Placement Optimization

Shahar Siegman

October 2015

1 Introduction

This document discusses Placement Optimization, the heart of Komoona's business. It is not meant to be mathematically rigorous. Nonetheless, the systematic formulation provides some useful insights and can serve as a basis for further discussion as the business model evolves.
The context is implicitly assumed to be a single placement over a predefined time period, typically a single day.

1.1 Assumptions

- For each impression we receive, we need to decide which SSP to direct the impression to, and what floor price to request.
- The SSP can either serve or passback the impression.

Further, the business model is stated as:

- The publisher requests a minimum floor price (eCPM), which must be delivered.
- Komoona's goal is to maximize the *combined revenue per impression* seen by Komoona and the publisher. The floor prices Komoona passes on to the SSPs should consider the publisher's best interest along with Komoona's rev–share needs.
- The publishers monitor their eCPM and fill rate. An eCPM signifcantly higher than floor price may in some cases create dissatisfaction, as they may expect higher fill rates with eCPMs closer to their stated floor price.

2 Single–Tag Problem

2.1 Background

We start the analysis with a formulation that doesn't allow for chaining tags, i.e. we are limiting the discussion to a situation where only a single request per impression is allowed. For simplicity, the formulation doesn't consider lost impressions/discrepancy - their addition to the model is straightforward.

2.2 Notation

Let us enumerate the SSP's $1, 2, \ldots, i, \ldots, N$.
Let I_i denote the fraction of impressions served to the i'th SSP within the time period, $\sum I_i = 1$, $I_i \geq 0$. I_i is a decision variable, since we have control over which SSP each impression is passed to.
Let x_i denote the floor price passed to SSP i. The x_i's are our 2nd set of decision variables. We assume a constant floor price per SSP. We'll show later that this restriction does not limit our ability to optimize rCPM.
$fill_i(x_i)$ denotes the proportion of served (filled) requests as a function of the floor price, and similarly, $e_i(x_i)$ denotes the eCPM.
rCPM (Revenue per a thousand impressions) shall be denoted by r. The three quantities are related by:

$$r_i(x_i) = fill_i(x_i) \cdot e_i(x_i) \tag{1}$$

2.3 Problem Formulation

We can now formulate the optimization problem:

$$\begin{aligned}
&\text{Maximize} && \sum_i I_i r_i(x_i) \equiv H(I_i, x_i) \\
&s.t. \\
&\forall i : I_i \geq 0, x_i > 0 \\
&\sum_i I_i = 1 \\
&\sum_i I_i fill_i(x_i) \geq f_1, \\
&\frac{\sum I_i r_i(x_i)}{\sum I_i fill_i(x_i)} \geq e_1
\end{aligned} \tag{2}$$

Where f_1 and e_1 are, respectively, minimum fill rate and minimum eCPM constraints that we would like to impose.

2.4 Discussion

2.4.1 Problem polynomial order

The above is a constrained optimization problem. Such problems are very common in operation research. Direct solutions exist for up to quadratic optimization functions and constraints. Problems of higher polynomial degree are usually solved iteratively, by expanding the objective and the constraints around a current point, solving the quadratic problem, and repeating the expansion around the new point, until the process converges.
In our case, r_i is quadratic in x_i since it is the multiplication of an increasing and a decreasing functions of x_i (namely, e_i and $fill_i$), and can reach an extremum within the domain of interest. H has therefore third-degree interaction terms, of the form $I_i x_i^2$.
As to the constraints, the first two constraints are obviously linear. The third is quadratic ($I_i \cdot x_i$), and the fourth (after multiplying both sides in the denominator and moving all terms to the LHS), is a third-degree inequality.
It is worthwhile to note that all the decision variables are assumed continuous. In case where eCPM price quantization plays a significant role, we may need to view our x_i's as discrete, making the problem an integer programming problem. Despite the seeming complexity of solving an optimization problem with third-order terms, the specific details of this problem lend themselves to a solution procedure that's not overly complex or time-consuming.

2.4.2 Insight on Optimized Solutions

This '*chainless*' formulation may seem over-simplified, as it doesn't allow for chains, which are extensively used in real-world problems. However, even without chains, two points arise that are relevant with and without chains.

1. Since r_i is expected to be a concave function of floor price (meaning it has a maximum somewhere), the revenue cannot be improved by varying the floor price . To see why, let's say for example that we decide to randomly pass a higher floor price for half of the impressions and a lower for the other half. We would obtain a similar eCPM as in the original case, but a lower rCPM (this can be seen graphically from the concaveness of the $r_i(x_i)$ function).

   In summary, varying the floor price between impressions *within an homogeneous population*, would only serve to decrease the average revenue per impression, and so is never a desired practice.

2. The optimal solution may involve a calibrated split of the traffic between two tags. This is a non-trivial outcome. Remember, this is not about chaining tags serially. Even if we don't get a "second chance" to serve an impression, we may still want to split the impressions we get between two

SSP's. The reason, as illustrated below, is tags with optimum points outside the constraints.

Here's an example to illustrate this result: Let's say we have two SSP's. One has optimal rCPM at a very high eCPM with low fill. The fill is too low for the publisher, so if we used this SSP exclusively, we would have to lower the floor price to pull fill up to an acceptable level - taking a potentially large hit in rCPM.

The other SSP we are considering, is the "mirror image": it can deliver excellent fill rates, if we allow eCPM to be low enough. By itself, we wouldn't be able to utilize it at its optimum - we would have to push eCPM up to an acceptable level, again taking a posibly large hit in rCPM.

So, can we do better if we combine the two? By combining, we'll be able to control the total effective eCPM through the serving ratio. Let's say that the minimum eCPM we require is \$1, and that we set the first SSP to deliver an eCPM of \$5, and the second's eCPM is set to \$0.75. Every served impression of the first creates an "eCPM surplus" of \$4 — allowing us 16 deliveries of \$0.75 without breaching our eCPM and boosting our fill rate.

3 Multi-Tag Problem

3.1 Subclasses of the multi-tag problem

It is now time to extend the discussion to situations where each impression can be passed to more than one SSP for potential fulfillment. There are a few variation that are possible:

- Parallel vs. Serial - can the same impression be passed simultaneously to a few SSP's, or does the underlying mechanism requires us to pass the impression sequentially, only if not served in the previous step?
- Fixed tags vs. dynamic floor price - are we limited to a single (or a few) preset floor price(s) in each SSP, or can we dynamically decide on the floor price?

At this point in time, it seems that three out of the four combinations have real-world applications (the missing combination being sequential serving with dynamic floor price). We'll start the discussion with sequential serving.

3.2 The Sequential serving scenario

3.2.1 notation

A *chain* s is an ordered list of SSP's to which an impression is served, by order. $s_{ij} = k$ notes that the jth SSP on the ith chain is SSP with ordinal k. In order

to maintain s as a rectangular matrix, while allowing different chains have different lengths, we can define a "dummy SSP" and assign it the ordinal 0. The maximum length of a chain (within a context of a given configuration) will be denoted by l.
We use x_{ij} to denote the floor price of the jth element in the ith chain. I_i denotes the ratio of impressions served to chain i.
The extensions of r and f are more nuanced. r_{ij} and f_{ij} will aptly denote the (expected) rCPM and fill of the jth element of the ith chain; However, we now need to consider that the expected values are conditional on the previous tags:

$$r_{ij} = r_{ij}(x_1, x_2, \ldots, x_j)$$
$$f_{ij} = f_{ij}(x_1, x_2, \ldots, x_j)$$

It will also be beneficial to define a binary serving variable:

$$d_{ij} = \begin{cases} 1, & \text{if impressions was served by SSP } j \text{ in chain } i. \\ 0, & \text{otherwise.} \end{cases}$$

3.3 Problem formulation

The problem formulation is an extension of the single-tag problem.

$$\begin{aligned}
&\text{Maximize} \sum_i I_i \sum_j r_{ij}(x_{i1}, \ldots, x_{ij}) \\
&s.t. \\
&\forall ij : I_i \geq 0, x_{ij} > 0 \\
&\sum_i I_i = 1 \\
&\sum_i I_i \Big(\sum_{k=1}^{l} \Big[\prod_{j=0}^{k-1} (1 - fill_{ij}(x_{ij})) \Big] fill_{ik}(x_{ik}) \Big) \geq f_1, \\
&\frac{\sum_i I_i \sum_j r_{ij}(x_1 \ldots x_j)}{\sum_i I_i \Big(\sum_{k=1}^{l} \Big[\prod_{j=0}^{k-1} (1 - fill_{ij}(x_{ij})) \Big] fill_{ik}(x_{ik}) \Big)} \geq e1 \cdot
\end{aligned} \tag{3}$$

By definition, $fill_{i0} \equiv 0$ for any i.

3.4 Notes

The above programming problem (3) is evidently more complex than the single-tag problem (2) presented earlier. The following subsections discuss ways to either tackle or work around the induced complexity.

3.4.1 Separation of variables

The above programming problem (3) is evidently more complex than the single-tag problem (2) presented earlier. The formulation above incorporates both the *chain construction* and *chain allocation* as a joint problem. While it is conceivable to numerically solve this simultaneous problem directly, a separation-of-variables approach should have advantages as a more intuitive, more manageable alternative. The separation approach is loosely described as follows:

1. Find candidate chains which are the best performers for a particular fill level.
2. Decide on the optimal allocation that satisfies the business constraints, among these candidate chains.

This approach may yield a suboptimal solution, but it is not likely to create a significant performance gap. The allocation problem resembles formulation (2), and is actually a simplification of that formulation since the x_i's are kept constant.

3.4.2 Choice of SSP's in chain

Although the s_{ij} matrix doesn't explicitly appear in (3), it is part of the specification of the solution. s_{ij} is therefore a matrix of implicit, discrete decision variables, which results in a very high number of possible combinatorial values[1]. On the other hand, in order to achieve good performance at the placement level, we require a small number of top-performing chains. So (luckily), while we need to *consider* a very large number of potential chains, we will end up allocating actual impressions only to a few. To summarize, the problem boils down to:

1. Search in the joint $\langle s_{ij}, x_{ij} \rangle$ space for a chain whose fill and rCPM represent a better tradeoff than the candidate chains already found.
2. Add the chain to the placement's list of candidates for allocation.
3. Prune previously found chains with worse tradeoffs.
4. Repeat until no more chains can be found, or some other stopping criteria is met.

With regard to point number 1 above, Komoona's current search is based on enumerating and evaluating all $\langle s_{ij}, x_{ij} \rangle$ combinations within a certain range of floor prices and chain lengths, using some "minimal appreciable difference" as an increment between adjacent x_{ij}'s. One way to streamline this

[SS]: How do we prune the chains today?

[1] Keep in mind a particular SSP order may appear in s multiple times, each with a different floor price combination.

computationally-heavy process is to apply a numerical unconstrained optimization technique to the x_{ij} space, while still enumerating the s_{ij} space.

3.4.3 Statistical dependence between a tag's performance and preceding tags' floor prices

The notation $r_{ij}(x_{i1}, \ldots, x_{ij})$ used above, which implies the r_{ij}'s depend on the floor prices of preceding tags in the chain, is rooted in the following reasoning: The auction process, which determines the revenue we see, contains random elements. The r_{ij}'s we use in our calculations represent the *expected value of* revenue per impression, based on all relevant knowledge we have at that point. Since we reach a downchain tag only if the impression wasn't served by the preceding tags, we qualify the traffic that that tag receives, i.e. it is no longer a random sample of the impression pool. When we change the floor price of preceding tags, we alter that "filter". In other words, such change affects not only the number of impressions that reach the downchain tag, but also their distribution in terms of their inherent value for the advertisers. This idea can be concisely expressed using statistically-oriented notation (and using d_{ij} as defined above): $r_{ij}(x_{i1}, \ldots, x_{ij}) = \mathbb{E}(r_{ij} | x_1, \ldots, x_j \wedge d_{i1}, \ldots, d_{ij-1} = 0)$