$ (the subsequent analysis holds for any change in the refresh time).

Consider an app session in the control, during which the ad system (after a complex sequence of steps like retrieval, prediction, scoring, budgeting, and auction) returns a sequence of ad impressions $\mathcal{C} = (i_1, i_2, \ldots, i_n)$ (with each impression of length R and with possible ad repeats). It is reasonable to assume that the experiment will see the sequence $\mathcal{E} = (i_1, i_2, \ldots, i_{n/2})$, i.e., the half-prefix of $\mathcal{C}$ (with each impression of length $2R$).

For $i = 1, \ldots, k$, let μ_i denote the number of impressions of ad a_i (call it the multiplicity) in the first half of $\mathcal{C}$. Note that μ_i is also the multiplicity of a_i in $\mathcal{E}$. Let ν_i denote the multiplicity of a_i in $\mathcal{C}$'s second half. Let $Clicks(\mathcal{C})$ and $Clicks(\mathcal{E})$ denote total clicks in control and experiment respectively.

The total amount of time on screen for a_i in $\mathcal{C}$ is $(\mu_i + \nu_i)R$, and in $\mathcal{E}$ is $\mu_i 2R$. From Equation 2, we have:

$$Clicks(\mathcal{C}) = \sum_{i=1}^{k} pCTR_i \left(1 - e^{-\lambda R(\mu_i + \nu_i)}\right) \qquad (3)$$
$$Clicks(\mathcal{E}) = \sum_{i=1}^{k} pCTR_i \left(1 - e^{-\lambda 2R\mu_i}\right) \qquad (4)$$

It does not seem possible to derive a closed form or a general prediction for the change in clicks, due to the sensitivity to the sequence $\mathcal{C}$, but we can evaluate a few broad cases:

- For the case of $\mu_i = \nu_i, \forall i \in [1..k]$, Equations 3 and 4 imply that there is no change in the number of clicks. This case corresponds loosely to having a uniform spread of each ad along the app session.
- The $\lambda R \to \infty$ case encodes a read rate much faster than the refresh rate ($1/\lambda \ll R$). Then (3), (4) simplify to $Clicks(\mathcal{C}) = \sum_{i=1}^{k} 1\{\mu_i + \nu_i > 0\} pCTR_i$, and $Clicks(\mathcal{E}) = \sum_{i=1}^{k} 1\{\mu_i > 0\} pCTR_i$. This immediately shows that the clicks cannot increase. Further, if every ad which occurs in $\mathcal{C}$ also occurs in its first half, i.e., also in $\mathcal{E}$, then the number of clicks remains constant. On the other hand, consider the skewed example $\mathcal{C}_1 = (a_1, a_1, \ldots\ldots, a_1, a_2, a_3, a_4, \ldots, a_k)$, in which a_1 repeats $k-1$ times. Suppose all $pCTR$s are equal to p. Then $Clicks(\mathcal{C}) \to (1+k)p$, while $Clicks(\mathcal{E}) = p$, meaning that clicks would drop dramatically, since the experiment would only show a_1.
- Finally, consider the case in which all ads in $\mathcal{C}$ are distinct. Then, for half the ads, $\mu_i = 1$ and $\nu_i = 0$, and for the other half, $\nu_i = 1$ and $\mu_i = 0$. Consider

the sub-case in which all ads have the same $pCTR$. So we get $Clicks(\mathcal{C}) = n(1 - e^{-\lambda R})$ and $Clicks(\mathcal{E}) = \frac{n}{2}(1-e^{-2\lambda R})$. So the ratio of change in clicks is $\frac{1+e^{-\lambda R}}{2}$, showing that clicks cannot increase in the experiment for any value of λR. Now consider the more realistic case in which the ads in $\mathcal{C}$ have decreasing $pCTR$s, and that the average $pCTR$ in the second half of $\mathcal{C}$ is lower by a factor, say $\alpha \in [0, 1]$, than the average $pCTR$ in the first half. Then the ratio of clicks becomes

$$\frac{Clicks(\mathcal{E})}{Clicks(\mathcal{C})} = \frac{(1 - e^{-2\lambda R})}{(1 + \alpha)(1 - e^{-\lambda R})} = \frac{1 + e^{-\lambda R}}{1 + \alpha}$$

In this case, the direction of change in clicks depends on which of α and $e^{-\lambda R}$ is larger.

3.2.3 **Conclusion**. In the real ad system, one can expect (a), the read rate to be not too fast and refresh times to be not too large (i.e., smaller value of λR) (b) the ads along the sequence to be relatively well-distributed (thus closer to the assumption $\mu_i = \nu_i, \forall i$), and (c) $pCTR$s decreasing along the ad sequence, since the auction selects winners based partially on $pCTR$. As seen above, all three factors lead to either no change or an increase in clicks. Our experiments show no change in clicks, or even a slight increase, which indicates that these factors likely hold on the average in our system. We note that in the related work [10], it was reported that an increase in refresh time led to a decrease in clicks. The analysis above may give an explanation for the conflicting results. While our results are averaged over many apps, the results in [10] were on specific apps controlled by the author, and one may conjecture that the balance between the relevant parameters turned out to decrease clicks in that setting.

4 THE SESSION AUCTION PROBLEM

In the previous section, we proposed a new click model to explain the observations from our data and experiments. We now ask the question of whether we can improve our ad allocation and pricing in light of this model; in particular, how do we auction out the entire user's session in a global manner, rather than per impression.

4.1 Problem Definition

For a given user-session of duration T, we have as input a set A of candidate ads and for each ad a in A, its base CTR prediction ($pCTR_a$) and its per-click bid ($BidCpC_a$). The ad's value-per-click v_a is private to the advertiser, and unknown to the auctioneer. The auctioneer has to choose a sequence of ads and time durations, $S = ((a_1, t_1), (a_2, t_2), \ldots, (a_s, t_s))$, with $\sum_{i=1}^{s} t_i \leq T$ (ads are allowed to repeat in S). Given such a sequence, the ads receive clicks according to the model defined in Sec. 3. The economic efficiency (aka *value*) of the sequence S is defined as

$$v(S) := \sum_{i=1}^{s} v_{a_i} \times \Pr\left(\text{Click on } (a_i, t_i) \mid (a_1, t_1), \ldots, (a_{i-1}, t_{i-1})\right)$$

The goal is to determine a sequence S to maximize the total session value, and also to price the ads in order to elicit truthful bids (i.e., $BidCpC_a = v_a, \forall\ a$). Furthermore, the problem is *online* since the auctioneer does not know T, i.e., when the user will switch out from the app and end the session. Thus the implementation must be: At time 0, choose (a_1, t_1); subsequently, if the session does extend to t_1, determine (a_2, t_2), and so on.

Minimum Impression Duration: The value $v(S)$ above is the sum of the surplus (v_{a_i} minus payment p_{a_i}) to the advertiser and to the app publisher (p_{a_i}), but does not account for user value. In our context, rapid refreshing of ads leads to a bad user experience, and also to increased cell bandwidth and battery usage. To capture this notion of user value, we introduce a constraint on allowable sequences S, via a min impression duration parameter τ_{min} (typically 30 seconds in practice):

$$\forall i \in [1, s] : \ t_i \geq \tau_{min}$$

Remark 1. *While the problem is formulated in the context of the Read-Process model, we only use two consequences of the model: (1) the probability of a click as a function of time on screen follows an exponential distribution, i.e., Eq.* (1)*, and (2) the probability of a click on either of two impressions of an ad, with durations t_1 and t_2 is identical to that on one longer impression with duration $t_1 + t_2$, i.e., Eq.* (2)*.*

We will also assume that the Poisson parameter λ is known; it can be estimated from ad click time data, by fitting an exponential CDF to the curve in Fig. 1.

4.2 Offline Allocation

In this section we study the structure of the optimal offline allocation, when the session length T is known in advance.

Consider a feasible sequence $S = (a_1, t_1), \ldots, (a_s, t_s)$, with $\sum_i t_i \leq T$. From Eq. (2), we see that we can assume *wlog* that the ads $a_1, ..., a_s$ are all distinct: If not, we can transform S via a series of swaps to move two occurrences of an ad into consecutive positions (and merge them into one longer impression), and not lose clicks or value. Thus the offline problem greatly simplifies to the following (relatively standard) problem of *auctioning a divisible good* (see, e.g., [6]): We have T units of a divisible good, and a set of agents (advertisers) A. If $a \in A$ is allocated t_a amount of the good, then its value is $v_a(t_a) := BidCpC_a \times pCTR_a(1 - e^{-\lambda t_a})$. The goal is to allocate the good among the agents and maximize the sum of their values. The minimum impression duration constraint becomes: $\forall\ a$, either $t_a \geq \tau_{min}$ or $t_a = 0$. We can then use VCG pricing in conjunction with the optimal allocation.

Consider building an optimal sequence, and consider an ad a which has already been allocated time t. The marginal value of a is defined as the rate of additional value generated by showing a for an infinitesimally longer amount of time, and it is $m(a, t) := BidCpC_a \times pCTR_a \times \lambda e^{-\lambda t}$. By a standard argument of equalizing the marginal values of the agents to some level m, we can show that the optimal solution takes the following form. Allocate $a_1, \ldots, a_k$ for times $t_1, \ldots, t_k$ (each $\geq \tau_{min}$) with $\sum_{i=1}^{k} t_i = T$, s.t., $\exists\ m$ with:

- $\forall i \in [1, k], s.t.\ t_i > \tau_{min} : m(a_i, t_i) = m$,

– Start with time $t = 0$. For each ad a, let t_a be the total time ad a has been shown to the user so far in the session (over all impressions of a). Initialize $t_a = 0,\ \forall a \in A$.
– While the session is not finished: At time t,
(1) Let a_t be an ad with highest marginal value
$$a_t \in argmax_{a\in A}\ BidCpC_a \times pCTR_a \times e^{-\lambda t_a}$$
Let m be the marginal value of a_t.
(2) Let m' be the next highest marginal value:
$$m' = max_{a\in A\setminus\{a_t\}}\ BidCpC_a \times pCTR_a \times e^{-\lambda t_a}$$
(3) Define d_t as the amount of time that we can show a_t until its marginal value equals m', i.e.,
$$m \times e^{-\lambda d_t} = m' \Rightarrow d_t = \tfrac{1}{\lambda} \ln \tfrac{m}{m'}$$
(4) Show ad a_t for $\hat{d}_t := \max\{d_t, \tau_{min}\}$ time.
(5) Update $t = t + \hat{d}_t$, and $t_{a_t} = t_{a_t} + \hat{d}_t$.

Figure 4: Online Allocation Algorithm (Sec. 5).

- $\forall i \in [1,k], s.t.\ t_i = \tau_{min} : m(a_i, t_i) \le m$,
- for all other ads a_j (with $t_j = 0$): $m(a_j, 0) \le m$.

Note that the optimal allocation may show ads with a marginal rate strictly lower than that of another shown ad. This happens due to the minimum duration constraint, and makes our analysis of the optimal allocation (in the sections below) rather challenging. The special case of $\tau_{min} = 0$ has a simple structure. For a given $m > 0$, for each a, set t_a to be the value which gives $m(a, t_a) = m$, or set $t_a = 0$ if $m(a, 0) < m$. Now find, via binary search, that value m^* for which $\sum_a t_a = T$.

Example 4.1. Offline allocation and pricing with $\tau_{min} = 0$: Suppose there are two ads, each with $pCTR = 1$, and with bids $b > 1$ for the first ad and 1 for the second. Then the offline solution first allocates time t for the first ad, at which point the two ads' marginal utilities are equal, i.e. $be^{-\lambda t} = 1$, i.e., $t = \ln(b)/\lambda$. If b is large enough such that $t \ge T$, then only the first ad is shown. Otherwise, the solution splits the remaining $T - t$ time equally between the two ads. So overall, first ad is shown for $(T+t)/2$ time, and second ad for $(T-t)/2$ time. The first ad's VCG price is its externality on the second ad, which is $e^{-\lambda(T-t)/2} - e^{-\lambda T} = e^{-\lambda T/2}\sqrt{b} - e^{-\lambda T}$.

5 ONLINE ALLOCATION ALGORITHM

We now present our Online Algorithm, whose challenge is to determine the winning ad and its duration in an online manner, without knowing the session duration T. Fig. 4 describes this natural algorithm: pick a highest marginal value ad and shows it for a duration of time at which its marginal value drops to the second highest marginal value (and at least a τ_{min} time). For simplicity we assume[11] $\tau_{min} > 0$.

[11]If $\tau_{min} = 0$, any online algorithm must be sub-optimal due to self-imposing a small min duration for making progress in the session.

5.1 Competitive Ratio

5.1.1 **Comparing to the correct OPT**. To understand the performance of the online algorithm ALG, we compare it to the optimal offline allocation OPT and bound the worst-case competitive ratio. Since OPT can have a complex structure, one option would be to compare ALG to an optimal allocation $OPT^{(\tau_{min}=0)}$ without the minimum duration constraint, having a simpler structure and possibly leading to a clean analysis (it is a valid comparison since $OPT^{(\tau_{min}=0)} \ge OPT$). However, the next example shows that the minimum duration constraint intrinsically makes the problem harder: the $OPT^{(\tau_{min}>0)}$ can be arbitrarily worse compared to $OPT^{(\tau_{min}=0)}$, for large τ_{min}.

Notation: For conciseness, we will denote $L = e^{-\lambda\tau_{min}}$.

Example 5.1. Consider a large number n of ads, all with identical values v and a total session time of τ_{min}. Any solution will show one of the ads for the entire time, for a value of $v(1-L)$. The unrestricted ($\tau_{min} = 0$) OPT will divide up the time equally between the n ads, getting a value of $nv(1 - L^{\frac{1}{n}})$, tending to $v\lambda\tau_{min}$ as $n \to \infty$. Thus the ratio of the two OPT values is $\frac{1-L}{\lambda\tau_{min}}$, that tends to 0 as $\tau_{min} \to \infty$.

A fairer comparison is thus to the offline optimal allocation with the same τ_{min} constraint. We start with an example with $\tau_{min} > 0$ where the online ALG is strictly sub-optimal, *even if the session length is a multiple of τ_{min}.*

Example 5.2. Consider session length $T = 4\tau_{min}$ and two ads with values v_1 and v_2 where $v_1 L^{-2.4} = v_2 L^{-1.6}$, implying $v_1 = v_2 L^{0.8}$. The offline optimum shows ad 1 for $2.4\tau_{min}$ and ad 2 for $1.6\tau_{min}$, which equalizes their marginal values. In the online algorithm, ad 1 is shown for τ_{min} time, then its marginal value becomes $v_1 L^{-1} < v_2$, after which ad 2 is be shown for τ_{min} time, and so on, giving an allocation of (v_1, v_2, v_1, v_2), each for τ_{min} time. This results in a strictly lower value compared to the offline optimum above.

5.1.2 **Algorithm Discretization.** To get a better handle on the analysis, we will slightly degrade the performance of the algorithm by a small discretization. Recall that in Step (3), we compute the duration d_t at which the marginal utility of the winner reduces to that of the runner-up, and we show the winner for d_t time (or τ_{min} if d_t is smaller). Here, we modify the algorithm to round d_t up to the nearest multiple of τ_{min} and show the winner for that time duration. We will use this discretized algorithm henceforth in the analysis. The following result shows that our online algorithm loses less than 10% in the worst case versus the offline optimum.

THEOREM 5.3. *The (discretized) Online Allocation Algorithm achieves a competitive ratio of $\frac{1}{1+L} + \frac{L(1-L)}{(1+L)\lambda\tau_{min}} \ge 0.901$ (where $L = e^{-\lambda\tau_{min}}$). There is a problem instance in which the Online Algorithm's ratio is no better than 0.963.*

The competitive ratio goes to 1 as $\tau_{min} \to 0$ as well as when $\tau_{min} \to \infty$ (e.g., the ratio is more than 0.99 with $\lambda\tau_{min} = 5$). Intuitively, for a fixed λ: (a) as $\tau_{min} \to 0$, there is no constraint, and ALG can pick very short impressions of

highest marginal value, and (b) as τ_{min} grows large, every impression of length τ_{min} or more has a very high probability of getting a read, and hence OPT loses any advantage from carefully choosing ad durations which are not multiples of τ_{min}. Computer plots (omitted due to space limits) have shown that the minimum value of the competitive ratio (which is at least 0.901) is obtained at $\lambda\tau_{min} \simeq 0.989$.

5.1.3 **Analysis.** We now prove Theorem 5.3; specifically:

- Prop. 5.4 shows that ALG is a prefix of a natural order $\mathcal{O}$ on potential impressions from all ads
- Lemma 5.5 shows that potential full-length impressions in OPT are also a prefix of $\mathcal{O}$
- Theorem 5.3 follows by separately comparing ALG to OPT's partial-length and full-length impressions.

For the analysis, we define a specific ordering over segments of the ads timelines. For each ad a, we divide the potential time that a can be shown into *segments* of length τ_{min}. For positive integers k, let a^k denote the kth segment of ad a; this is the segment that corresponds to the time duration $[(k-1)\tau_{min}, k\tau_{min})$ that a is shown for (this may or may not be contiguous with previous segments). Define the *score* of the segment a^k, as $score(a^k) = BidCpC_a \cdot pCTR_a \cdot \lambda \cdot L^{k-1}$. This is the marginal value of starting to show a for additional time, assuming it was already shown for $(k-1)\tau_{min}$ time previously; the total incremental value provided by showing a^k is $BidCpC_a \cdot pCTR_a \cdot (L^{k-1} - L^k) = score(a^k)(\frac{1-L}{\lambda})$.

Define $\mathcal{O}$ as the sorted order on all segments a^k (over all ads a, and all positive integers k), sorted by their score in descending order. Define $\mathcal{O}_a$ as the restriction of $\mathcal{O}$ to segments of ad a. We can think of ALG and OPT as both picking segments from $\mathcal{O}$, where the segments can be *integral*, i.e., show the entire segment of duration τ_{min}, or *fractional*, i.e., show only a prefix of the segment. From the definition of ALG and of $\mathcal{O}$, we immediately get:

Proposition 5.4. *ALG picks a prefix of $\mathcal{O}$, where all but the last prefix segment are integral, and the last segment is fractional if and only if total time T is not a multiple of τ_{min}.*

Now suppose OPT shows ad a for $\hat{t}_a \in [k\tau_{min}, (k+1)\tau_{min})$ time (k a positive integer). Divide this time duration into k integral segments of τ_{min} time, and at most one fractional segment (of length at most τ_{min}). It is clear by construction of $\mathcal{O}$ that, for each ad a, OPT picks a prefix of the order $\mathcal{O}_a$, with the last segment being possibly fractional. Note that OPT can have as many fractional segments as the distinct ads it picks, while ALG has at most one fractional segment.

Unlike ALG, OPT itself may not be a prefix of $\mathcal{O}$. Denote by OPT_I the integral segments picked by OPT, and OPT_F the fractional segments picked by OPT. We get a handle on OPT by studying the structure of OPT_I and OPT_F.

Lemma 5.5. *OPT_I is a prefix of $\mathcal{O}$.*

For brevity, we omit the proof (standard switch argument).

Order the fractional segments in OPT_F by the score of the corresponding segment in $\mathcal{O}$, which recall, is defined as the marginal value at the start of the segment. It is possible that some integral segment p_1 not chosen by OPT may have a higher score than the score of some fractional segment $p_2 \in OPT_F$. This can happen if p_1 was the first segment of the corresponding ad, which must be picked integrally (due to the minimum duration constraint), and picking other fractional segments including p_2 instead leads to larger value. We now proceed to bound OPT_I and OPT_F by ALG.

Corollary 5.6. *For each ad i if OPT picks it k integral times (and possibly one more fractional impression), then ALG picks it at least k integral times.*

The corollary follows immediately from Lemma 5.5, the fact that ALG picks exactly a prefix of $\mathcal{O}$ (with the last segment possibly fractional), and that both ALG and OPT have to cover the same total time T. This corollary says that ALG picks all the segments in OPT_I; let ALG_1 denote this part of ALG, and let ALG_2 denote the rest of the integral segments picked by ALG as well as the at most one fractional segment that ALG may have picked. By definition,

$$ALG_1 = OPT_I \tag{5}$$

The total time used by ALG_2 is equal to the time used by OPT_F. Now we know that ALG_2 is a prefix of $\mathcal{O}\backslash OPT_I = \mathcal{O}\backslash ALG_1$. Therefore we can match every segment p in ALG_2 to a set F_p of fractional segments in OPT_F, such that (a) the total time duration of the segments in F_p is equal to the duration of the segment p (which is τ_{min}, except for the last segment if that is fractional), and (b) $score(p) \geq score(p'), \forall\ p' \in F_p$. This can be achieved by matching the next segment p in ALG_2 (according to $\mathcal{O}$), to the next set of fractional segments in OPT_F (again according to $\mathcal{O}$) which sum up to a time equal to that of p.

The total value of p is $score(p)(\frac{1-L}{\lambda})$. The total value in F_p is at most $score(p)\tau_{min}$, which can be achieved if each segment in F_p is infinitesimally small. (For the last segment of ALG_2, if it is fractional, then we get a similar and in fact a more favorable, comparison). Thus

$$ALG_2 \geq \frac{1-L}{\lambda\tau_{min}} OPT_F \tag{6}$$

Furthermore, for every ad, OPT has to pick at least one integral segment in order to pick a fractional segment, i.e., $|OPT_I| \geq |OPT_F|$ (in Ex. 5.2 $|OPT_I| = 3, |OPT_F| = 2$). Since $ALG_1 = OPT_I$ and $ALG_2 \leq OPT_F$, we want to maximize the proportion of OPT_F in OPT if we want to minimize the competitive ratio. This happens when OPT has exactly one integral segment and one almost integral (i.e., of length $\tau_{min} - \epsilon$ for small $\epsilon > 0$) segment per ad. This gives[12]:

$$OPT_F \leq L \cdot OPT_I \tag{7}$$

Putting (5), (6), and (7) together, we get:

$$\frac{ALG}{OPT} \geq \frac{1}{1+L} + \frac{L(1-L)}{(1+L)\lambda\tau_{min}}$$

which is minimized at $\lambda\tau_{min} \approx 0.99$ with a minimum value of at least 0.901. This proves the lower bound on the competitive ratio in Theorem 5.3.

[12]We thank Zun Li (lizunks@sjtu.edu.cn) for pointing out an error in a previous version of this bound.

5.1.4 **Upper Bound.** There are two ads a_1, a_2 with value v and one additional ad a_3 with value $vL + \epsilon$, for a small $\epsilon > 0$. The total time is $3\tau_{min}$. ALG picks a_1, a_2, a_3 for τ_{min} time each (one integral segment for each ad). OPT picks a_1 and a_2 for $1.5\tau_{min}$ time each (one integral and one fractional segment of size $0.5\tau_{min}$ for a_1 and a_2). Then

$$\frac{ALG}{OPT} = \frac{2(1-L)+L(1-L)}{2(1-L)+2L(1-L^{\frac{1}{2}})}$$

It can be shown that this value is minimized at $\lambda\tau_{min} \approx 1.108$ (i.e. $L = 0.3302$), giving an upper bound of approximately 0.963, proving the upper bound in Theorem 5.3.

5.2 Pricing

We use VCG pricing at the end of the session: if we remove agent i, then other agents will increase their share of the divisible good. The difference in their total values is i's VCG price. As a technical point, we note that while the allocation of the online algorithm is not optimal, it is *Optimal-in-Range* since it optimizes over allocations in which each ad has a duration equal to a multiple of τ_{min} (here we assume that T itself is a multiple of τ_{min}). Hence the VCG scheme retains its incentive compatibility (IC) properties. Pricing must be done at the end of each session because per-impression pricing, which although practically convenient, cannot be made IC.

6 ONLINE AUCTION: EXPERIMENTS

We implemented and experimented with the online allocation algorithm from Sec. 5. We now present the experiment results comparing this to the baseline of a fixed refresh rate.

Experimental setup. From a representative subset of mobile devices, our study randomly assigns each one with probability 0.5 to either control or treatment. A given device only shows ads with the same algorithm for setting refresh rate: fixed default rate if device is in control and the optimal online policy if in treatment. In order to limit the ad load on users, we set τ_{min} to be no greater than the default refresh rate. We conduct the experiment over three months.

Note that the algorithm may make very different allocation decisions compared to the control which picks the highest value ad and shows for a fixed time period: (a) it may choose different winners since it correctly normalizes an ad's $pCTR$ based on the ad's total time from past impressions in the session, and (b) it may show the winner for a different, dynamically determined, duration derived by comparing its marginal value to that of the runner up.

Practical issues limited our online policy implementation:

- User actions (in-app navigation or opening a new app), may yield lower actual durations than intended.
- In a running system, ad selection is affected by the model responsible for predicting CTR. Due to the radical change in the algorithm, the pCTR is not always accurate for the treatment arm. This reduces the efficacy of the optimal algorithm. In a stable production (not experiment), we can normalize the pCTR based on the time-on-screen feature.
- Policy requires static refresh rate for some ad types.

Table 1 discusses the relative change in key ad metrics when refresh rate is set using the online allocation algorithm. For the Economic Efficiency metric, we assume that the $BidCpC$ of each advertiser is also its value-per-click, which is reasonable since we price to make the auction IC.

Metric	Change	Interpretation
Impressions	−19.9% ± 0.4%	Impressions decrease as the algorithm tends to show each impression for longer than the control, since the duration is based on runner-up's value.
Clicks	+1.2% ± 0.6%	Note the modest change despite the large decrease in ad impressions. This is consistent with our earlier findings in Sec. 2.4.
CTR	+28.2% ± 0.8%	Clicks have a modest increase, but impressions have a significant decrease.
Economic Efficiency (Value)	+8.3% ± 1.1%	This significant increase in the optimization objective (with same session duration) shows the power of our approach.
Revenue	+2.3% ± 1%	Generally value and revenue increases are comparable. In this instance, revenue grows less, likely because the pCTR model has not fully caught up, under-predicting CTR of runner-up ads (used for pricing).

Table 1: Relative differences in metrics when setting the refresh rate using the optimal online allocation.

Besides the improvements in advertiser and publisher metrics, refreshing ads less often reduces the load from ad calls on the user device and on ad provider servers.

7 CONCLUSIONS

We conducted a large-scale study on the effect of ad refreshes on user click behavior in the context of ads in mobile apps. Motivated by the experimental data, we proposed a new click model that can explain these findings, and identified key factors that determine how changes in refresh rates would affect CTR. Based on this model, we developed a framework for session auctions, and constructed an online algorithm guaranteed to be within 10% of the optimal ad sequence, even if one knows beforehand how long the session is. A live experiment demonstrated the potential of this approach in reducing ad load and improving economic efficiency.

There are interesting directions for extending this work. If we could better determine when users pay attention to ads, we could further optimize economic efficiency. Also, this suggests that we should discount time on screen when users are highly engaged with the source app. A competitive auction without the non-cumulative effect assumption would be relevant for ad formats (like video ads) suitable for brand awareness.

REFERENCES

[1] Gagan Aggarwal, Jon Feldman, S. Muthukrishnan, and Martin Pál. 2008. Sponsored Search Auctions with Markovian Users. In *Internet and Network Economics, 4th International Workshop, WINE 2008, Shanghai, China, December 17-20, 2008. Proceedings*. 621–628. https://doi.org/10.1007/978-3-540-92185-1_68

[2] Gagan Aggarwal, Ashish Goel, and Rajeev Motwani. 2006. Truthful auctions for pricing search keywords. In *Proceedings of the 7th ACM conference on Electronic commerce*. ACM, 1–7.

[3] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. 2007. Internet Advertising and the Generalized Second-Price Auction: Selling Billions of Dollars Worth of Keywords. *American Economic Review* 97, 1 (March 2007), 242–259. https://doi.org/10.1257/aer.97.1.242

[4] Daniel G. Goldstein, Randolph Preston McAfee, and Siddharth Suri. 2012. Improving the Effectiveness of Time-based Display Advertising. In *Proceedings of the 13th ACM Conference on Electronic Commerce*. 639–654.

[5] Jiaping Gui, Ding Li, Mian Wan, and William G. J. Halfond. 2016. Lightweight Measurement and Estimation of Mobile Ad Energy Consumption. In *Proceedings of the 5th International Workshop on Green and Sustainable Software (GREENS '16)*. ACM, New York, NY, USA, 1–7. https://doi.org/10.1145/2896967.2896970

[6] Ramesh Johari and John N Tsitsiklis. 2004. Efficiency loss in a network resource allocation game. *Mathematics of Operations Research* 29, 3 (2004), 407–435.

[7] Ramesh Johari and John N Tsitsiklis. 2009. Efficiency of scalar-parameterized mechanisms. *Operations Research* 57, 4 (2009), 823–839.

[8] David Kempe and Mohammad Mahdian. 2008. *A Cascade Model for Externalities in Sponsored Search*. Springer Berlin Heidelberg, Berlin, Heidelberg, 585–596. https://doi.org/10.1007/978-3-540-92185-1_65

[9] H. Brendan McMahan, Gary Holt, D. Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, Sharat Chikkerur, Dan Liu, Martin Wattenberg, Arnar Mar Hrafnkelsson, Tom Boulos, and Jeremy Kubica. 2013. Ad Click Prediction: a View from the Trenches. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*.

[10] V. N. X. Truong. 2016. Optimizing mobile advertising using ad refresh interval. In *2016 International Conference on Electronics, Information, and Communications (ICEIC)*. 1–4. https://doi.org/10.1109/ELINFOCOM.2016.7562948

[11] Narseo Vallina-Rodriguez, Jay Shah, Alessandro Finamore, Yan Grunenberger, Konstantina Papagiannaki, Hamed Haddadi, and Jon Crowcroft. 2012. Breaking for Commercials: Characterizing Mobile Advertising. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference (IMC '12)*. ACM, New York, NY, USA, 343–356. https://doi.org/10.1145/2398776.2398812

[12] Hal R Varian. 2007. Position auctions. *international Journal of industrial Organization* 25, 6 (2007), 1163–1178.