Online Allocation of Display Ads with Smooth Delivery

Anand Bhalgat*
University of Pennsylvania
bhalgat@seas.upenn.edu

Jon Feldman
Google Inc, New York
jonfeld@google.com

Vahab Mirrokni
Google Inc, New York
mirrokni@google.com

ABSTRACT

Display ads on the Internet are often sold in bundles of thousands or millions of impressions over a particular time period, typically weeks or months. Ad serving systems that assign ads to pages on behalf of publishers must satisfy these contracts, but at the same time try to maximize overall quality of placement. This is usually modeled in the literature as an *online allocation problem*, where contracts are represented by overall delivery constraints over a finite time horizon. However this model misses an important aspect of ad delivery: time homogeneity. Advertisers who buy these packages expect their ad to be shown *smoothly* throughout the purchased time period, in order to reach a wider audience, to have a sustained impact, and to support the ads they are running on other media (e.g., television). In this paper we formalize this problem using several nested packing constraints, and develop a tight $(1-1/e)$-competitive online algorithm for this problem. Our algorithms and analysis require novel techniques as they involve online computation of multiple dual variables per ad. We then show the effectiveness of our algorithms through exhaustive simulation studies on real data sets.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Nonnumerical Algorithms and Problems; H.4.0 [**Information Systems**]: General

Keywords

Online Matching, Ad Allocation, Display Ads, Smooth Delivery

1. INTRODUCTION

Advertisers purchase bundles of thousands or even millions of display ad impressions from web publishers, and these bundles often represent just one piece of an overall seasonal marketing campaign. Think for example of a national clothing store: the summer campaign is carefully designed to reach as many customers as possible through similarly-themed ads on myriad different websites, TV stations and other media. It is essential that customers see a consistent message across media, and that throughout the time period customers are being exposed to ads.

Display ad serving systems that assign ads to pages on behalf of web publishers must satisfy the contracts with advertisers, respecting targeting criteria and delivery goals. Modulo this, publishers try to allocate ads intelligently to maximize overall quality (measured for example, by clicks). This has been modeled in the literature (e.g., [3, 7, 8]) as an online allocation problem, where quality is represented by edge weights, and contracts are enforced by overall delivery constraints: advertiser i may not receive more than $N(i)$ impressions. There is a $(1-1/e)$-approximate algorithm known for this problem using primal-dual techniques. While advertisers certainly appreciate getting high-quality placements, this treatment of the problem could lead to very undesirable allocations; nothing in the model prevents the publisher from showing a particular ad only on one day out of a month-long campaign, or only between 1:00 and 2:00am. *We note that, most current ad serving systems implement the smooth delivery constraint in some form to avoid such scenario, however their techniques do not achieve any bounded approximation guarantee in the adversarial case.*

Our Results and Techniques. In this paper we consider the problem of *time-homogenous* online ad assignment in a display ad serving system. We introduce *smooth delivery* constraints, where advertiser i may not receive more than $N(i,t)$ impressions before time t. The objective is to maximize the overall weight of assignment while respecting these constraints. (We give a more formal model in the next section.) We give a $(1-1/e)$-approximate algorithm using primal-dual techniques for the online ad assignment problem under arbitrary smooth delivery constraints in the *free disposal* model [7]. Building on the previous primal-dual algorithms that maintain one dual variable per advertiser, our algorithm and its analysis require new techniques for handling an entire vector of dual variables for each advertiser. Thankfully the online portion of the algorithm remains simple and can be implemented efficiently.

Our main technical contribution is how we handle the update of multiple, interacting dual variables simultaneously. In previous primal-dual algorithms, each incoming ad is assigned to the advertiser that minimizes an *adjusted* weight, equal to the edge weight minus the dual variable corresponding to that advertiser. The art is then to come up with a dual update rule that will maintain dual feasibility and a bounded gap between the primal and dual solutions. The natural extension to smoothness constraints subtracts off an entire *vector* of dual variables from the edge weight for each advertiser, and thus the design space for update rules is considerably larger, with complex interactions between the dual variables. To overcome this challenge, we derive careful accounting rules for the dual variables, where each variable corresponds to a set of impres-

*This work was done when the author was interning at Google Research, New York.

KDD'12, August 12–16, 2012, Beijing, China.

sions, and has a value equal to the exponential average of the set associated with it (or possibly the marginal value of that set over a later set). The heart of the algorithm lies in a *merge* operation where the impression sets assigned to two dual variables are joined, and we must prove a key technical lemma about a certain property of exponential average functions: for two sequences S_1 and S_2 with weighted exponential average δ_1 and δ_2, the weighted exponential average of their merged sequence is at most $\frac{\delta_1|S_1|+\delta_2|S_2|}{|S_1|+|S_2|}$.

Finally, we study the performance of our algorithms on real data for several web publishers against algorithms in [3, 7, 8], as well as some natural heuristics used in practice to ensure smooth delivery. Because *free disposal* is a stylized model largely necessitated by worst-case analysis, we look at overall metrics as well as smooth delivery metrics. Our algorithm matches [3,7,8] in the total weight of the assignment, it ensures substantially better delivery guarantees than both, and improves on the average quality of assignment (average CTR). It is contrary to perception that the delivery guarantees of our algorithm are substantially better than the heuristics used in practice for smooth delivery, as they are specifically tailored towards this objective. Thus, not only are our algorithms applicable to adversarial settings with traffic spikes, they also achieve superior performance on real-world data sets and metrics.

Related Work. Our work is closely related to online ad allocation problems, including the *Display Ads Allocation (DA)* problem [1, 6, 7, 11], and the *AdWords (AW)* problem [5, 8]. In both of these problems, the publisher must assign online impressions to an inventory of ads, optimizing efficiency or revenue of the allocation while respecting pre-specified contracts. In the AdWords(AW) problem [5, 8], advertiser i has a budget $B(i)$ on the total spend, and assigning impression j to advertiser i consumes $w(i,j)$ of his budget; whereas in the DisplayAds (DA) problem, advertiser i is associated with capacity $N(i)$, and assigning impression j to advertiser i consumes 1 unit from his capacity. In both problems, the objective is to maximize the total welfare. These problems have been studied in the stochastic as well as the adversarial arrival model. For the adversarial arrival model, a (1-1/e) approximation is known for both problems [5,7,8]. In the stochastic arrival model, a $(1-\varepsilon)$-approximation has been recently developed using primal-dual techniques [1,5,6,11]. Control-based adaptive algorithms have also been applied in the stochastic model; they achieve asymptotic optimality following an updating rule inspired by the primal-dual algorithms [10]. We note that none of the stochastic algorithms provide a guaranteed approximation in the adversarial model. As a real world application can experience unexpected traffic spikes, the desirable algorithms should be able to deal with unexpected traffic spikes while performing better for stochastic models [9].

We describe some of the other related work. Alaei *et al* [2] consider the problem of balanced allocation in the context of off-line ad allocation problem. Recently, Chen *et al* [4] gave a scheme that enables advertisers to alter their bids in real time.

We would like to mention that that none of these results consider *multiple budgets or capacity constraints* per advertiser. Our analysis requires a new set of techniques as we need to deal with computing multiple dual variables per ad in an online manner.

Organization We formally define the smooth delivery model for the Display Ads Allocation problem in Section 2. In Section 3, we describe our algorithm and prove the main result. Finally, we highlight important experimental observations in Section 4.

2. PRELIMINARIES

In this section, we formally define the Display Ads Allocation problem under *smooth delivery constraint*. There are n advertisers, $\mathbb{A}_1, \mathbb{A}_2, ..., \mathbb{A}_n$. Impressions arrive one-by-one; when an impression arrives, it can be assigned to one of the advertisers. We denote the j^{th} impression by $\mathbb{I}_j$ and the welfare of assigning $\mathbb{I}_j$ to $\mathbb{A}_i$ by $w(i,j)$. The total number of impressions is m.

The period of impressions' arrival is split into t intervals and an advertiser specifies the nature of impressions' delivery over these intervals. In particular, $\mathbb{A}_i$ does not wish to be assigned more than $N(i,k)$ impressions in first k intervals[1], for each $1 \le k \le t$. We refer to such capacity constraint as the *smooth delivery constraint*. The objective is to maximize the total welfare of the impressions assigned subject to the advertisers' capacity constraints.

We denote the number of impressions that arrive in first k intervals by $M(k)$, and the interval in which $\mathbb{I}_j$ arrives by $T(j)$. We assume $N(i,k+1) \ge N(i,k)$ for each k and $N(i,0) = 0$ for each i.

To enforce the capacity constraints, we apply the *free disposal* model [7], which says that capacity constraints are not strictly enforced during the run of the algorithm, rather they are used to determine the final value of the solution. In the smooth delivery setting, that means the following: Let S_i be the set of impressions assigned to advertiser i. Now let $S_i' \subseteq S_i$ be the maximum-weight subset such that S_i' has no more than $N(i,k)$ impressions from the first k intervals, for every k. Then, S_i' is used in the objective function to determine the value of the solution.

3. THE AD ALLOCATION ALGORITHM

In this section, we present a primal-dual based algorithm for the display ads problem with smooth delivery constraints. The LP for the offline matching problem is given below. The variable $x(i,j)$ is the fraction of $\mathbb{I}_j$ assigned to $\mathbb{A}_i$. The objective function is to maximize total welfare, the first set of constraints are the capacity constraints that guarantee the smooth-delivery of impressions, and the second set of constraints indicate that each impression is assigned to at most one advertiser.

Maximize:	$\sum_j w(i,j)x(i,j)$	
Subject to:		
For each $1 \le i \le n, 1 \le k \le t$:	$\sum_{1 \le j \le M(k)} x(i,j)$	$\le N(i,k)$
For each $1 \le j \le m$:	$\sum_i x(i,j)$	≤ 1
For each i, j :	$x(i,j)$	≥ 0

The following LP is its dual.

Minimize:	$\sum_{i,k} \beta(i,k)N(i,k) + \sum_j z(j)$	
Subject to:		
For each $1 \le i \le n, 1 \le j \le m$:	$\sum_{T(j) \le l \le t} \beta(i,l) + z(j) \ge w(i,j)$	
For each $1 \le i \le n, 1 \le k \le t$:	$\beta(i,k) \ge 0$	
For each j :	$z(j) \ge 0$	

Algorithm.

We first illustrate the overall structure of dual variables. There are t beta variables corresponding to an advertiser, one for each of t capacity constraints. A beta variable $\beta(i,k)$ goes through three phases during the course of the algorithm: (a) inactive, (b) active, and (c) dead. It is *inactive* at the start of the algorithm, it becomes *active* when the first impression in the k^{th} interval arrives and *may*

[1]In practice, the advertiser only specifies the value of total delivery and its expected delivery pattern. The display ad serving system computes the values of $N(i,k)$ for $1 \le k \le t$ based on this information. However, the algorithm and its approximation factor are independent of the way $N(i,k)$s are computed.

become dead in some later interval $k' > k$. The value of a beta variable is 0 when it is inactive or dead.

At any given state in the algorithm, for any i,k, let $F(i,k)$ be the largest integer k' such that $k' < k$ and $\beta(i,k')$ is active. If no such k' exists, then we define $F(i,k)$ to be 0. During the course of the algorithm, an active beta variable for an advertiser is associated with a (sub)set of impressions (already) assigned to him. If $\beta(i,k)$ is active when $\mathbb{I}_j$ arrives, then the set of impressions associated with it is the set of impressions assigned to $\mathbb{A}_i$ from the $(F(i,k)+1)^{th}$ interval up to the k^{th} interval. We denote this set by $S(i,k)$. *Thus the sets associated with an advertiser's active beta variables form a partition over impressions assigned to him up to that stage.*

We propose three different ways by which the value of an active beta variable is determined; and they correspond to three algorithms that we design, namely (a) smooth-greedy (b) smooth-avg, (c) smooth-exp: we compute a candidate value $\delta(i,k)$ for an active beta variable $\beta(i,k)$ in these three different algorithms as follows.

1. *Smooth-Exp*: Let $w_1, w_2, ..., w_{|S(i,k)|}$ be the weights of impressions in $S(i,k)$ in a decreasing order. Then $\delta(i,k) = \frac{1}{\left(\left(1+\frac{1}{|S(i,k)|}\right)^{|S(i,k)|}-1\right)|S(i,k)|} \sum_{1\le j\le|S(i,k)|} \left(1+\frac{1}{|S(i,k)|}\right)^{j-1} w_j$
2. *Smooth-Greedy*: $\delta(i,k)$ is the smallest weight of an impression in $S(i,k)$.
3. *Smooth-Avg*: $\delta(i,k)$ is the average weight of impressions in $S(i,k)$.

We illustrate the smooth-avg algorithm in Section 3.1 and its analysis is given in Section 3.2. The smooth-greedy algorithm is exactly same as the smooth-avg algorithm (other than computing the value of an active beta variable) and we skip its analysis for lack of space. The smooth-exp algorithm uses the same basic framework as the smooth-avg algorithm with certain crucial differences; we illustrate the differences in the smooth-exp algorithm, and its analysis in Section 3.3.

Our results for smooth-greedy and smooth-avg hold even when t is as large as m, i.e. the advertiser can specify his capacity constraint from the beginning to the arrival of each impression. For the $(1-1/e)$-approximation, the smooth-exp algorithm makes certain assumptions (to be described later).

3.1 The Smooth-Avg Algorithm

We now illustrate the smooth-avg algorithm. At the start of the algorithm, (a) for each i,k, $\beta(i,k) = 0$, (b) the set $S(i,k)$ has $N(i,k) - N(i,k-1)$ elements and each element is 0, and (c) for each $j, 1 \le j \le m$, $z_j = 0$. Algorithm 1 illustrates one iteration of the algorithm: the incoming impression is $\mathbb{I}_j$ and the current interval is k. The algorithm either assigns the impression to one of the advertisers or leaves it unassigned.

Given an iteration of the algorithm, we use $\beta_{start}(i,l), \beta_{end}(i,l)$ to indicate the value of a beta variable at the start and the end of the iteration respectively, and $\beta(i,l)$ refers its current value. We define $F_{start}(i,k)$, $F_{end}(i,k)$, $\delta_{start}(i,k)$, $\delta_{end}(i,k)$, $S_{start}(i,k)$ and $S_{end}(i,k)$ similarly to indicate corresponding quantities at the start and the end of the iteration respectively, and $F(i,k)$, $\delta(i,k)$ and $S(i,k)$ refer to their current values. We define $\Delta(\beta(i,k))$ to be $\beta_{end}(i,k) - \beta_{start}(i,k)$.

Interpretation of the Algorithm.

We now provide a simple interpretation of the algorithm. We consider an instance with 2 intervals. Up to the end of the first interval, $\beta(i,1)$ is the average weight of impressions in $S(i,1)$, i.e. the set of $N(i,1)$ *top* impressions assigned to $\mathbb{A}_i$ in the first interval. In the second interval, there are two phases.

(1) In the first phase, $S(i,2)$ is the set of *top* $(N(i,2) - N(i,1))$ impressions assigned to $\mathbb{A}_i$ in the second interval. In this phase, the average weight of impressions in $S(i,2)$ is less than the average weight of impressions in $S(i,1)$. During this phase, $\beta(i,2)$ and $(\beta(i,1) + \beta(i,2))$ are the average weights of impressions in $S(i,2)$ and $S(i,1)$ respectively. Thus $\beta(i,1)$ stores the *marginal* over $\beta(i,2)$.

(2) When the average weight of impressions in $S(i,2)$ exceeds the average weight of impressions in $S(i,1)$ for the first time, then (a) the set $S(i,1)$ merges with $S(i,2)$ and $S(i,2)$ has size $N(i,2)$ after merge, (b) $\beta(i,1)$ is dead, and (c) $\beta(i,2)$ is the average weight of *top* $N(i,2)$ impressions assigned to $\mathbb{A}_i$ so far.

Figure 3.1 illustrates these phases of the algorithm.

Algorithm 1 One Iteration of Smooth-Avg Algorithm

Find advertiser $\mathbb{A}_i$ which maximizes $w(i,j) - \sum_{k\le l\le t} \beta(i,l)$.
if $\sum_{k\le l\le t} \beta(i,l) < w(i,j)$ **then**
  Assign $\mathbb{I}_j$ to $\mathbb{A}_i$.
  $z_j \Leftarrow b(i,j) - \sum_{k\le l\le t} \beta(i,l)$.
  Let w_{min} be the minimum weight element in the set $S(i,k)$.
  Then $S(i,k) \leftarrow S(i,k) \setminus \{w_{min}\} \cup \{w(i,j)\}$.
  Compute $\delta(i,k)$ for the set $S(i,k)$.
  $\beta(i,k) \Leftarrow \delta(i,k)$
  while $\beta(i,k) \ge \sum_{F(i,k)\le l\le t} \beta_{start}(i,l)$ **do**
    $\beta(i,F(i,k)) \Leftarrow 0$ and change its state to dead.
    $S(i,k) \Leftarrow S(i,k) \cup S(i,F(i,k))$.
    Compute $\delta(i,k)$ for the set $S(i,k)$.
    $\beta(i,k) \Leftarrow \delta(i,k)$
  end while
  $\beta(i,F(i,k)) \leftarrow \beta(i,F(i,k))$
    $- \sum_{F(i,k)+1\le l\le k} (\beta_{end}(i,l) - \beta_{start}(i,l))$
end if

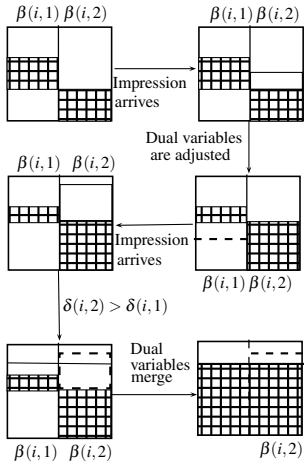

Figure 1: Smooth-Avg Algorithm. The left box represents the first interval and the right box represents the second. The height of the shaded area indicates the value of corresponding beta variable.

3.2 Analysis of the Smooth-Avg Algorithm

The goal of this section is to prove the following theorem:

THEOREM 3.1. *The approximation ratio of the smooth-avg algorithm is* $1/2$.

We first establish some important properties of beta variables. We call an iteration to be *merge-free* if no beta variables change their state to dead in the iteration. We note an important property of a merge free iteration.

FACT 1. *In a merge-free iteration of the algorithm, at most two beta variables change their value. Further more, if* $\mathbb{A}_i$ *is assigned the impression and the current interval is k, then*

$$\Delta(\beta(i,k)) = -\Delta(\beta(i,F(i,k)))$$

and remaining beta variables remain unchanged.

In the following lemma, we establish an important invariant maintained by the algorithm.

LEMMA 3.1. *For any* i,l, *when a beta variable* $\beta(i,l)$ *is active, then* $\delta(i,l) = \sum_{l\le p\le t}\beta(i,p)$.

PROOF. Given a beta variable $\beta(i,l)$, we first consider the l^{th} interval of the algorithm. In the l^{th} interval, $\beta(i,p) = 0$ for $p > l$ and the algorithm ensures that $\beta(i,l) = \delta(i,l)$. The set associated with $\beta(i,l)$ remains unchanged from the start of the $(l+1)^{th}$ iteration until $\beta(i,l)$ becomes dead. Hence it suffices to show that, while $\beta(i,l)$ is still active, $\sum_{l\le p\le t}\beta(i,p)$ does not change.

Consider the j^{th} iteration of the algorithm, and let the corresponding interval be $k(>l)$. There are two cases based on the type of the iteration. If that iteration is merge-free, then as $\beta(i,l)$ is still active, $F(i,k) \ge l$, and by using Fact 1, we get $\sum_{l\le p\le t}\beta_{end}(i,p) = \sum_{l\le p\le t}\beta_{start}(i,p)$.

Now we consider the case when the j^{th} iteration involves a merge operation. In this case, the algorithm ensures that

$$\Delta(\beta(i,F_{end}(i,k))) = -\left(\sum_{F_{end}(i,k)+1\le l\le k}\Delta(\beta(i,l))\right)$$

This proves the lemma. □

We now prove the correctness of the smooth-avg algorithm by establishing two properties (a) the dual at the end of the algorithm is feasible, and (b) in every iteration, (the change in the dual)/(the change in the primal)≤ 2. The feasibility of the dual solution follows from the following lemma:

LEMMA 3.2. *The sum* $\sum_{l\le p\le t}\beta(i,p)$ *never decreases during the course of the algorithm for any* i,l.

PROOF. We prove the claim by induction on iterations of the algorithm. We consider j^{th} iteration of the algorithm and let k be the corresponding interval. If $\mathbb{I}_j$ is not assigned to any advertiser, then the claim follows by the induction assumption as no beta variable is changed. Now we consider the case when it is assigned to $\mathbb{A}_i$. We show that $\sum_{l\le p\le t}\beta(i,p)$ increases for $F_{end}(i,k) \le l \le k$ and remains unchanged otherwise. The latter follows as the algorithm ensures,

$$\Delta(\beta(i,F_{end}(i,k))) = -\left(\sum_{F_{end}(i,k)+1\le l\le k}\Delta(\beta(i,l))\right)$$

Now we show that $\sum_{l\le p\le t}\beta(i,p)$ increases for $F_{end}(i,k) \le l \le k$. If the iteration is merge free and I_j is assigned to $\mathbb{A}_i$, then $\Delta(\beta(i,k)) > 0$, $\Delta(\beta(i,F_{start}(i,k))) = -\Delta(\beta(i,k))$, and remaining beta variables remain unchanged. Thus the claim holds in this case. Now we consider the case when the iteration is not merge free. Let $\{\beta(i,k_1),\beta(i,k_2),...,\beta(i,k_d)\}$ (with $k_1 < k_2 < ... < k_d$) be the set of beta variables that die in this iteration. We note that $F_{start}(i,k_l) = k_{l-1}$ for $2 \le l \le d$, $F_{start}(i,k) = k_d$ and $F_{start}(i,k_1) = F_{end}(i,k)$.

Applying the precondition for merge operation on the last merge operation of the iteration, we get that the average weight of impressions in the set $\left(\cup_{2\le p\le d}S_{start}(i,k_p)\right)\cup S(i,k)\cup\{w(i,j)\}/\{w_{min}\}$ is greater than $\delta_{start}(i,k_1)$ i.e. the average weight of impressions in $S_{start}(i,k_1)$. Thus $\beta_{end}(i,k)$, which is the average weight of impressions in $S_{end}(i,k) = \left(\cup_{1\le p\le d}S_{start}(i,k_p)\right)\cup S(i,k)\cup\{w(i,j)\}/\{w_{min}\}$, is greater than $\delta_{start}(i,k_1) = \sum_{k_1\le p\le k}\beta_{start}(i,p)$. Thus we get

$$\beta_{end}(i,k) > \sum_{k_1\le p\le k}\beta_{start}(i,p) = \sum_{F_{end}(i,k)+1\le p\le t}\beta_{start}(i,p)$$

as the other beta variables in the sum are 0. This completes the proof of the lemma. □

We are now ready to prove Theorem 3.1:

PROOF. (Theorem 3.1) We compute the change in the primal and dual in the j^{th} iteration of the algorithm. If $\mathbb{I}_j$ remains unassigned, then the primal and the dual solutions do not change. When it is assigned to $\mathbb{A}_i$, the change in the primal is $w(i,j) - w_{min}$ and it suffices to show that the change in the dual is at most $2(w(i,j) - w_{min})$ to establish the theorem.

We first consider the case when the j^{th} iteration is merge-free. In this case, by Lemma 3.1, we have $\Delta(\beta(i,k)) = -\Delta(\beta(i,F(i,k))) = \frac{w(i,j)-w_{min}}{N(i,k)-N(i,F(i,k))}$ and other dual variables remains unchanged. Thus the total change in the dual solution is

$$\begin{aligned} &= \Delta(\beta(i,F(i,k)))N(i,F(i,k)) + \Delta(\beta(i,k))N(i,k) + z_j \\ &= w(i,j) - w_{min} + z_j \end{aligned}$$

Moreover $z_j = w(i,j) - \beta(i,k) \le w(i,j) - w_{min}$. Thus the change in the dual is at most $2(w(i,j) - w_{min})$.

Now we consider the case when the j^{th} iteration is not merge free. Let $S_{start}(i,k_1),S_{start}(i,k_2),...,S_{start}(i,k_d)$ be the sets that merge with $S_{start}(i,k)$ in the current iteration, such that $k_1 < k_2 < ... < k_d$. As a result, the beta variables $\beta(i,k_1),\beta(i,k_2),...,\beta(i,k_d)$ change their state to dead. For simplicity, we denote $F_{end}(i,k)$ at the end of the iteration by k_0.

The value of $\delta_{end}(i,k)$ is the average weight of impressions in $S_{end}(i,k) = \left(\cup_{1\le p\le d}S_{start}(i,k_p)\right)\cup S_{start}(i,k)\cup\{w(i,j)\}/\{w_{min}\}$. Now we compute the average weight of impressions in $S_{end}(i,k)$. The average weight of impressions in $S_{start}(i,k_l)$ is $\delta_{start}(i,k_l)$ and $|S_{start}(i,k_l)| = N(i,k_l) - N(i,k_{l-1})$ for $1 \le l \le d$. By Lemma 3.1, for an active beta variable $\beta(i,l)$, $\delta_{start}(i,l) = \sum_{l\le p\le t}\beta_{start}(i,p)$. As $\beta(i,p) = 0$ for $p > k$, we get

$$\begin{aligned} &\beta_{end}(i,k)\left(N(i,k) - N(i,k_0)\right) \\ &= \delta_{start}(i,k)\left(N(i,k) - N(i,k_d)\right) \\ &+ \sum_{1\le l\le d}\delta_{start}(i,k_d)\left(N(i,k_l) - N(i,k_{l-1})\right) + w(i,j) - w_{min} \\ &= \beta_{start}(i,k)\left(N(i,k) - N(i,k_0)\right) \\ &+ \sum_{1\le l\le d}\beta_{start}(i,k_l)\left(N(i,k_l) - N(i,k_0)\right) + w(i,j) - w_{min} \qquad (1) \end{aligned}$$

Thus we get

$$\Delta(\beta(i,k)) = \frac{\sum_{1<l<d}\beta_{start}(i,k_l)(N(i,k_l)-N(i,k_0)) + w(i,j) - w_{min}}{N(i,k)-N(i,k_0)}$$

The variable $\beta(i,k_0)$ is active at the end of iteration. Using Lemma 3.1, we get

$$\delta_{start}(i,k_0) = \delta_{end}(i,k_0) = \sum_{k_0\le l\le t}\beta_{end}(i,l) = \beta_{end}(i,k_0) + \beta_{end}(i,k)$$

as other beta variables are 0. Thus we get,

$$\begin{aligned}\Delta(\beta(i,k_0)) &= -\sum_{k_0<l\le t}\Delta(\beta(i,l)) \\ &= -\Delta(\beta(i,k)) - \sum_{1\le p\le d}\beta_{start}(i,k_p) \qquad (2)\end{aligned}$$

The change in the dual is equal to

$$= \Delta(\beta(i,k))N(i,k) + \sum_{0\le l\le d}\Delta(\beta(i,k_l))N(i,k_l) + z_j$$

Using Equations 1 and 2, we get

$$= \sum_{1\le l\le d}\beta_{start}(i,k_l)\left(N(i,k_l)-N(i,k_0)\right) - \sum_{1\le l\le d}\beta_{start}(i,k_l)\left(N(i,k_l)-N(i,k_0)\right) + w(i,j) - w_{min} + z_j$$

$$= w(i,j) - w_{min} + z_j \le 2(w(i,j)-w_{min})$$

This completes the proof of the theorem. □

3.3 Smooth-Exp Algorithm

In this section, we illustrate our result for the smooth-exp algorithm. The algorithm uses the same basic framework as the smooth-avg algorithm; we illustrate the differences between the two algorithms and its correctness proof in this section.

We first establish an important property of weighted exponential average functions in the following lemma, that will be crucial to the smooth-exp algorithm.

LEMMA 3.3. *Given two sequences S_1 and S_2, with weighted exponential averages δ_1 and δ_2 respectively, the weighted exponential average of their merged sequence $S_1 \cup S_2$ is at most $\frac{\delta_1|S_1|+\delta_2|S_2|}{|S_1|+|S_2|}$*

PROOF. Let $n_1 = |S_1|, n_2 = |S_2|$, and $n = n_1 + n_2$. Let $a_1, a_2, ..., a_{n_1}$ and $b_1, b_2, ..., b_{n_2}$, be elements in S_1, S_2 in an ascending order. Let $S = c_1, c_2, c_3, ..., c_n$ be their merged sequence. We shall construct an ordered sequence $S' = d_1, d_2, d_3, ..., d_n$ such that its weighted exponential average is no more than $\frac{\delta_1 n_1 + \delta_2 n_2}{n_1+n_2}$ and S' ensures following property:

PROPERTY 1. *For every $1 \le i \le n$, $\sum_{1\le j\le i} c_j \le \sum_{1\le j\le i} d_j$, and $\sum_{1\le j\le n} c_j = \sum_{1\le j\le n} d_j$.*

Let $\alpha(i,n_1) = \frac{(1+1/n_1)^{n_1-i}}{n_1((1+1/n_1)^{n_1}-1)}$, i.e the i^{th} term in exponential weights for a sequence of size n_1. Similarly, we define $\alpha(i,n_2)$ and $\alpha(i,n)$. The weighted exponential averages of S and S' are $\sum_{1\le i\le n}\alpha(i,n)c_i$ and $\sum_{1\le i\le n}\alpha(i,n)d_i$ respectively. As S and S' are in an ascending order, using Property 1, we get,

$$\sum_{1\le i\le n}\alpha(i,n)c_i \le \sum_{1\le i\le n}\alpha(i,n)d_i$$

which proves the lemma. So it remains to establish the existence of sequence S'.

Let $\gamma = n_1/n$. For any sequence $Q = q_1, q_2, q_3, ..., q_{|Q|}$ and $0 \le r \le |Q|$, we define $\texttt{Prefix}(Q,r) = \sum_{1\le j\le \lfloor r\rfloor} q_j + (r - \lfloor r\rfloor) * q_{\lceil r\rceil}$. We construct S' as follows: for every $i, 1 \le i \le n$

$$\begin{aligned} d_i &= \gamma R(1,i) + (1-\gamma)R(2,i) \\ \text{where} & \\ R(1,i) &= \texttt{Prefix}(S_1, i\gamma) - \texttt{Prefix}(S_1,(i-1)\gamma) \quad \text{and} \\ R(2,i) &= \texttt{Prefix}(S_2, i(1-\gamma)) - \texttt{Prefix}(S_2,(i-1)(1-\gamma)) \end{aligned}$$

Now we check properties of S': (a) as S_1 and S_2 are sorted, S' is also sorted in an ascending order, and (b) as the sum of first i elements in S' is the sum of first $i \times n_1/n$ and $i \times n_2/n$ elements from S_1 and S_2 respectively, it easily follows that S' satisfies Property 1.

Now it remains to establish the bound on the exponential weighted average of S'. We first note that δ_1, i.e. the exponential average of S_1, can be written as

$$\sum_{1\le i\le n_1} a_i\alpha(i,n_1) = a_1\left(\sum_{1\le i\le n_1}\alpha(i,n_1)\right) + \sum_{2\le i\le n_1}\left((a_i - a_{i-1})\left(\sum_{i\le j\le n_1}\alpha(j,n_1)\right)\right)$$

It suffices to show that the weighted exponential average of sequence $R_1 = R(1,1), R(1,2), ..., R(1,n)$ is upper bounded by δ_1. Then, by symmetry, the weighted exponential average of sequence $R_2 = R(2,1), R(2,2), ..., R(2,n)$ is upper bounded by δ_2, and by definition of S', we get the required upper bound on its weighed exponential average.

The weighted exponential average of R_1 can be written as

$$a_1\textstyle\sum_{1\le i\le n}\alpha(1,i) + \sum_{2\le i\le n_1}(a_i - a_{i-1})\left(\sum_{\lceil\frac{i-1}{\gamma}\rceil+1\le j\le n}\alpha(j,n)\right) + \sum_{2\le i\le n_1}(a_i-a_{i-1})\left(\lceil\tfrac{i-1}{\gamma}\rceil - \tfrac{i-1}{\gamma}\right)\alpha\left(\lceil\tfrac{i-1}{\gamma}\rceil, n\right)$$

Let $Z_1(i)$ and $Z(i)$ be multipliers of term $(a_i - a_{i-1})$ for sequences S_1 and R_1 respectively. As $\sum_{1\le i\le n}\alpha(1,i) = \sum_{1\le i\le n_1}\alpha(i,n_1) = 1$, and $\forall i \ge 2, (a_i - a_{i-1}) \ge 0$, it suffice to show that $Z_1(i) \ge Z(i)$ for each $2 \le i \le n_1$. Then we have

$$\begin{aligned} Z_1(i) &= \tfrac{(1+1/n_1)^{(n_1-i+1)}-1}{(1+1/n_1)^{n_1}-1} \quad \text{and} \\ Z(i) &= \tfrac{(1+1/n)^{(n_1-i+1)/\gamma}-1}{(1+1/n)^n-1} \end{aligned}$$

We note an important property in the following lemma, it can be easily proven by differentiation.

LEMMA 3.4. *For any fixed $c, 0 \le c \le 1$, the function $\frac{(1+1/n)^{cn}-1}{(1+1/n)^n-1}$ is non-increasing in n.*

By choosing $c = (n_1 - i + 1)/n_1$, we get $Z_1(i) \ge Z(i)$. This completes the proof. □

Algorithm: The algorithm is similar to the smooth-avg algorithm with one exception: the difference occurs in merge operation due to different behavior shown by exponential average function compared to "average" function. For any two sequences S_1 and S_2 with exponential average δ_1 and δ_2, the merged sequence can have exponential average less than $\frac{\delta_1|S_1|+\delta_2|S_2|}{|S_1|+|S_2|}$ (Lemma 3.3). This creates issue in the feasibility of dual if we follow the same algorithm as smooth-avg. We handle the merge operation as follows: in the iteration where $\beta(i,k)$ merges with $\beta(i,F(i,k))$, $\mathbb{I}_j$ has been assigned to $\mathbb{A}_i$ before the merge operation. Let $S(i,k) = S_{start}(i,k) \cup w(i,j) \backslash w_{min}$, and $\delta(i,k)$ be its weighted exponential average. Let $\gamma = \max\{\beta_{start}(i,k), \delta(i,k)\}$. Then we assign the following value to $\beta(i,k)$ as result of the merge operation

$$\frac{\gamma|S_{start}(i,k)| + (\beta_{start}(i,k) + \beta_{start}(i,F(i,k)))|S_{start}(i,F(i,k))|}{|S_{start}(i,F(i,k))| + |S_{start}(i,k)|}$$

Similar to the smooth-avg algorithm, among the variables that are active at the end of the iteration, the value changes for only two of them: $\beta(i,k)$ and $\beta(i,F_{end}(i,k))$. Furthermore, $\beta(i,F_{end}(i,k))$ is changed such that, we have

$$\Delta(\beta(i,F_{end}(i,k))) = -\sum_{F_{end}(i,k)+1\le l\le t}\Delta(\beta(i,l))$$

Thus they continue to hold the marginal over the next active beta variable. Now we note important properties ensured by the algorithm.

FACT 2. *In the k^{th} interval, (a) before any merge operation involving $\beta(i,k)$, the algorithm ensures $\beta(i,k) = \delta(i,k)$, (b) after a merge operation involving $\beta(i,k) \geq \delta(i,k)$, and (c) $\beta(i,k)$ never decreases.*

FACT 3. *For any i,l, when a variable $\beta(i,l)$ is active, then $\sum_{l \leq p \leq t} \beta(i,p) \geq \delta(i,l)$.*

FACT 4. *During the course of the algorithm, $\sum_{l \leq p \leq t} \beta(i,p)$ does not decrease for any i,l.*

It can be checked that these properties continue to hold for a merge-free iteration. Now we consider an iteration in which $\beta(i,k)$ merges with $\beta(i,F(i,k))$; we have: (a) the current interval is k^{th} interval and $\beta(i,>k) = 0$, (b) $\delta_{start}(i,k) \leq \beta_{start}(i,k)$, $\beta(i,>k) = 0$, and (c) $\delta_{start}(i,F(i,k)) \leq \beta_{start}(i,k) + \beta_{start}(i,F(i,k))$. By Lemma 3.3, the weighed exponential average of $S_{end}(i,k)$ after merging is upper bounded by

$$\frac{\gamma|S_{start}(i,k)| + \delta_{start}(i,F(i,k))|S_{start}(i,F(i,k))|}{|S_{start}(i,F(i,k))| + |S_{start}(i,k)|} \leq$$
$$\frac{\gamma|S_{start}(i,k)| + (\beta_{start}(i,k) + \beta_{start}(i,F(i,k)))|S_{start}(i,F(i,k))|}{|S_{start}(i,F(i,k))| + |S_{start}(i,k)|}$$

This ensures that $\delta_{end}(i,k) \leq \beta_{end}(i,k)$. Furthermore, it can be checked that the merge operation does not increase the value of the dual solution and the dual continues to remain feasible. When more than one merge operation take place in the same iteration for an advertiser, we treat it as a sequence of merge operations such that in step, one beta variable merges with $\beta(i,k)$, and each step is treated in the same way as discussed above. We now establish the performance ratio of the algorithm.

THEOREM 3.2. *If $(N(i,k) - N(i,k-1))$ is large for every i,k, then the performance ratio of the smooth-exp algorithm is $(1 - 1/e)$.*

PROOF. We compare the change in the primal and the dual solution in the j^{th} iteration of the algorithm, let k be the corresponding interval. If $\mathbb{I}_j$ remains unassigned, then the primal and the dual solutions do not change. If $\mathbb{I}_j$ is assigned to $\mathbb{A}_i$, then the change in the primal is $w(i,j) - w_{min}$. If the j^{th} iteration is merge free, then the change in the dual is $\Delta(\beta(i,k))N(i,k) - \Delta(\beta(i,k))N(i,F(i,k)) + z_j$. Furthermore, we have

$$\Delta(\beta(i,k)) \leq \delta_{end}(i,k) - \beta_{start}(i,k) \leq \delta_{end}(i,k) - \delta_{start}(i,k)$$
$$\leq \frac{w(i,j)}{(N(i,k) - N(i,F(i,k)))(e-1)} + \frac{\delta_{start}(i,k)}{N(i,k) - N(i,F(i,k))}$$
$$- \frac{e \cdot w_{min}}{(e-1)(N(i,k) - N(i,F(i,k)))}$$

Thus the change in the dual is

$$= \Delta(\beta(i,k))N(i,k) - \Delta(\beta(i,k))N(i,F(i,k)) + z_j$$
$$\leq \frac{w(i,j)}{e-1} + \delta_{start}(i,k) - \frac{e \cdot w_{min}}{e-1} + w(i,j) - \sum_{k \leq l \leq t} \beta_{start}(i,l)$$

Using $\beta_{start}(i,p) = 0$ for $p > k$ and $\beta_{start}(i,k) \geq \delta_{start}(i,k)$, the change in the dual is bounded by

$$\leq \frac{w(i,j)}{e-1} + \beta_{start}(i,k) - \frac{e \cdot w_{min}}{e-1} + w(i,j) - \beta_{start}(i,k)$$
$$= \frac{(w(i,j) - w_{min})e}{e-1}$$

Thus the change in the dual is at most $e/(e-1)$ times the change in the primal. When the iteration involves a merge operation, we consider two stages involved in the iteration, (a) in the first stage, we add the impression to $S(i,k)$, $\beta(i,k)$ increases and $\beta(i,F(i,k))$ decreases and the change in this step is bounded by $(e/(e-1))(w(i,j) - w_{min})$, (b) as the value of $\beta(i,F(i,k))$ is now negative, $\beta(i,k)$ and $\beta(i,F(i,k))$ merge (potentially followed by a sequence of more merge operations), and using Lemma 3.3, there is no increase in the dual objective in this step. This completes the proof. □

Recall that, the algorithm in [7] requires each advertiser to have a large capacity to get a $(1 - 1/e)$ approximation. We now show that, even a stronger version of this assumption does not suffice to get a $(1 - 1/e)$-approximation in the smooth delivery setting, and the assumption made in Theorem 3.2 is indeed required. We defer its proof to the full version.

LEMMA 3.5. *For the Display Ads Allocation problem, no deterministic algorithm can do better than $1/2$, even when $N(i,k)$ is large for every i and large k.*

4. EXPERIMENTAL EVALUATION

We study the performance of our algorithm on display ads data for a set of 10 anonymous publishers working with Google (DoubleClick) display advertising system. The data set is collected over a period of one week for each publisher. The number of impressions in a data set range from 200 thousands to 3 millions.

Weight of an Advertiser-Impression Pair: We associate the CTR for an impression-advertiser pair with the advertiser's welfare from the assignment. Thus the objective of a display ad system is to assign impressions subject to the capacity constraints so that the (expected) number of clicks is maximized.

The Delivery Model and Penalty: We assume a linear delivery model for every advertiser, i.e. the capacity of $\mathbb{A}_i$ after arrival of j impressions is $N(i,j) = \frac{j}{m}N(i)$. We measure under-delivery and over-delivery of the assignment at 200 equally spaced milestones during the course of the algorithm.

As there is a penalty for under-delivery of impressions to a publisher, the algorithm should keep the under-delivery as small as possible. Even though there is no penalty for assigning impressions more than the capacity of the advertiser in the free-disposal model, *the impressions that are not assigned can be used for other properties in Google, hence it is imperative to keep even the over-delivery of impressions as small as possible.*

Algorithms Compared: We analyze the performance of smooth-avg, smooth-exp and smooth-greedy against the two reference algorithms:

Pd-Avg: We compare against a baseline online matching algorithm with free disposal [7] considering the overall capacity constraint but ignoring the smooth delivery constraints. In particular, we consider the pd-avg online matching algorithm developed by [7]. The algorithm is as follows: For advertiser $\mathbb{A}_i$, let $S(i)$ be the set of *top* $N(i)$ impressions assigned to $\mathbb{A}_i$; if there are less than $N(i)$ impressions assigned to $\mathbb{A}_i$, then we add elements with weight 0 so that $|S(i)| = N(i)$. The algorithm associates a dual variables $\beta(i)$ to $\mathbb{A}_i$, where the value of $\beta(i)$ is the average weight of impressions in $S(i)$. The algorithm assigns the impression to the advertiser which maximizes $w(i,j) - \beta(i)$. We expect this algorithm to achieve a better total welfare, but more overdelivery/underdelivery as it does not obey the tighter smooth delivery constraints.

A Heuristic: It is a natural baseline heuristic used in practice that adopts the pd-avg algorithm from [7] to ensure smooth delivery. It

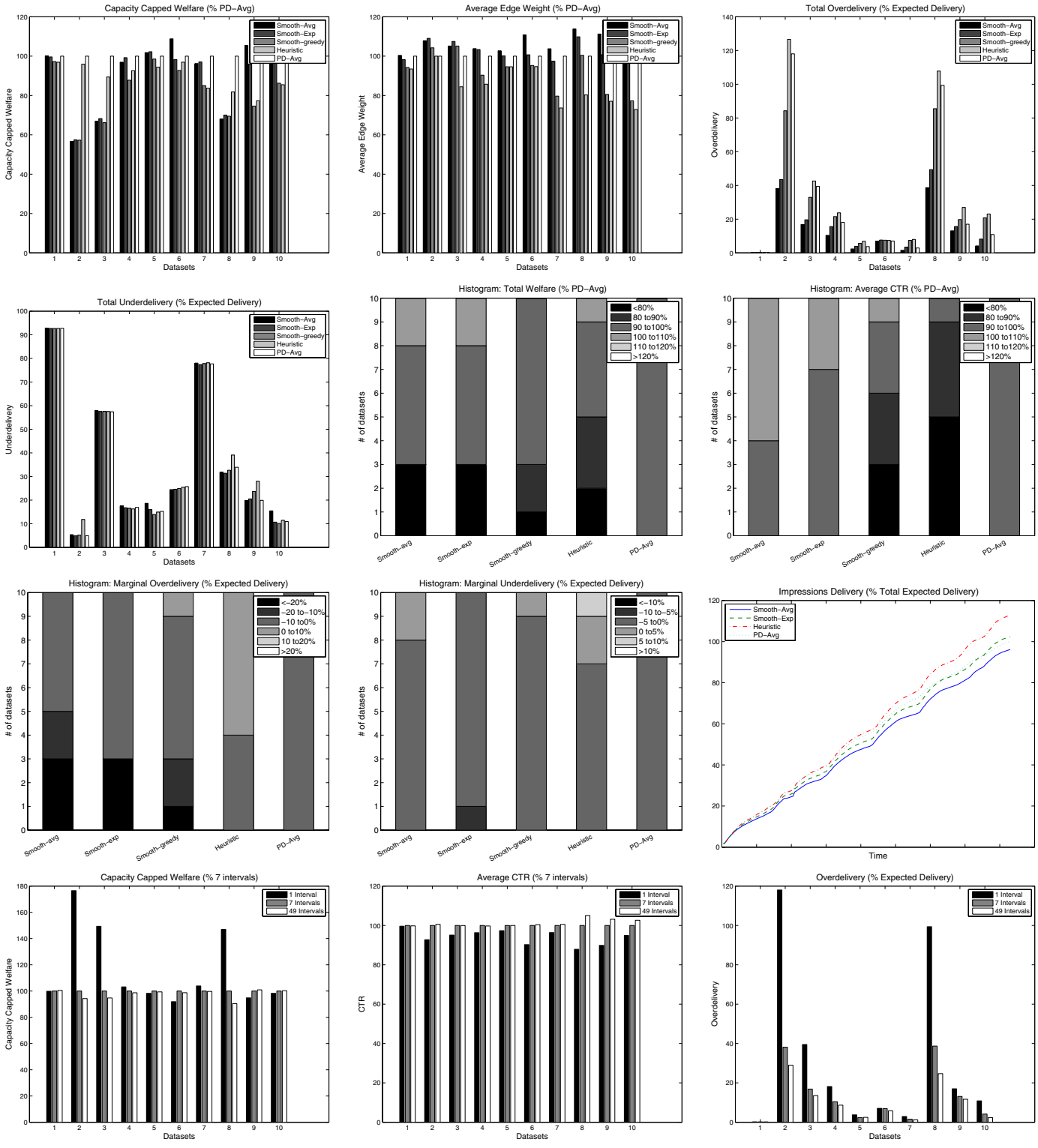

Figure 2: First four figures (all in the first row and the first in the second row) plot performance metrics of various smooth delivery algorithms for each data-set, the next four figures are histograms of data-sets based on various performance metrics for these algorithms, and the sixth figure shows time characteristics of impressions delivery for a sample data-set. The figures in the last row plot performance metrics for different number of intervals in each data-set for smooth-avg.

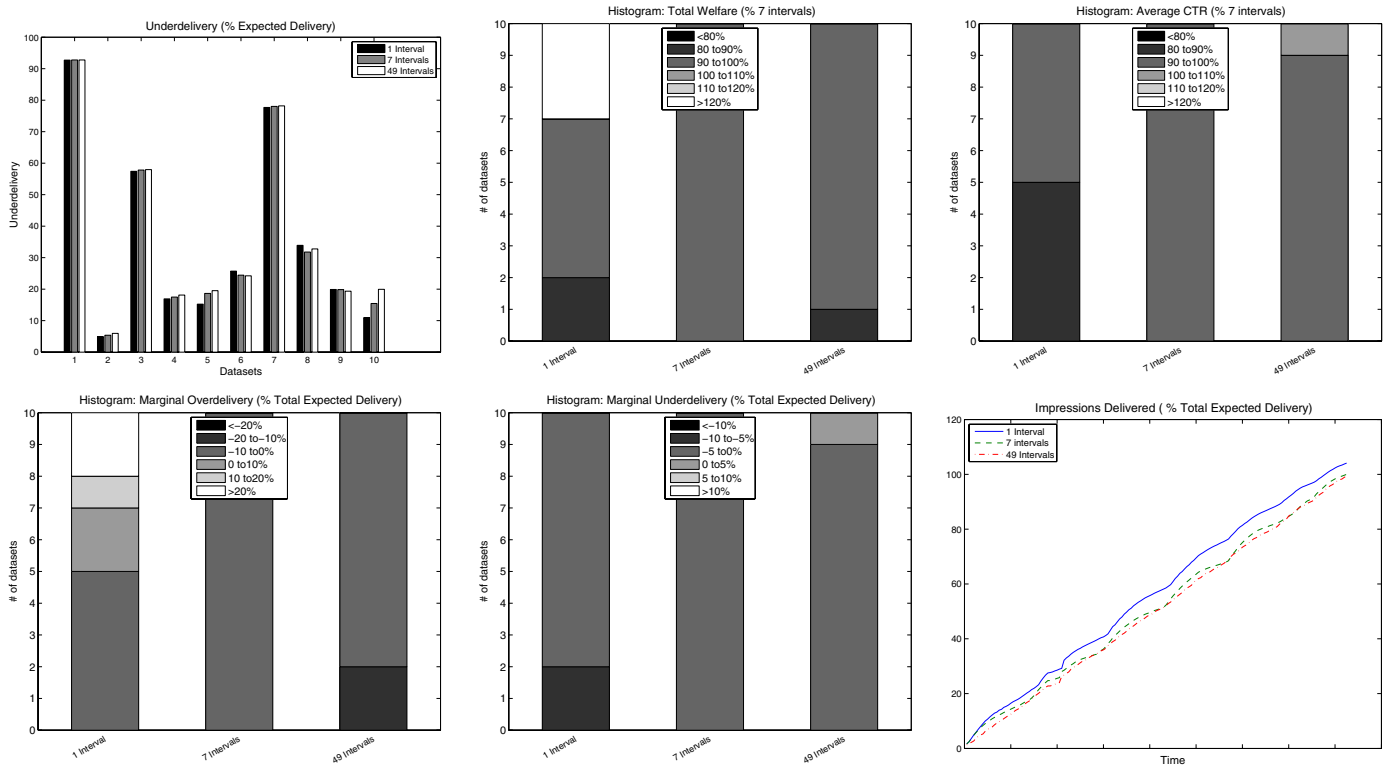

Figure 3: The first figure plots the value of under delivery for different number of intervals in each data-set for smooth avg, the next four figures are histograms of data-sets based on various performance metrics, and the last figure in the last row show time characteristics of impressions delivery for a sample data-set.

is very similar to the algorithm described above, the only difference is that in this heuristic, before arrival of $\mathbb{I}_j$, the capacity of $\mathbb{A}_i$ is considered to be $\frac{j}{m}N(i)$ (instead of $N(i)$). Therefore, in the algorithm above, $S(i)$ for heuristic algorithm is a set of $\frac{j}{m}N(i)$ edge weights corresponding to the *top* $\frac{j}{m}N(i)$ impressions assigned to $\mathbb{A}_i$. We expect this algorithm to have better delivery guarantee, and worse total weight of the assignment.

We use 7 intervals in the implementation of smooth algorithms, and we measure delivery guarantees at 200 equally spaced milestones.

Results and Metrics: The average statistics over 10 data sets can be found in the Table 1. It uses the following set of metrics for the comparison.

(1) *Total Welfare* is the total weight of the assignment regardless of the delivery constraint, relative to the pd-avg algorithm.
(2) *Capacity Capped Welfare* is the total weight of the assignment taking the smooth capacity constraint into consideration, relative to the pd-avg algorithm. This is the most relevant metric.
(3) *Average edge weight* in the assignment, relative to the pd-avg algorithm.
(4) *Under-delivery/over-delivery* at the end of the algorithm as a fraction of the expected delivery.
(5) *Accumulated under-delivery/over-delivery* summed over 200 milestones, as a fraction of the accumulated expected delivery.

These statistics for individual data-sets can be found in the Figure 2. We also plot histograms of data-set based on the performance metrics (Figure 2), they show the number of data-sets achieving a certain performance for the given metric relative to the pd-avg algorithm. The under-delivery/over-delivery histograms show the marginal under-delivery/over-delivery compared to the pd-avg algorithm, measured as a fraction of the total expected delivery. All other plots measure the performance metric as a fraction of the pd-avg algorithm.

Salient Observations.

Overall, smooth-exp and smooth-avg algorithms ensure significantly less over-delivery (by about 15% of the expected delivery) than the pd-avg algorithm and the heuristic, and have a matching under-delivery guarantee. Although they (smooth-exp and smooth-avg algorithms) achieve a marginally lower overall weight of the assignment, they ensure significantly larger *average edge weight in the assignment*.

Delivery Guarantees of Smooth Algrorithms: It is interesting to note that smooth-avg and smooth-exp have *less over-delivery* and *more average edge weight* in the assignment *for every data* set compared to the pd-avg and the heuristic algorithm. *This is contrary to expectations since the heuristic is specifically designed to ensure smooth delivery, at a cost of less overall weight of the assignment.* While smooth-exp has under-delivery less than the two reference algorithm on all datasets, smooth-avg performs better in 8 out of 10 datasets. Furthermore, smooth-avg has less over-delivery than smooth-exp for every data set.

We plot values of various performance metrics measured at 200 observation points for a sample data-set (Figure 2). The performance is measured relative to the pd-avg algorithm. In each plot, we can observe 7 concave regions, they correspond to the seven intervals. At the beginning of the interval, the *current* beta variable is

Table 1: Performance of various smooth algorithms averaged over 10 Data Sets

Metric	Smooth-Exp	Smooth-Avg	Smooth-Greedy	Heuristic	Pd-Avg
Total Welfare	94.22	95.69	95.69	93.35	100
Capacity Capped Welfare	90.28	89.25	81.51	89.45	100
Average Edge Weight	106.48	102.64	92.17	85.68	100
Over-delivery	11.51	15.13	28.14	36.54	26.30
Accumulated Over-delivery	13.24	16.72	28.58	37.35	31.68
Total Under-delivery	26.16	24.44	24.17	26.20	24.91
Accumulated Under-delivery	36.15	35.22	35.50	37.52	35.53

Table 2: Effect of Number of Intervals

Metric	1 Interval	7 Intervals	49 Intervals
Total Welfare	109.41	100	98.66
Average Edge Weight	94.04	100	101.19
Over-delivery	26.29	11.50	9.85
Accumulated Over-delivery	31.68	13.24	9.96
Total Under-delivery	24.91	26.15	26.84
Accumulated Over-delivery	35.52	36.16	36.87

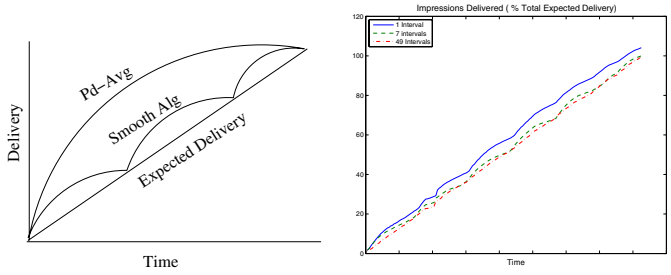

Figure 4: The figure on the left illustrates the reason for the less over-delivery of smooth-algorithms. The figure on the right is the delivery curve for a sample data-set over different values of intervals.

zero and hence impressions are assigned as a faster rate. As the *current* beta variable increases, the rate of assignment of impressions reduces.

Smooth-Avg vs Smooth-Exp Algorithm: Even though the smooth-exp algorithm has a better approximation guarantee, its implementation requires storing all impressions assigned to an advertiser so far and the update rules are computationally expensive as it involves computing the weighted exponential average of a set. In comparison, the smooth-avg algorithm only needs to maintain the value of beta variables and its update rules are simple. Furthermore, the performance of the smooth-avg algorithm is no worse than the smooth-exp algorithm on data. This makes the smooth-avg algorithm a better candidate for practical applications.

Number of Intervals and Smoothness: We analyze the effect of increasing the number of intervals. Towards this, we use three different values of intervals: 1, 7 and 49. We use smooth-avg algorithm and measure over-delivery and under-delivery at 200 uniformly spaced milestones. The average statistics can be found in the Table 2. These statistics for individual data-sets are given in the Figures 2 and 3. We also plot histograms of data-sets based on these performance parameters, and the values of various performance metrics measured at 200 observation points for a sample data-set (Figure 3). The performance is measured relative to the smooth-avg algorithm with 7 intervals.

We observe that the overall smoothness of the delivery increases as we increase the number of intervals. Though the average edge weight and the delivery guarantees improve with the increase in the number of intervals from 1 to 7, the corresponding gains by increasing the number of intervals to 49 are marginal and it leads to the drop the total weight of the assignment. We note that the over-delivery for every data-set reduces as we increase the number of intervals from 1 to 7 as well as from 7 to 49.

Reasons for Better Over-Delivery Guarantee of Smooth Algorithms.

In the pd-avg algorithm, the value of a beta variables increase from 0 to to its final value by the end of the algorithm and an impression is assigned only when its weight is more than the beta variable's value. If we assume an IID model of arrival, then as the value of a beta variable increases, the rate of delivery decreases as less impressions have weight more than the value of the beta variable. Thus the effective delivery curve is *concave*. With multiple intervals, the beta value is *reset to* 0 multiple times, thus the eventual delivery curve is made up of multiple small concave regions and its sum of distance form a linear expected delivery is less. (See Figure 1 for an illustration.)

5. REFERENCES

[1] S. Agrawal, Z. Wang, and Y. Ye. A dynamic near-optimal algorithm for online linear programming. Working paper posted at http://www.stanford.edu/ yyye/.

[2] S. Alaei, R. Kumar, A. Malekian, and E. Vee. Balanced allocation with succinct representation. In *KDD*, pages 523–532, 2010.

[3] N. Buchbinder, K. Jain, and J. Naor. Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue. In *ESA*. Springer, 2007.

[4] Y. Chen, P. Berkhin, B. Anderson, and N. R. Devanur. Real-time bidding algorithms for performance-based display ad allocation. In *KDD*, pages 1307–1315, 2011.

[5] N. Devanur and T. Hayes. The adwords problem: Online keyword matching with budgeted bidders under random permutations. In *ACM EC*, 2009.

[6] J. Feldman, M. Henzinger, N. Korula, V. Mirrokni, and C. Stein. Online stochastic packing applied to display ad allocation. In *ESA*, 2010.

[7] J. Feldman, N. Korula, V. Mirrokni, S. Muthukrishnan, and M. Pal. Online ad assignment with free disposal. In *WINE*, 2009.

[8] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized online matching. In *FOCS*, 2005.

[9] V. Mirrokni, S. O. Gharan, and M. ZadiMoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation problems. In *SODA*, 2012.

[10] B. Tan and R. Srikant. Online advertisement, optimization and stochastic networks. *CoRR*, abs/1009.0870, 2010.

[11] E. Vee, S. Vassilvitskii, and J. Shanmugasundaram. Optimal online assignment with forecasts. In *ACM EC*, 2010.