Yield Optimization with Binding Latency Constraints

Dmitri I. Arkhipov*, John G. Turner†, Michael B. Dillencourt*, Paul L. Torres‡ and Amelia C. Regan*
*Donald Bren School of Information and Computer Science, University of California Irvine
†Paul Merage School of Business, University of California Irvine
‡Solon Technologies, 5151 California Avenue Suite 100, Irvine CA 92617
Corresponding author: D.I. Arkhipov (darkhipo@ics.uci.edu)

***Abstract*—Programmatic advertising is an actively developing industry and research area. Some of the research in this area concerns the development of optimal or approximately optimal contracts and policies between publishers, advertisers and intermediaries such as ad networks and ad exchanges. Both the development of contracts and the construction of policies governing their implementation are difficult challenges, and different models take different features of the problem into account. In programmatic advertising decisions are made in real time, and time is a scarce resource particularly for publishers who are concerned with content load times. Policies for advertisement placement must execute very quickly once content is requested; this requires policies to either be pre-computed and accessed as needed, or for the policy execution to be very efficient. In this paper we formulate a stochastic optimization problem for per publisher ad sequencing with binding latency constraints. We adopt a well known heuristic optimization technique to this problem and evaluate it's performance on real data instances. Our experimental results indicate that our heuristic algorithm is near optimal for instances where an optimality calculation is feasible, and superior to other reasonable approaches for instances when the calculation is not feasible.**

***Keywords*—online advertising; ad scheduling; heuristics.**

I. Introduction

In programmatic advertising when a web user's browser requests an advertisement from a publisher's content server that server or the user's browser (depending on system design choices) sends an ad request to the publisher's ad server. The publisher's ad server may then use an ad network as a broker to sell the available ad space to advertisers. The ecosystem is fairly complex and involves multiple interactions with varied agents, often with competing interests.

The added value that the ad network provides is access to its in-network advertisers. In our use In-network means that the ad network has an established programmatic communication channel with this set of advertisers. Those web content servers that use an ad network are referred to as in-network publishers. Thus, an ad network brokers ad sales between in-network publishers and in-network advertisers in real-time. The contract between the ad network and each in-network advertiser specifies the price at which the in-network advertiser will buy ad space but does not require an advertiser to buy each ad space offered to it by the ad network.

In this paper we use the term "advertiser" loosely: in our use it references an entity which is interested in purchasing ad space at a deterministic price. In industry terms this is most direcly applicable to "preferred contracts", but the term applies also to ad space sales on a real time bidding (RTB) platform, or to "direct sales". The reader of this work is perhaps best served by thinking of an "advertiser" simply as a server which decides to purchase or not purchase ad space on solicitation with the knowledge that the situation generalizes usefully to more complex arangements that exist in the industry.

The contract between the ad network and each in-network publisher specifies the proportion of the revenue (usually specified in cost per millis (CPM)) earned by the ad network in selling the ad space to be filled by content generated and served by the in-network publisher to any in-network advertiser.

The contract between the ad network and each in-network publisher further specifies the maximal allowable servicing-time/latency for each ad request. Here both servicing-time and latency refer to the time spent in finding an advertiser willing to buy the ad space indicated by the ad request. This constraint ensures a guaranteed content load time. The ad network imposes a formula for calculating penalties for service-times exceeding the deadlines; the size of the penalty is positively related to the lateness of the service time of the ad request.

When a sufficiently long delay is detected the ad network will return a "house-ad" thus terminating the ad request. A house-ad is an advertisement for which the ad network is not paid; it is an ad that is generated as a fall back option. Generally, a house-ad advertises the ad network itself (for example "we can sell your ad here", or a service owned by the same parent company) and the value of deploying a house-ad is considered to be negligible. Without loss of generality we assume that deploying house-ads will generate no profit for the ad network.

Displaying a house-ad indicates the failure to find a willing in-network advertiser to purchase a given ad request in the allotted time. This means in effect, that the ad network purchased the ad space. However, we assume that any loss associated with this transaction is negligible and assign it a value of 0 to simply indicate the lack of revenue.

The penalty for exceeding the maximum allowable servicing-time is large enough to ensure that extending service time of the ad request beyond this point will reduce the revenue earned in the event of success. Two classes of applicable penalty functions are:

1) A constant penalty for exceeding the deadline: The penalty is paid regardless of whether the last solicited advertiser has responded.
2) A non-constant penalty for exceeding the deadline: Paid after the last solicited advertiser responds.

A constant penalty is consistent with the policy of showing a house ad immediately as the deadline is exceeded. A non-

constant penalty is consistent with always waiting for the last solicited advertiser to respond. A non-constant penalty can be charged by the publisher (explicitly or implicitly) for introducing delays and lowering the web user's experience. A non-constant penalty function can thus ensure that a small delay shouldn't be penalized excessively, but a large delay can be prohibitive, so the ad network will reject solicitations that are likely to incur such a prohibitive penalty. In this paper we explore the case of a constant penalty.

A. One by One Solicitation (OBOS)

The protocol between the ad network and the in-network advertisers is very simple. A web-user's browser requests data from an in-network publisher; this publisher then requests an ad from the ad network. The ad network receives the request and filters out all in-network advertisers from whom it does not expect to receive a reply quickly enough to honor the service time contract between the ad network and the publisher.

The remaining set of advertisers are known as the set of feasible in-network advertisers. The ad network will then sequentially solicit its in-network advertisers in some order with all identifying information regarding the ad request, the associated ad publisher, ad space and web-user. We refer to these sequential solicitations as "one by one solicitation" or by the acronym (OBOS). Each in-network advertiser responds with a "yes" or "no" message to indicate purchase or non-purchase of the ad request. After each unsuccessful solicitation the solicited advertiser is removed from the set of feasible advertisers. Some feasible advertisers may become "certainly unprofitable" after an attempt. After each attempt the amount of time before the contract deadline expires diminishes. Therefore some advertisers while still feasible (because they have not yet been attempted) will be "certainly unprofitable". Even were the solicitation to succeed the net revenue derived from such an advertiser would be non-positive (at some time the penalty of attempting a given advertiser-attempt will out-weigh the benefits of its success even if the successful solicitation execution experiences no delay).

The protocol terminates when either one of the solicited advertisers agrees to buy the ad space or the ad network finds that none of the remaining unsolicited advertisers are feasible and "potentially profitable". In the case where the ad space was bought, the space if filled with the purchasing advertiser's ad, and the published content is viewed by the web user. In the case where no feasible "potentially profitable" advertisers remain the ad network fills the ad space with a house ad.

The situation in which two advertisers are assigned to the same ad request is forbidden. This constraint indicates that we may not solicit an ad request to multiple in-network advertisers simultaneously. In fact we must solicit in-network advertisers sequentially, advancing to the next advertiser in the sequence only when the previous advertiser's rejection is received. Of course, once an advertiser purchases the ad request we no longer solicit other advertisers to purchase the same ad request. The greater part of the processing time for an ad request solicitation is spent at the advertiser as it processes the request, decides on a response, encodes and transmits the response. Thus, modifying the order in which the solicitations occur can greatly alter the full delay experienced by the user in loading the full publisher page.

Within the context of serving programmatic advertising there are two predominant programming models, real time bidding "RTB" platforms or "waterfall" platforms, executing the protocol we have termed "OBOS". For a given ad request an ad network will execute either "RTB" or "OBOS" in order to sell the ad space represented by the ad request.

In industry, the sequence in which advertisers are solicited for a particular ad space is called a "waterfall" or "daisy chain", and the process of creating the waterfall is known as "yield optimization". The process of executing OBOS as we have defined it is known within the industry as "executing synchronous ad calls", "daisy chaining ad tags", or "falling through a waterfall".

The problem of finding the optimal (with respect to expected value) sequence to solicit the advertisers in is a difficult challenge. Examining each possible sequence would require $N!$ operations (assuming that N is the number of in-network advertisers). Clearly this is too computationally expensive to be considered except for very small values of N. In this work we explore less computationally expensive methods for determining this optimal sequence, and a heuristic method that performs very close to optimal.

In real time bidding an RTB platform solicits multiple advertisers simultaneously and chooses the highest bidder who's bid arrived within the time limit; the winner will then pay the price bid by the second-highest bidder (a sealed-bid second-price auction).

In an RTB system:

- Advertisers must be able to assign a bid value for each ad space.
- Advertisers will not know at the time they bid whether their bid was successful or not.

B. Ad Space Valuation

Dynamically assigning a bid value for a given ad space is not an easy problem. If a bidder is bidding their true valuation as is assumed in second price auctions then the seller has the opportunity to incrementally learn the buyer's valuation.

When auctions are repeated for non-unique goods there is a high probability that similar interactions will occur. In cases where repeated auctions for similar goods are occurring it is understood that second price auctions are not incentive compatible, sub-optimal. In this situation buyers bidding their true valuations is not an equilibrium. Because truth telling is not an equilibrium, advertisers find it difficult to determine what price to bid on a given impression.

Consequently, many advertisers choose to fix a price with the ad network and pay this price each time an acceptable ad impression comes in. The advertiser only needs to accept valuations where the estimated value of the request is at least as high as the contract price.

From a system wide perspective when prices are agreed upon in advance it is easier for the ad network to prioritize who to send requests to and achieve a near optimal match between publishers and advertisers as compared to sending requests to all valid advertisers.

C. Ad Space Sale Uncertainty and Volume

Advertisers face pacing constraints for their ad campaigns. The pacing of an advertising campaign specifies the rate at which funds dedicated towards the campaign are to be spent. How closely the true amount spent matches the intended amount as specified in the pacing for a campaign is an

important measure of the quality of the ad network facilitating ad space sales. In industry terms the ad network is acting as a "Demand Side Platform" (DSP) with respect to advertisers (or "creatives"), and as a Supply Side Platform (SSP) with respect to publishers. Another important measure is the total number of ad spaces filled for a given campaign. For any given campaign, an advertiser will prefer an ad network which can maximize the number of appropriate ad spaces filled, while keeping close to the pacing of the campaign.

Even if a bid price for a particular ad space were easy to determine, an advertiser would be uncertain of the success of their bid, and thus would be unable to adjust future bid pacing quickly. As campaigns deviate from the required pace the bid amounts can increase. If the amount spent is too low, such increases can result in bidding higher than would be necessary to acquire ad space and thus decrease the total amount of ad spaces filled by the campaign. The difficulty of properly determining a bid price for an ad space increases the uncertainty of whether a particular RTB bid is successful.

In practice ad networks will often give up the potentially higher prices available from an RTB for the certainty and greater volume offered by a fixed price ad space purchaser. Greater predictability removes some of the uncertainty associated with consistently maintaining campaign pacing.

An ad network, which receives ad requests from blocks of publishers is better positioned to guarantee both quantity and pace.

D. Traffic

Finally, an added benefit of the OBOS approach over RTB is that the number of messages broadcast by the ad network is significantly reduced. Where in RTB a message is sent and received between the ad network and all applicable advertisers, in OBOS these messages are exchanged sequentially until an ad request is filled successfully. With respect to the number of messages exchanged RTB is the worst case of OBOS. Because prices are known in advance, determining a market clearing price is not necessary.

E. Summary

In summary there are several reasons why one-by-one solicitation with a fixed price may be preferable to an auction mechanism:

- Bid price selection is handled periodically and so does not need to be determined dynamically.
- Smaller deviations from pacing goals than with RTB.
- Fewer network messages are generated than with RTB.

F. Simplifications

The protocol described above diverges slightly from the situation seen in practice. Ad networks might solicit even advertisers they do not expect to respond in time. Ad networks will solicit advertisers until the service time negotiated with their publisher has nearly elapsed and at that time if the ad space has not yet been purchased the ad network will display a house ad. Doing so leads to a troubling situation where an advertiser may be in the process of deciding whether or not to purchase ad space when the ad space expires and is no longer available for purchase. A decision to purchase the ad request at such a moment will result in the inability of the ad network to fulfill the purchase. While this could be problematic, it happens less frequently than it would if publishers solicited advertisers simultaneously.

In the model that we develop in this paper we do not model this complication directly. Rather, we assume that the ad network must wait for each solicited advertiser to respond before either soliciting another advertiser or serving ad content (whether a house ad, or an advertiser ad) to the publisher. This means that in our model an ad network may violate the latency constraint by an arbitrary amount of time. We combat this simplification by assigning a penalty function that is monotonically increasing with time. We deduct this penalty from any profit generated by an ad space sale thus penalizing solutions likely to violate the latency constraint proportional to the size of the violation.

II. Related Work

Though relatively recent, web advertising is a complex and studied field. In this paper we examine one small component of the web advertising ecosystem, and in this section we review the work most related to our own. Readers with a broader focus will find Kurula et al's recent survey [1] of direct advertising interesting. Muthurishnan's excellent characterization of the mechanics of ad exchanges [2] is also an important resource. Wang and Yuan [3] provide an introduction and tutorial to the real time bidding (RTB) model, and Yuan et al [4] provide a detailed analysis of a real RTB system.

A variety of works address different aspects of the online advertising ecosystem. For example, Lang et al [5] provide efficient algorithms for selecting from valid demand-supply paths through a network of real-time bidding systems. Lang et al's work effectively addresses the problem of finding revenue valid, optimal ad space trades under conditions where advertisers are assumed to accept valid ad placements. In our work we do not make this assumption but rather model the acceptance and rejection of ad space as a choice made by the advertiser's ad server (these choices can be modeled as random variables). Another aspect where our approach differs from Lang et al's approach is that they consider a valid path as one in which certain filtering constraints are satisfied whereas we assume that filtering has occurred prior to the optimization. Our assumption is based on the specification and the speed of adoption of the OpenRTB standard [6] which standardizes the determination of advertiser-publisher validity and in many cases removes the need for explicit querying.

Our approach places greater importance on timely delivery guarantees than [5]. This is consistent with guarantees made explicit in the **tmax** field of the OpenRTB standard [6]. Finally, Lang et al [5] treat an RTB auction as the primary sales mechanism employed by ad networks in selling ad space; in fact it is often the case that an ad network will treat its set of in-network advertisers preferentially and will choose when possible to sell ad space to them directly in the manner discussed in section I. Selling ad space directly to an in-network advertiser can often be done more quickly than participating in an ad auction, and it avoids the revenue sharing discussed in [5].

Chakrabory et al [7] analyzes a related problem from the perspective of an ad exchange; in their work they develop an optimization framework accounting for both allocation (of ads to ad networks) and solicitation (which ad networks to solicit for a bid on ad request). Our research, while related

to that work, differs in several ways. They choose to model the bandwidth constraints of an auction (ad exchanges wasting bandwidth on bid solicitations that are unsuccessful are themselves less successful) by these constraints they intend to tighten solicitation selectivity (those ad networks more likely to win an ad request are more likely to be solicited with it. Of course their model has the additional objective of optimizing revenue. Their model implicitly (and reasonably) assumes that conservation of bandwidth will translate to a lower latency system (thus ensuring timely delivery).

Our model addresses the problem encountered by an ad network rather than an ad exchange. The distinction between the two entities is that an ad network usually attempts to service a set of in network advertisers prior to (or selectively for certain ad requests) using an ad exchange for running an auction. The protocol between the in network advertisers and the ad network (described in section I) specifies that solicitations must occur sequentially, rather than in an ad auction format (wherein solicitations are made asynchronously). In our work (in contrast to [7]) we address the service time constraints of the problem explicitly while asserting that sequential solicitation will naturally use less bandwidth than asynchronous solicitation. We think that modelling service time constraints is a reasonable choice as those constraints are explicitly encoded in the OpenRTB standard [6] and thus their violation could arguably be a breach of the agreement between the ad network and the in network publishers.

Mirrokni and Nazerzadeh[8] describe and analyse several families of preferred deals in comparison to reservation contracts and the RTB mechanism. Their work develops a method for constructing approximately optimal (in terms of publisher revenue) preferred deals. In this paper we discuss the subclass of preferred deals characterized as $(\rho, \mu = 0)$ (in their lexicon), that is prices per publisher, advertiser pair are fixed, and the advertiser is under no obligation to purchase any of the offered impressions. They demonstrated that the use of this class of preferred deal results in reduced revenue for the seller when compared to the use of an RTB mechanism with a buyer reserve price. However, this class of contract is not uncommon. Impression buyers can be interested in this class of deal because it provides a guaranteed impression channel, with a known price, and predictable impression rate; this is valuable when seller valuations fluctuate due to deviations from target pacing. The algorithm proposed for choosing the next buyer to solicit for a general preferred deal is at least linear in the number of advertisers, which is problematic for cases where the number of applicable advertisers is small, and timing constraints stringent. Our work explores this class of contract, with seller imposed round trip time constraints not included in the analysis of Mirrokni and Nazerzadeh.

Balseiro et al's work [9] on yield optimization develops and analyses an optimal policy for selecting a next advertiser to solicit, however their model like that of Mirrokni and Nazerzadeh doesn't account for timing constraints at the publisher. Amiri et al's work on developing advertisement schedules through a Lagrangian relaxation of the ad placement problem [10] is perhaps most related to our work in this paper in that their static orderings can be implemented easily either on ad servers or on web user's browsers. However, their formulation also does not explicitly account for delay constraints or delay stochasticity.

III. Describing the Static Problem

For ease of explanation, in our problem description we stay as close as possible to the variable labeling notation from Powell [11]. In our formulation and model we use discrete time units. While the class of distributions that we use for delay time (lognormal distributions) during our empirical evaluation is defined on the continuous time domain we discretize variable realizations into units with small fixed granularity.

At a time $t = 0$ we are assigned a set of N advertisers: $\mathcal{A} = \{1, ..., N\}$. We are given a time budget/deadline $t^{max} > 0$. We are constrained to sell the available ad space to no more than 1 of these advertisers. We offer the ad space to advertisers sequentially. We define a function that gives the penalty associated with receiving a response from the last solicited advertiser (a) solicited after t time units have elapsed since the first advertiser in our sequence was solicited. This penalty, $c\Big((t - t^{max})^+, a\Big)$ will be 0 until $t > t^{max}$ (the contract deadline has passed).

A common arrangement specifies that net-revenue is non-negative (ad networks do not pay fees for missing deadlines), however they make no profit on responses that are not filled in time. Equation 1 represents a penalty function which achieves this arrangement. While our approach works for any monotonically increasing penalty function we use the penalty function in equation 1 for our empirical evaluations in section VII.

$$c\Big((t - t^{max})^+, a\Big) = \begin{cases} 0 & : t \leq t^{max} \\ r_a & : t > t^{max} \end{cases} \tag{1}$$

Each successive solicitation will succeed or fail. A solicitation of a will succeed with some known probability p_a and will fail with probability $1 - p_a$. Success on solicitation a will provide a known revenue r_a, otherwise it provides zero revenue. Successful on solicitation a will take T_a^s time units while failure will take T_a^f time units. T_a^f, T_a^s are random variables with known means μ_a^f, μ_a^s and known standard deviations σ_a^f and σ_a^s. We must produce a policy in the form of a static sequence π in which to attempt solicitations. As soon as a single solicitation is complete, we are finished.

A statically optimal policy is a sequencing of the solicitations with maximal expected revenue. In other words, an optimal policy is a permutation $\pi = P(1, N)$ such that $\pi(i)$ indicates the order in which the i'th solicitation is to be attempted; this permutation is chosen at time $t = 0$ and remains fixed over time.

IV. Formulating the Problem

In this section we develop an objective function for the problem we have introduced. We assume that both the probability of a solicitation success, and the distributions describing the time spent in attempting it, are independent of the probabilities of any other solicitation.
Define the following independent random variables:

$$\begin{aligned} X_i &= 1 \text{ if success at solicitation } i, \text{ 0 otherwise} \\ T_i^f &= \text{time spent to fail at solicitation } i \\ T_i^s &= \text{time spent to succeed at solicitation } i \end{aligned} \tag{2}$$

Let $\overline{X_i} = 1 - X_i$.

Notice that the following is true:

$$\begin{aligned} p_i &= \mathbb{E}[X_i] && \text{probability of success at solicitation } i \\ \mu_i^f &= \mathbb{E}[T_i^f] && \text{mean failure time of solicitation } i \\ \mu_i^s &= \mathbb{E}[T_i^s] && \text{mean success time of solicitation } i \end{aligned} \tag{3}$$

Let $\rho(\pi)$ represent the net revenue from attempting a particular permutation π. $\rho(\pi)$ is defined as a sum of N terms $\psi_{\pi(i)}$:

$$\rho(\pi) = \sum_{i=1}^{N} \Big(\psi_{\pi(i)}\Big) \tag{4}$$

Each term $\psi_{\pi(k)}$ is defined:

$$\psi_{\pi(k)} = X_{\pi(k)} \cdot \prod_{i=1}^{k-1} \Big(\overline{X_{\pi(i)}}\Big) \cdot \left(r_{\pi(k)} - c\Big(T^s_{\pi(k)} + \sum_{j=1}^{k-1} T^f_{\pi(j)} - t^{max}, \pi(k)\Big)\right) \tag{5}$$

By linearity of expectation, and independence, the expected net revenue is:

$$\mathbb{E}[\psi_{\pi(k)}] = \mathbb{E}[X_{\pi(k)}] \cdot \prod_{i=1}^{k-1} \Big(\mathbb{E}[\overline{X_i}]\Big) \cdot \left(r_{\pi(k)} - \mathbb{E}\Big[c\Big(T^s_{\pi(k)} + \sum_{j=1}^{k-1} T^f_{\pi(j)} - t^{max}, \pi(k)\Big)\Big]\right) \tag{6}$$

After applying the relabellings from equation 3:

$$\mathbb{E}[\psi_{\pi(k)}] = p_{\pi(k)} \cdot \prod_{i=1}^{k-1} \Big(1 - p_{\pi(i)}\Big) \cdot \left(r_{\pi(k)} - \mathbb{E}\Big[c\Big(T^s_{\pi(k)} + \sum_{j=1}^{k-1} T^f_{\pi(j)} - t^{max}, \pi(k)\Big)\Big]\right) \tag{7}$$

Thus, the expected net revenue of a permutation/policy $\mathbb{E}[\rho(\pi)]$ is:

$$\begin{aligned} \mathbb{E}[\rho(\pi)] &= \mathbb{E}\Big[\sum_{k=1}^{N} \Big(\psi_{\pi(k)}\Big)\Big] \\ &= \sum_{k=1}^{N} \left(p_{\pi(k)} \cdot \prod_{i=1}^{k-1} \Big(1 - p_{\pi(i)}\Big) \cdot \left(r_{\pi(k)} - \mathbb{E}\Big[c\Big(T^s_{\pi(k)} + \sum_{j=1}^{k-1} T^f_{\pi(j)} - t^{max}, \pi(k)\Big)\Big]\right)\right) \end{aligned} \tag{8}$$

This expected net revenue is the optimization objective for our problem.

V. Static Optimization Formulation

In this section we define some constraints needed to make the problem computationally tractable,and to formulate the static optimization problem in terms of those constraints. Finally, we reformulate the optimization problem into a form in which the dynamic programming technique can be successfully applied to optimally solve small instances of our problem.

To make our problem computationally tractable we define certain auxiliary variables to help introduce bounds on our computation. We treat time as a discrete variable so we must set reasonable computational bounds over its domain. We define t^{a_stop} for $a \in \mathcal{A}$ where $r_a - c\Big(t^{a\_stop} - t^{max}, a\Big) = 0$; that is, each advertiser a has a time value t^{a_stop} after which it will no longer be profitable to solicit advertiser a (even if we were guaranteed a success). Let's define $t_{stop} = \max(t^{a\_stop} \mid a \in \mathcal{A})$. t^{stop} represents the number of time units past t^{max} after which it will always be unprofitable to continue attempting solicitations in set $\mathcal{A}$. The t^{max} deadline for the set of advertisers should be no more than t^{stop} time units. Attempting solicitations at or after time t^{stop} is guaranteed to be unprofitable.

We begin by defining our objective function $f(\pi)$ developed in section IV and factoring in the variables introduced to limit computation, the expected value of a policy π :

$$f(\pi) = \mathbb{E}(\rho(\pi)) = \sum_{i}^{N} p_{\pi(i)} \cdot \prod_{j<i} (1 - p_{\pi(j)}) \cdot \sum_{t=0}^{\max(\{t^{\pi(j)^{stop}} | j>i\})} \left(\Big(r_{\pi(i)} - c\Big(t - t^{max}, a\Big)\Big) \cdot Pr\Big((T^s_{\pi(i)} + \sum_{\pi(j)<\pi(i)} T^f_{\pi(j)}) = t\Big)\right) \tag{9}$$

Next we introduce the constraints and give the full static optimization problem (parameterized on set $\mathcal{A}$, and t^{max}). We label this problem $\mathbb{P}(\mathcal{A}, t^{max})$:

$$\begin{aligned} &\underset{\pi(1)\ldots\pi(N)}{\text{Max}} f(\pi) \\ &\text{Subject To:} \\ &\quad y_{i,j} \in \{0,1\} && \forall (i,j) \in \{1,...,N\} \times \{1,...,N\} \\ &\quad \pi(i) \in \mathbb{Z}_{\geq 0} && \forall i \in \{1,...,N\} \\ &\quad \pi(i) \leq N && \forall i \in \{1,...,N\} \\ &\quad \pi(j) = \sum_i i \cdot y_{i,j} && \forall j \in \{1,...,N\} \\ &\quad \sum_i y_{i,j} = 1 && \forall i \in \{1,...,N\} \\ &\quad \sum_j y_{i,j} = 1 && \forall j \in \{1,...,N\} \end{aligned} \tag{10}$$

In the optimization problem (10) above the variables $y_{i,j}$ represent the scheduling of a solicitation to advertiser i to be the $j'th$ attempted solicitation. Thus the constraint set $\forall i \in \{1,...,N\}$ $(\sum_i y_{i,j} = 1)$ represents the constraint that each solicitation is assigned to be attempted in only one position in the order. The constraint set $\forall j \in \{1,...,N\}$ $(\sum_i y_{i,j} = 1)$ represents the constraint that each position in the order in which advertisers are solicited has exactly one assigned advertiser.

VI. Solving the Static Problem Heuristically

A brute force solution to this problem is intractable for even small numbers of advertisers. The problem seems well suited to Feo, Resende, and Ribeiro's Greedy Randomized Adaptive Search Procedure (GRASP) [12] meta-heuristic. We chose the GRASP metaheuristic because it makes few assumptions about the problem structure, and does not necessitate careful consideration of primitives specific to the metaheuristic framework (such as cross-over in genetic algorithms). GRASP is also primitively parallelizable, so it can be easily scaled up to make full use of all available parallel computing power. Whereas when parallelizing a genetic algorithm implementation network topologies and solution recombination methods can significantly affect the value of the resulting solution [13]; a parallel GRASP implementation should not be affected by these attributes.

We implemented an instance of this meta-heuristic for the purpose of computing a static sequence/permutation in which advertisers will be solicited. Once the solution is calculated, advertisers are solicited in the order indicated in the solution until a time at which no advertiser can be profitably solicited. This time is calculated exactly as the value t^{stop} from section V though the set of available actions over which this value is calculated my be reduced after each solicitation. We first describe the algorithm at a high level, next algorithm 1 gives the pseudo-code for our local search. Finally, we present a description of subroutines used in our pseudo-code.

1) Our local search begins by shuffling the advertisers into a random permutation.
2) The incumbent solution (the current best) generates a fixed window (say 1000) of successor solutions by one of two methods.
3) Method 1 <Intensify>: A child solution is a clone of the parent, but the order of 2 advertisers (chosen at random) is swapped (known as 2-swap).
4) Method 2 <Diversify>: The sub-sequence between 2 points in the permutation (chosen at random) is shuffled (known as scramble).
5) When the improvement from one window to the next increases(decreases), a temperature term increases(decreases) proportionate to the window to window change.
6) A higher temperature makes method 1 (intensify) more likely, a lower temperature makes method 2 (diversify) more likely.
7) When the window to window improvement becomes lower than a threshold execution will terminate.

Algorithm 1

procedure GRASP_OBOS($\mathcal{A}$) ▷ Set $\mathcal{A}$ of advertisers.
  $x \leftarrow$PERMUTEATRANDOM($\mathcal{A}$)
  $v \leftarrow$EVALUATEPERMUTATION(x)
  $v^- \leftarrow v - \epsilon$
  $\tau \leftarrow 1.0$ ▷ τ represents temperature.
  while $(v - v^- < \epsilon)$ **do** ▷ For some small ϵ.
    $v^- \leftarrow v$
    for $i \in (0, ..., W)$ **do** ▷ W is a window size.
      if GETRANDOM( ) $< \tau$ **then**
        $x' \leftarrow$ TWOSWAP(x) ▷ Intensify.
      else
        $x' \leftarrow$ TWOSHUFFLE(x) ▷ Diversify.
      end if
      $v' \leftarrow$ EVALUATEPERMUTATION(x')
      if $v' > v$ **then**
        $x \leftarrow x'$
        $v \leftarrow v'$
      end if
    end for
    $\tau \leftarrow \tau \cdot \frac{v - v^-}{v^-}$
  end while
  return x
end procedure

In algorithm 1 we make use of several subroutines which we did not define. Below we will briefly describe these.

- **PermuteAtRandom:** Will permute the input set of advertisers into a random order.
- **EvaluatePermutation:** Returns the expected net revenue of the permutation according to equation 9.
- **GetRandom:** Returns a random value in $(0, 1)$.
- **TwoSwap:** Returns a modified copy of the passed in sequence after an application of the classic 2-swap heuristic (see for example [14]).
- **TwoShuffle:** Returns a modified copy of the passed in sequence after an application of the classic scramble primitive (see for example [15]).

In our experiments our local search solution performs roughly 1% worse than the optimal solution on problem instances; small enough that the optimal solution could be calculated reasonably quickly.

VII. Empirical Evaluation

Using real data obtained from a successful ad network we evaluated the performance of our policy over many simulated runs over each of a number of test instances. Our test suit is composed of several instances, each of which describes the set of applicable advertisers for a given publisher. Each instance is best interpreted from the point of view of a publisher. From this perspective each instance contains data relevant to choosing the next advertiser to solicit given the amount of elapsed time since the content has begun to load.

Each instance in our suite is identified by the tuple $(\texttt{publisher\_id}, t^{max})$, `publisher_id` uniquely identifies the publisher and equivalently the instance. As described in the previous sections t^{max} is the maximal allowable servicing-time specified between the publisher and their ad network. Each advertiser a described in each instance is identified by a tuple $(\texttt{advertiser\_id}, r_a, p_a, \mu_a^f, \mu_a^s, \sigma_a^f, \sigma_a^s)$. `advertiser_id` uniquely identifies the advertiser, $r_a, p_a, \mu_a^f, \mu_a^s, \sigma_a^f, \sigma_a^s$ are described in section III.

We compare the performance of `GRASP_OBOS` relative to a suite of heuristics used in practice and for small instances, and relative to the optimal dynamic programming solutions. We evaluate the efficacy of each policy by taking the average net revenue resulting from executing the policy over 10000 realized paths. A realized path refers to the state transitions and the collection of realized samples drawn from random variable distributions during the process of a simulated ad request. Following a realized path in our simulation entails sampling from distributions representing the success or failure of each solicitation of a given advertiser and the success or failure time of that solicitation. A realized path terminates either when we successfully solicit an advertiser (and collect the associated revenue), or when we exhaust our time budget for the publisher ($t > t^{max}$).

The success or failure of any given solicitation for a particular publisher is drawn from a Bernoulli distribution specific to that advertiser a, and characterised by the sufficient statistic p_a. The delay time of failure or success for a particular advertiser is drawn from a lognormal distribution characterized by the sufficient statistics (μ_a^f, σ_a^f) and (μ_a^s, σ_a^s) respectively. Lognormal distributions are commonly used to model failure and delay times of both natural and human processes (see for example [16–21]).

Two common solutions to this problem are to solicit advertisers in non-increasing revenue (this is the optimal order if the content load time was not a consideration) and to

solicit them at random. One reason that these approaches are commonly used is that they do not require collecting and aggregating the data necessary to characterize the probabilities of success, or the delay times. In our figures these approaches are labeled 'GMR'(greedy, maximal revenue) and 'RANDOM' respectively.

If the data necessary to make the characterizations were collected, an initially appealing approach would be to choose the next advertiser greedily, choosing the advertiser that maximizes the expected revenue of successfully soliciting a given advertiser (unsuccessful solicitations and their delays are not considered). We label this greedy strategy 'GMER' (greedy maximal expected revenue). Explicitly, if the time at evaluation is t, for advertiser a the expected value is:

$$p_a\Big(r_a \cdot Pr(t + T_a^s \leq t^{max}) - c\Big(t + T_a^s - t^{max}, a\Big) \cdot Pr(t + T_a^s > t^{max})\Big) \tag{11}$$

Notice that we can evaluate the probability terms in equation 11 from our Bernoulli success distributions and lognormal delay distributions (in the later case through the use of the cumulative density (CDF) function).

Finally, a slightly more sophisticated heuristic would be to greedily choose the next advertiser based on the weighted sum of two terms; one term representing the revenue of each advertiser r_a and the other representing the expected revenue of that advertiser (see equation 11). Let the weight coefficients be $\alpha_1 = \frac{t}{t^{max}}$ and $\alpha_2 = 1.0 - \alpha_1$. Our heuristic valuation will choose the advertiser a maximizing the expression:

$$\alpha_2 \cdot r_a + \alpha_1 \cdot p_a\Big(r_a \cdot Pr(t + T_a^s \leq t^{max}) - c\Big(t + T_a^s - t^{max}, a\Big) \cdot Pr(t + T_a^s > t^{max})\Big) \tag{12}$$

Equation 12 is a convex combination of our two terms. The intuition behind this heuristic is that we wish to choose the largest revenue terms early in our sequence (as would be optimal if delay time were not a factor), then as we approach the deadline we wish to take the probability of succeeding within the remaining available time more and more into account. We refer to the policy represented by this heuristic (equation 12) as 'GCC' (greedy convex combination) in figures 1 and 2.

Each test instance in test suite 1 (figure 1) contains 10 applicable advertisers; this is a sufficiently small number that we may calculate the true optimal value for the problem by a dynamic programming formulation in a reasonable amount of time. Test suite 2 contains instances with advertiser set sizes of 11 and more (tests in the figure are non-decreasing in advertiser set size). Our exact dynamic programming solution method is similar to Richard Bellman's approach to the traveling salesman problem in [22]; we omit the details due to space limitations. The results given in figure 1 are average percent deviation from optimality for each test, the x-axis represents the value of the optimal solution. Table I aggregates these results across the tested instances.

In test suite 2 (figure 2) test instances represent sets of applicable advertisers of sizes between 11 and 15 (inclusive). The tests are listed non decreasing in the number of applicable advertisers in each instance. An optimal value for tests in suite 2 could not be effectively computed by our dynamic program (this is an exponential time algorithm), thus rather than presenting the data as deviations from optimality we instead present the raw averages for each algorithm. Table II aggregates these results across the tested instances. Instances 'test-110' to 'test-76' have advertiser set size 11, 'test-253' to 'test-92' have advertiser set size 12, 'test-16' has advertiser set size 13, 'test-129' to 'test-17' have advertiser set size 14, and instances 'test-239' and 'test-354' has advertiser set size 15.

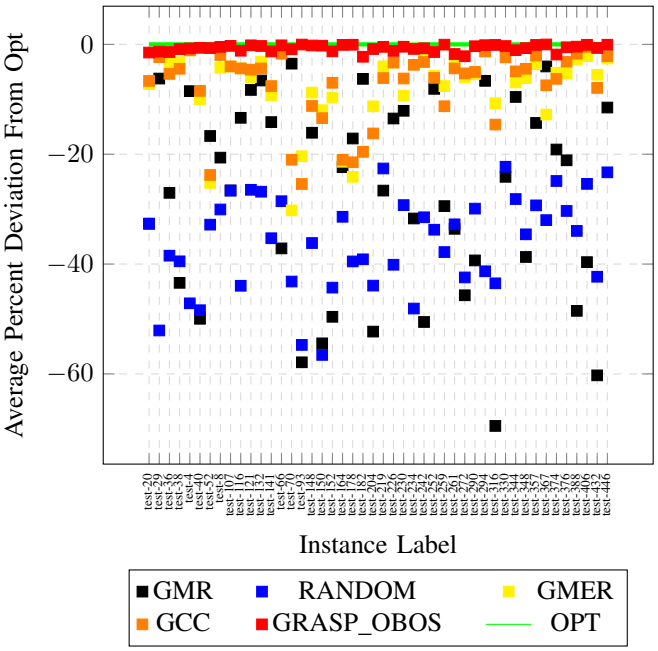

Figure 1. Advertisers sets of size 10

Table I
AVERAGE PERCENT DEVIATION OVER ALL REALIZED PATHS, OVER ALL INSTANCES

Heuristic	Average Percent Deviation
GMR	-27.14
RANDOM	-36.03
GMER	-7.71
GCC	-7.62
GRASP_OBOS	-0.67

Table II
AVERAGE NET REVENUE OVER ALL REALIZED PATHS, OVER ALL INSTANCES

Heuristic	Average Net Revenue (CPM)
GMR	0.149
RANDOM	0.146
GCC	0.255
GMER	0.249
GRASP_OBOS	0.268

VIII. CONCLUSION

While our problem can be solved by dynamic programming and while the problem can be solved off-line, for sequences

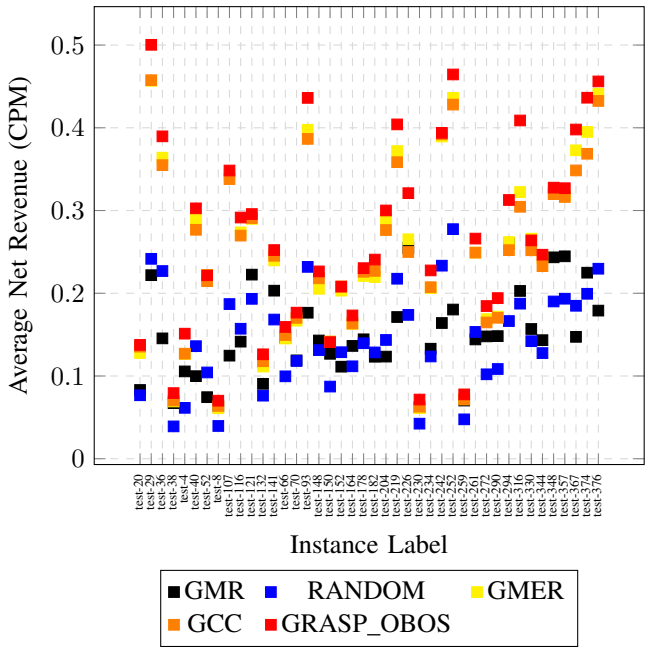

Figure 2. Advertisers sets of sizes $11, 12, 13, 14, 15$

of more than 10 advertisers the DP is not tractable. Therefore we have developed a heuristic solution for the static ordering variation of this problem. The heuristic is an adaptation of the greedy randomized search procedure (GRASP) meta-heuristic.

Our meta-heuristic solves the static ad sequencing problem efficient and we have evaluated it on an extensive data set provided to us by an ad network. Empirical evaluation shows that `GRASP_OBOS` is on average within 1% of optimal for problems which can be solved by dynamic programming. For larger problems we show that `GRASP_OBOS` outperforms any of the heuristics we know of that are being used in practice.

Our tests shows that our `GRASP_OBOS` Heuristic is close to optimal for small problems and scales well to much larger problems. Ad networks must solve hundreds or thousands of these problems every day, so scalability is an important consideration.

Acknowledgments

Dmitri Arkhipov was partially supported during this period of work by a UCCONNECT Fellowship. Amelia Regan was partially supported by the Reuben Smeed Professorial Fellowship at University College London. We gratefully acknowledge that support. Any errors or omissions are those of the authors alone.

References

[1] N. Korula, V. Mirrokni, and H. Nazerzadeh, "Optimizing display advertising markets: Challenges and directions," *Available at SSRN 2623163*, 2015.

[2] S. Muthukrishnan, "Ad exchanges: Research issues," in *Internet and network economics*. Springer, 2009, pp. 1–12.

[3] J. Wang and S. Yuan, "Real-time bidding: A new frontier of computational advertising research," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*. ACM, 2015, pp. 415–416.

[4] S. Yuan, J. Wang, and X. Zhao, "Real-time bidding for online advertising: measurement and analysis," in *Proceedings of the Seventh International Workshop on Data Mining for Online Advertising*. ACM, 2013, p. 3.

[5] K. Lang, J. Delgado, D. Jiang, B. Ghosh, S. Das, A. Gajewar, S. Jagadish, A. Seshan, C. Botev, M. Bindeberger-Ortega *et al.*, "Efficient online ad serving in a display advertising exchange," in *Proceedings of the fourth ACM international conference on Web search and data mining*, ACM. ACM, 2011, pp. 307–316.

[6] B. Riordan-Butterworth, "Openrtb api specification," Interactive Advertising Bureau, http://www.iab.net/media/file/OpenRTB-API-Specification-Version-2-3.pdf, Tech. Rep., 2014, version 2.3.

[7] T. Chakraborty, E. Even-Dar, S. Guha, Y. Mansour, and S. Muthukrishnan, "Selective call out and real time bidding," in *Internet and Network Economics*. Springer, 2010, pp. 145–157.

[8] V. Mirrokni and H. Nazerzadeh, "Deals or no deals: Contract design for online advertising," *Available at SSRN 2693037*, 2015.

[9] S. R. Balseiro, J. Feldman, V. Mirrokni, and S. Muthukrishnan, "Yield optimization of display advertising with ad exchange," *Management Science*, vol. 60, no. 12, pp. 2886–2907, 2014.

[10] A. Amiri and S. Menon, "Efficient scheduling of internet banner advertisements," *ACM Transactions on Internet Technology (TOIT)*, vol. 3, no. 4, pp. 334–346, 2003.

[11] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2007.

[12] M. G. Resende and C. C. Ribeiro, "Greedy randomized adaptive search procedures: Advances, hybridizations, and applications," in *Handbook of metaheuristics*. Springer, 2010, pp. 283–319.

[13] D. I. Arkhipov, D. Wu, and A. C. Regan, "A simple genetic algorithm parallelization toolkit (sgaptk) for transportation planners and logistics managers," in *Transportation Research Board 94th Annual Meeting*, no. 15-1361, 2015.

[14] Z. Michalewicz and D. B. Fogel, *How to solve it: modern heuristics*. Springer Science & Business Media, 2013, pp. 40–43.

[15] T. Bäck, D. B. Fogel, and Z. Michalewicz, *Evolutionary computation 1: Basic algorithms and operators*. CRC Press, 2000, vol. 1, pp. 245–246.

[16] C. J. Alpert, F. Liu, C. Kashyap, and A. Devgan, "Delay and slew metrics using the lognormal distribution," in *Proceedings of the 40th annual Design Automation Conference*, ACM. ACM, 2003, pp. 382–385.

[17] S. Clark and D. Watling, "Modelling network travel time reliability under stochastic demand," *Transportation Research Part B: Methodological*, vol. 39, no. 2, pp. 119 – 140, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0191261504000414

[18] D. A. Grove and P. D. Coddington, "Analytical models of probability distributions for mpi point-to-point communication times on distributed memory parallel computers," in *Distributed and Parallel Computing*. Springer, 2005, pp. 406–415.

[19] W. J. van der Linden, "A lognormal model for response times on test items," *Journal of Educational and Behavioral Statistics*, vol. 31, no. 2, pp. 181–204, 2006.

[20] D. Khan, N. Navet, B. Bavoux, J. Migge *et al.*, "Aperiodic traffic in response time analyses with adjustable safety level," in *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, IEEE. IEEE, 2009, pp. 1–9.

[21] D. P. Strum, J. H. May, and L. G. Vargas, "Modeling the uncertainty of surgical procedure times comparison of log-normal and normal models," *The Journal of the American Society of Anesthesiologists*, vol. 92, no. 4, pp. 1160–1167, 2000.

[22] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 61–63, 1962.