OPERATIONS RESEARCH
Vol. 59, No. 1, January–February 2011, pp. 1–16
ISSN 0030-364X | EISSN 1526-5463 | 11 | 5901 | 0001

DOI 10.1287/opre.1100.0852
© 2011 INFORMS

INFORMS holds copyright to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

OR PRACTICE

Scheduling of Dynamic In-Game Advertising

John Turner
The Paul Merage School of Business, University of California, Irvine, Irvine, California 92697, john.turner@uci.edu

Alan Scheller-Wolf, Sridhar Tayur
Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213
{awolf@andrew.cmu.edu, stayur@andrew.cmu.edu}

Dynamic in-game advertising is a new form of advertising in which ads are served to video game consoles in real time over the Internet. We present a model for the in-game ad-scheduling problem faced by Massive Inc., a wholly owned subsidiary of Microsoft, and a leading global network provider of in-game ad space. Our model has two components: (1) a linear program (solved periodically) establishes target service rates, and (2) a real-time packing heuristic (run whenever a player enters a new level) tracks these service rates. We benchmark our model against Massive's legacy algorithm: When tested on historical data, we observe (1) an 80%–87% reduction in make-good costs (depending on forecast accuracy), and (2) a shift in the age distribution of served ad space, leaving more premium inventory open for future sales. As a result of our work, Massive has increased the number of unique individuals that see each campaign by, on average, 26% per week and achieved 33% smoother campaign delivery as measured by standard deviation of hourly impressions served.

Subject classifications: dynamic in-game advertising; video game advertising; display advertising; revenue management; linear programming; goal programming.
Area of review: OR Practice.
History: Received July 2008; revisions received March 2009, July 2009, December 2009; accepted February 2010. Published online in *Articles in Advance* February 8, 2011.

1. Introduction

Video games have incorporated static ads for decades: for example, an old racing game may have a billboard that always displays the same Sunoco ad. Recently, however, technology and business relationships have matured to allow Internet-enabled consoles (e.g., Xbox, PCs) to *dynamically* serve ads over time, creating an entirely new ad market: Dynamic in-game ad technology allows in-game billboards to display different ads to different players based on their demographic, the time of day, the day of week, and possibly other parameters. The in-game advertising industry is growing quickly; at present, revenues are projected to reach $800 million by 2012 (Cai 2007).

At the heart of an in-game ad system is the *ad server*: When a player enters a new level of a game, his console connects to the ad server via the Internet and requests new ad graphics for billboards, stadium walls, and other locations where ads are shown in the level. The ad server decides which ads to serve to this player, functioning like a Web server delivering banner ads. But, unlike on the Web, where a selected ad is almost certainly seen, it is common for only a fraction of selected in-game ads to be seen. Billable ad time is thus recorded only when, as part of game play, the player navigates through the level and passes locations where ads are displayed. For this reason, and also because of additional constraints (such as *saturation*, *competition*, and *context*, discussed below), scheduling in-game ads is significantly more complicated than scheduling banner ads.

The ad server is operated by a network provider, an intermediary between game publishers and advertising agencies. We focus on the scheduling problem faced by Massive Inc., a wholly owned subsidiary of Microsoft and a leading global network provider. Game publishers allow Massive to sell and serve ads in their games, and consequently receive a portion of the generated revenues; ad agencies buy ad campaigns directly from Massive.

The operational problem network providers like Massive face is how to schedule and serve ads to players over time so as to make the best use of their inventory of ad space. Campaigns purchased by an ad agency specify a target number of *impressions* (ads seen by gamers), a rough schedule for serving these impressions over time, and also a desired mix (e.g., 60% in sports games, 40% in the rest of the games). A campaign's delivery may also be restricted to certain geographic areas and/or times of the day. In addition, the network provider must also manage (1) *saturation*: it is undesirable for a single player to simultaneously see many copies of the same campaign; (2) *competition*: campaigns of two competing brands—e.g., Coke and

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Pepsi—should not be served to the same player; and (3) *context*: ads should not seem out of place within the game—e.g., Coke ads belong on virtual soda machines, Tide ads do not.

The size, scope, and complexity of Massive's problem are such that even if there were no system uncertainty, optimization of their ad server would be difficult. But, of course, uncertainty is present—there are three primary sources: (1) the acquisition of new games, (2) the sale of new campaigns, and (3) error in inventory forecasts of ad space. This last factor, uncertainty in the amount of ad space, arises because the number of players, the types (demographics) of players, and the ad space that the players actually see during game play are not known when the scheduling problem needs to be solved. Thus, campaigns sometimes fall short of their impression goals or deviate from the desired pattern of delivery; in that case, the network provider offers the advertiser a *make-good*: the campaign is extended or the advertiser is offered a refund or credit for future use.

We present the first planning/scheduling model and algorithm for dynamic in-game advertising. Our model has two components: (1) a linear program called the *Weekly Plan* is solved periodically to establish target service rates, and (2) a packing heuristic called the *Real-Time Algorithm* is run whenever a player enters a new level, to serve impressions in accordance with these service rates. Benchmarking our model against Massive's legacy algorithm using historical data, we observe (1) an 80%–87% reduction in make-good costs (depending on forecast accuracy), and (2) a shift in the age distribution of served ad space, leaving more premium inventory open for future sales. Massive has begun a staged implementation of our model, and to date has benefitted from a 26% average increase in the number of unique individuals that see each campaign each week, and 33% smoother campaign delivery, as measured by standard deviation of hourly impressions served.

We proceed as follows: In §2, we review the literature on media planning and scheduling, and describe existing models for broadcast television and webpage banner ads. We introduce the problem in §3, the Weekly Plan LP in §4, and the Real-Time Algorithm in §5. We benchmark our algorithm against Massive's legacy algorithm in §6, and comment on our implementation learnings in §7. We conclude with comments and list future work in §8.

2. Literature Review

To the best of our knowledge, ours is the first academic treatment of the scheduling of in-game advertisements. The process of dynamic in-game advertising is well documented (Chambers 2005, Svahn 2005); however, operational problems, such as the scheduling problem we examine, have not been studied. In the traditional media planning literature, the optimization problem is usually that of a single advertiser and not of a network provider serving many advertisers. In these models, the advertiser chooses among advertising vehicles (e.g., newsprint, television, radio) to maximize some combination of *reach* (audience size), frequency, and campaign duration subject to a budget constraint (see textbooks by Rossiter and Danaher 1998 and Gensch 1973). One typical assumption is *wearout* (the effectiveness of an ad decreases as the number of exposures to the same person increases); papers in this line of research include Thompson (1981) and Simon (1982), which arrive at an optimal pulsing strategy of advertising expenditure over time. Although some of these concepts are relevant, it is most instructive to compare our model with those that take the perspective of a third party scheduling many advertisers, such as models that plan TV commercials and webpage banner ads. Structurally, in-game advertising sits between TV and Web advertising: It has well-defined contracts like TV, yet decisions are made at a very fine granularity, as on the Web.

TV Commercials. Advertisers purchase 60%–80% of the year's ad space during a two- to three-week period in May called the *up-front market*; the remaining ad space is sold first-come-first-serve in the *scatter market*. In contrast, in-game ads are sold throughout the year, so the division between up-front and scatter markets is not profound. Therefore, papers (e.g., Araman and Popescu 2010) that determine the optimal up-front versus scatter trade-off are not directly applicable.

Bollapragada et al. (2002) use goal programming to produce a sales plan for a single campaign that allocates commercial slots from TV shows such that the advertiser's preferences are honored as closely as possible. Inventory is booked for each advertiser in sequence, allowing each advertiser to request changes to their plan before it is finalized. Audience size is assumed to be deterministic and there is no mention of make-goods allocation when schedules are not executed as planned. This is a static up-front market problem; in contrast, we consider the dynamic problem in which new campaigns and games materialize over a rolling horizon. Furthermore, our Real-Time Algorithm is significantly different than the low-level ad-slotting "ISCI Rotator" algorithms used for TV because in our case each viewer can be shown different ads. Finally, Zhang (2006) also solves the (static, deterministic, up-front market) TV commercial scheduling problem, but in their case they assume that client negotiations are minimal, allowing them to simultaneously schedule all campaigns via a mixed-integer program.

Web Advertising. The most common objective in Web advertising is maximizing click-through (e.g., Chickering and Heckerman 2003, Nakamura and Abe 2005), a concept that does not apply to in-game ads. However, some papers explore the essential dynamic trade-off: choosing between serving the campaign with the highest bid versus the one farthest from achieving its contracted impression goal. Approaches include the online algorithm of Mehta et al. (2007) and the large LP solved by column generation

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

by Abrams et al. (2007). Of these, Abrams et al. (2007) include promising computational results on historical data; however, the authors note some challenges to implementing the full-scale algorithm in practice. In addition to tailoring our focus to in-game advertising, our problem definition is more comprehensive than either of these papers: we solve a multiobjective problem with many practical constraints not present in Abrams et al. (2007) or Mehta et al. (2007) (e.g., to spread impressions over time and across various ad spaces).

3. Definitions and Problem Statement

Games, Zones, Inventory Elements, and Regions. A video game is subdivided into many *zones*, each having one or more *inventory elements* (the generic name given to in-game billboards and other locations where ads are displayed). Typically, each level of a game is a separate zone. Within each zone, inventory elements are grouped into *regions* based on their spacial proximity; all inventory elements visible from the same vantage point are typically in the same region.

Requests, Arrivals, Game Instances, and Spots. When a gamer enters a new zone, his console sends a *request* to the ad server to select ad campaigns for all inventory elements in the zone. From the ad server's perspective, an *arrival* of a gamer has just occurred. This arrival spawns a *game instance*; the gamer will continue to see the ads that were selected by the ad server for this game instance until the gamer departs from this zone. Although some inventory elements can only show one ad per game instance, others (e.g., an in-game Jumbotron) cycle through a sequence of ads; we call each element of the sequence an ad *spot*.

Adtime. An inventory element in one game instance provides *adtime* equal to the total amount of time—not necessarily contiguous—that the inventory element appears on-screen. Adtime can be aggregated over multiple inventory elements and over multiple game instances; thus, we can consider quantities such as the expected adtime of Zone 3 of Game A on Monday.

Impressions. We measure billable adtime in *impressions*. Massive defines an impression as 10 seconds of time—not necessarily contiguous—in which a gamer sees the same ad campaign on-screen, possibly across multiple inventory elements in the same game instance. For example, given a zone with two single-spot inventory elements, if these elements log 7 and 8 seconds of adtime, respectively, in one game instance, then either: (1) the same campaign was served in both inventory elements, so $\lfloor(7+8)/10\rfloor = 1$ impression is counted toward that campaign's impression goal; or (2) different campaigns were served in these inventory elements, and no impressions are counted for either campaign ($\lfloor 7/10\rfloor = \lfloor 8/10\rfloor = 0$).

Paying and Nonpaying Campaigns. Paying campaigns specify an impression goal of q_k and a unit price of p_k, providing revenue of $p_k \times q_k$. We can assume that Massive receives this revenue up-front when the campaign is negotiated. Subsequent scheduling decisions affect Massive's ability to deliver the campaign; thus penalty costs (make-goods) may be incurred should campaign k not be delivered as promised. Nonpaying campaigns include house ads, public service announcements (PSAs), and default graphics; these do not generate any revenue, do not have minimum impression goals, and are served as "filler" when no paying campaigns can be served.

Campaign Targeting—Audience Demographics and Context. Paying campaigns currently specify targeting constraints along three dimensions: geography, time, and context; i.e., impressions must come from specific geographic regions, time slots, and inventory elements, respectively. The geographic dimension is indexed by 300+ designated market area (DMA) codes that allow Massive to constrain service to metropolitan areas across the United States and/or key areas in other parts of the world. There are 42 combinations of {Weekday, DayPart} tuples for the time dimension, one for each of six DayParts (contiguous groups of hours, e.g., Prime Time = 7 P.M.–12 A.M.) in each of the seven weekdays. Finally, there are 1 to 50 inventory element classes (IECs) per game that group inventory elements by context (e.g., "all virtual soda machines"). By targeting specific games, geographic regions, and times, a campaign can be tailored for specific audience demographics. In contrast, IEC targeting ensures proper *context* by integrating ads in realistic places: Coke ads, not Tide ads, belong on virtual soda machines.

Competition and Saturation. Within each game instance, (1) campaigns of competing brands (e.g., Coke and Pepsi) cannot be shown, (2) each campaign can be served in at most ω_z spots, and (3) when a campaign is served in multiple spots, the spots should preferably be in different regions. The quantity ω_z is called the *saturation cap* and is typically between a quarter to half the number of inventory elements in zone z.

Campaign Delivery Quality. In addition to a campaign's aggregate impression goal q_k, the advertiser usually specifies other impression goals either explicitly or implicitly. These include mix requirements (e.g., at least 40% of impressions should come from sports games) and delivery schedules (e.g., uniformly spread delivery over time, or deliver more impressions the week before a new product launch). Mix requirements are modeled with an impression goal $q_{(I,k)}$ for certain groups of games I that campaign k desires; we call I a *MixComponent*. Delivery schedules can include weekly impression goals q_k^t and, if needed, weekly impression goals for each MixComponent $q_{(I,k)}^t$. It is usually also important to serve each campaign impressions from many different games because part of the value of contracting with a network provider such as Massive is derived from obtaining access to advertising space in a breadth of games from different publishers.

Publisher Concerns. Because game publishers receive a portion of the revenues generated by the impressions served

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

in their games, the network provider must attempt to schedule ads across games to avoid negatively affecting the revenue stream of a particular publisher.

Supply. Inventory is not games, or zones, or inventory elements, but rather *eyeballs*: the total inventory available to fulfill a campaign is the number of impressions generated by gamers that match that campaign's targeting. We use point forecasts of *supply* in our Weekly Plan LP, which are computed from the following point estimates of adtime:

- $\bar{s}_i^t$ = expected adtime of game i, week t;
- $\bar{s}_{id}^t$ = expected adtime of game i, week t, DMA d;
- $\bar{s}_{ie}^t$ = expected adtime of game i, week t, inventory element e; and
- $\bar{s}_{iw}^t$ = expected adtime of game i, week t, {Weekday, DayPart} w.

Defining the set of zones in game i as Z_i, the set of inventory elements in zone z as E_z, and the sets of all DMAs and all {Weekday, DayPart} tuples as D and W, respectively, we compute "breakouts"—proportions of adtime that come from a single inventory element, DMA, or {Weekday, DayPart}:

- $b_{id}^t = \bar{s}_{id}^t / \sum_{d' \in D} \bar{s}_{id'}^t$ = proportion of game i, week t's adtime that comes from DMA d;
- $b_{ie}^t = \bar{s}_{ie}^t / \sum_{z \in Z_i,\, e' \in E_z} \bar{s}_{ie'}^t$ = proportion of game i, week t's adtime that comes from inventory element e; and
- $b_{iw}^t = \bar{s}_{iw}^t \big/ \sum_{w' \in W} \bar{s}_{iw'}^t$ = proportion of game i, week t's adtime that comes from {Weekday, DayPart} w.

We use these breakouts to generate estimates of supply that are used by the Weekly Plan LP. This final step requires the additional notation:

- E_{zk} = set of inventory elements in zone z that matches the targeting of campaign k;
- D_k, W_k = sets of DMAs and {Weekday, DayPart} tuples that match the targeting of campaign k, respectively;
- P_e = set of spots in inventory element e;
- s_e = expected adtime of inventory element e in one game instance;
- $s^p(e) = s_e / |P_e|$ = expected adtime of any spot $p \in P_e$, assuming each spot of P_e is equally likely to be on-screen; and
- $\varpi \in [0, 1]$ = a factor that approximates the conversion ratio between adtime and impressions. We assume that when measuring aggregate supply over many inventory elements and/or large periods of time, adtime and impressions are approximately equal modulo this scaling constant. Note that if adtime was not rounded down to compute billable impressions, then $\varpi = 1$. We used $\varpi = 0.9$, which was consistent with the aggregate amount of rounding down in our data set.

Labeling the inventory elements of E_{zk} such that $s^p(1) \geqslant s^p(2) \geqslant \cdots \geqslant s^p(|E_{zk}|)$, we define $\hat{E}_{zk} = \{1, \ldots, \omega_z\}$ as the set of the "largest" ω_z inventory elements in zone z that match the targeting of campaign k. Finally, we compute the supply estimates for the Weekly Plan LP:

- $s_i^t = \varpi \bar{s}_i^t$ = estimated number of impressions provided by game i in week t;
- $s_{ik}^t = \varpi \bar{s}_i^t (\sum_{d \in D_k} b_{id}^t)(\sum_{w \in W_k} b_{iw}^t)(\sum_{z \in Z_i} \sum_{e \in E_{zk}} b_{ie}^t)$ = estimated number of impressions provided by game i in week t that match the targeting requirements of campaign k; and
- $\hat{s}_{ik}^t = \varpi \bar{s}_i^t (\sum_{d \in D_k} b_{id}^t)(\sum_{w \in W_k} b_{iw}^t)(\sum_{z \in Z_i} \sum_{e \in \hat{E}_{zk}} b_{ie}^t)$ = estimated number of *nonsaturated* impressions provided by game i in week t that match the targeting requirements of campaign k.

Demand. The *demand* for ad space arises from ad agencies purchasing campaigns. For each week $t \in T$ of the planning horizon, Massive uses a proprietary method to estimate the distributions of aggregate demand-to-come for three groups of games: premium, middle, and discount *tiers*.

Tightness of Capacity—Sell-Through. We first define:

- ι_{ae} = adtime of inventory element e in the game instance started by arrival a (measured ex post);
- A_z^t, A_{zk}^t = sets of arrivals for zone z in week t, and for zone z in week t that match the targeting of campaign k, respectively;
- K_{ae} = set of campaigns with targeting that matches both arrival a and inventory element e; and
- I_k = set of games that match the targeting of campaign k.

Then, $\iota_k^t = \varpi \sum_{i \in I_k} \sum_{z \in Z_i} \sum_{a \in A_{zk}^t} \sum_{e \in E_{zk}} \iota_{ae}$ is the number of impressions counted ex post in week t that *could* have been used by campaign k (assuming we disregard competition and saturation constraints). The *sell-through* of inventory element e in the game instance started by arrival a in week t is $\gamma_{ae} = \sum_{k \in K_{ae}} (q_k^t / \iota_k^t)$, the proportion of inventory element e in arrival a that would have been allocated to paying campaigns *if* each campaign were served at the same rate across the entire inventory space it targeted. The population mean μ_γ and population standard deviation σ_γ describe sell-through over aggregated blocks of inventory; for example, the network sell-through over the planning horizon (i.e., computed over all games Υ and weeks T) has mean, second moment, and standard deviation:

$$\mu_\gamma = \frac{\sum_{t \in T} \sum_{i \in \Upsilon} \sum_{z \in Z_i} \sum_{a \in A_z^t} \sum_{e \in E_z} \iota_{ae} \gamma_{ae}}{\sum_{t \in T} \sum_{i \in \Upsilon} \sum_{z \in Z_i} \sum_{a \in A_z^t} \sum_{e \in E_z} \iota_{ae}},$$

$$\mu_{\gamma^2} = \frac{\sum_{t \in T} \sum_{i \in \Upsilon} \sum_{z \in Z_i} \sum_{a \in A_z^t} \sum_{e \in E_z} \iota_{ae} \gamma_{ae}^2}{\sum_{t \in T} \sum_{i \in \Upsilon} \sum_{z \in Z_i} \sum_{a \in A_z^t} \sum_{e \in E_z} \iota_{ae}},$$

$$\sigma_\gamma = \sqrt{\mu_{\gamma^2} - \mu_\gamma^2}.$$

Legacy Algorithm. Massive's existing scheduling algorithm—the *legacy algorithm*—uses a single impression goal q_k for each campaign and assumes that impressions should be spread uniformly across all weeks (i.e., it does not support delivery schedules that ramp up impressions the week before a product launch). When a new arrival spawns a game instance, the legacy algorithm checks whether each campaign k is ahead or behind schedule and adjusts the service rates accordingly. Because the legacy algorithm serves

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

campaign k at the same rate in each game i that its targeting matches, it is *usually* good at spreading impressions across MixComponents, providing impressions in a broad set of games, and satisfying publisher concerns. The most systematic shortcoming of the legacy algorithm is that it is myopic; because it does not use supply estimates, it is unable to proactively increase the service rate of a campaign if shortages are anticipated later.

4. Weekly Plan LP

We periodically solve a linear program called the *Weekly Plan* to get x^t_{ik}, the number of impressions allocated to campaign k from game i in week t. We update the target service rates $\lambda^t_{ik} = x^t_{ik}/s^t_{ik}$ whenever the Weekly Plan is resolved; the Real-Time Algorithm (see §5) uses these rates to select specific campaigns for the inventory elements of a given game instance.

The Weekly Plan is similar to the goal program of Bollapragada et al. (2002), which is used to generate TV ad sales plans at NBC. Both formulations have an objective that reserves premium inventory for future sales while minimizing penalty costs for relaxing the many goal constraints. The NBC sales plan has mix constraints and weekly weighting constraints, which take a client's goal $x = q$ (# of commercial slots assigned from this subset of inventory = target) and relaxes it to $l - y \leqslant x \leqslant u + y$, where $y \geqslant 0$ is a slack variable penalized in the objective and (l, u) are lower and upper bounds. We follow the same construction; however, our mix constraints specify an impression goal $q_{(I,k)}$ for MixComponent I (which is a group of games), whereas at NBC the mix targets are for a single TV show. We also add a weekly mix constraint based on the target $q^t_{(I,k)}$, a constraint that spreads impressions across all games within a MixComponent, and a constraint that ensures that each game is assigned a minimum dollar value in revenue. The NBC sales plan bounds the number of slots allocated to a given {Show, Week}, which partially enforces saturation and competition constraints; saturation is then handled by a heuristic they call the ISCI Rotator, which spreads commercials of the same campaign apart chronologically. We follow a similar approach by including bounds on x^t_{ik} that partially enforce the saturation cap, and defer the consideration of competition and saturation to the Real-Time Algorithm. Finally, we note that NBC's goal program is an integer program (as discrete commercial slots are allocated) that generates a sales plan for a single campaign, whereas we allocate impressions for many campaigns simultaneously using a linear program.

The Weekly Plan LP appears in Figure 1. For compactness, some quantities are shown with lower and upper bounds within the same constraint; hence, slack variables exist on both sides of some constraints. The variables are λ, v, w, x, y, z; all other quantities are constants; see Table 1 for the full list of notation. Because we re-solve the Weekly Plan LP over time, we divide time into two parts:

Figure 1. The Weekly Plan LP.

$$\max \; -\sum_{k\in K}\pi_k y_k - \sum_{k\in K, t\in T_k}\beta_k y^t_k - \sum_{k\in K, I\in\Phi_k}\tau_k y_{(I,k)} - \sum_{k\in K, I\in\Phi_k, t\in T_k}\eta_k y^t_{(I,k)} - \sum_{k\in K, I\in\Phi_k, t\in T_k}\xi_k v^t_{(I,k)} - \sum_{t\in T, i\in\Upsilon}\zeta_i z^t_i + \sum_{I\in\Psi, t\in T, m=1..n}\theta^t_{(I,m)}\dot{w}^t_{(I,m)}$$

$$\text{s.t.} \quad \dot{x}_k + y_k = (q_k - a_k)^+ \quad \forall k\in K, \tag{1}$$

$$l^t_k - y^t_k \leqslant x^t_k \leqslant u^t_k + y^t_k \quad \forall t\in T_k, \forall k\in K, \tag{2}$$

$$l_{(I,k)} - y_{(I,k)} \leqslant x_{(I,k)} \quad \forall I\in\Phi_k, \forall k\in K, \tag{3}$$

$$l^t_{(I,k)} - y^t_{(I,k)} \leqslant x^t_{(I,k)} \quad \forall I\in\Phi_k, t\in T_k, \forall k\in K, \tag{4}$$

$$s^t_{ik}\lambda^t_{(I,k)} - v^t_{(I,k)} \leqslant x^t_{ik} \leqslant s^t_{ik}\lambda^t_{(I,k)} + v^t_{(I,k)} \quad \forall i\in I, \forall I\in\Phi_k, t\in T_k, \forall k\in K, \tag{5}$$

$$\sum_{k\in K} p_k x^t_{ik} + z^t_i \geqslant r^t_i \quad \forall t\in T, i\in\Upsilon, \tag{6}$$

$$\dot{x}^t_i + \dot{w}^t_i = \dot{s}^t_i \quad \forall t\in T, i\in\Upsilon, \tag{7}$$

$$\dot{x}^t_{ik} \leqslant \dot{\hat{s}}^t_{ik} \quad \forall i\in I_k, t\in T_k, \forall k\in K, \tag{8}$$

$$x^t_{ik} - \dot{x}^t_{ik} = a^t_{ik} \quad \forall i\in I_k, t\in T_k, \forall k\in K, \tag{9}$$

$$\dot{w}^t_{(I,m)} \leqslant \dot{\vartheta}^t_{(I,m)} \quad \forall I\in\Psi, m = 1..(n-1), t\in T, \tag{10}$$

$$\sum_{m=1..n}\dot{w}^t_{(I,m)} - \sum_{i\in I}\dot{w}^t_i = 0 \quad \forall I\in\Psi, t\in T, \tag{11}$$

$$\dot{x}_k - \sum_{i\in I_k, t\in T_k}\dot{x}^t_{ik} = 0 \quad \forall k\in K, \tag{12}$$

$$x_k - \dot{x}_k = a_k \quad \forall k\in K, \tag{13}$$

$$x^t_k - \sum_{i\in I_k} x^t_{ik} = 0 \quad \forall t\in T_k, \forall k\in K, \tag{14}$$

$$x_{(I,k)} - \sum_{i\in I, t\in T_k}\dot{x}^t_{ik} = a_{(I,k)} \quad \forall I\in\Phi_k, \forall k\in K, \tag{15}$$

$$x^t_{(I,k)} - \sum_{i\in I} x^t_{ik} = 0 \quad \forall I\in\Phi_k, t\in T_k, \forall k\in K, \tag{16}$$

$$\dot{x}^t_i - \sum_{k\in K}\dot{x}^t_{ik} = 0 \quad \forall t\in T, i\in\Upsilon, \tag{17}$$

$$\lambda, v, w, x, y, z \geqslant 0 \quad \text{for all forms of these variables.} \tag{18}$$

The constant a denotes all impressions of x *achieved* up to the present, and $\dot{x}$ denotes all impressions of x planned into the future. Hence, $x = a + \dot{x}$. Similarly, dots on other constants and variables indicate quantities for the *remainder* of the relevant time interval.

Constraints (1) through (5) are goal-type constraints that model campaign contract requirements that may be relaxed and penalized in the objective: Constraint (1) (end-of-horizon impression goal) makes sure that for each campaign k, the number of impressions planned in the *remainder* of the horizon $\dot{x}_k$, plus the planned shortfall y_k, must equal the *remaining* end-of-horizon impression goal $(q_k - a_k)^+$; constraint (2) (weekly impression goals) ensures that the number of planned impressions x^t_k is between the bounds l^t_k and u^t_k for each campaign k and week t; constraint (3) (end-of-horizon mix target) ensures that at least $l_{(I,k)}$ impressions from MixComponent I are allocated to campaign k; and constraint (4) (weekly mix targets) ensures that at least $l^t_{(I,k)}$ impressions from Mix-

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Table 1. Notation for Weekly Plan LP.

Indices

i, k, t = a game, a campaign, a week,
m = a segment of the piecewise-linear expected revenue function for unused inventory,
I = a set of games (can be either a MixComponent or a tier).

Sets

Υ, K, T = all games, all paying campaigns, all weeks,
I_k = games that campaign k can be served in,
T_k = weeks that campaign k can be served in.

Collections (sets of sets)

Φ = all MixComponents,
Φ_k = MixComponents for campaign k,
Ψ = all tiers of games.

Parameters

a_k, $a_{(I,k)}$, a^t_{ik} = # of impressions served to date for campaign k, for campaign k in MixComponent I, and for campaign k in game i during week t,
l^t_k, u^t_k = bounds on the # of impressions to show for campaign k in week t,
$l_{(I,k)}$, $l^t_{(I,k)}$ = lower bounds for the # of impressions to serve to campaign k of MixComponent I over all weeks, to campaign k of MixComponent I in week t,
n = # of piecewise-linear segments in expected revenue function,
p_k = price per impression paid by campaign k,
q_k = end-of-horizon impression goal for campaign k,
r^t_i = minimum revenue amount for game i, week t,
s^t_i = expected supply of game i in week t,
s^t_{ik} = expected supply of game i in week t that matches campaign k,
$\hat{s}^t_{ik}$ = expected nonsaturated supply of game i in week t that matches campaign k,
$\theta^t_{(I,m)}$ = marginal revenue in the mth segment of the piecewise-linear expected revenue function for tier I, week t,
$\vartheta^t_{(I,m)}$ = width of the mth segment of the piecewise-linear expected revenue function for tier I, week t.

Slack Variables and Penalty Costs

$v^t_{(I,k)}$, ξ_k = maximum absolute deviation from the nominal number of impressions to serve campaign k in game i at week t, taken over all games i in MixComponent I, penalty cost for not spreading impressions across all games in a MixComponent,
y_k, π_k = shortfall amount, penalty cost for the end-of-horizon impression goal for campaign k,
y^t_k, β_k = amount bounds violated, penalty cost for the impression goal for campaign k, week t,
$y_{(I,k)}$, τ_k = amount bounds violated, penalty cost for the end-of-horizon impression goal for campaign k, MixComponent I,
$y^t_{(I,k)}$, η_k = amount bounds violated, penalty cost for the impression goal for campaign k, MixComponent I, week t,
z^t_i, ζ_i = shortfall amount, penalty cost for the minimum revenue target of game i, week t.

Decision Variables

w^t_i = impressions of game i available to satisfy future demand in week t,
$w^t_{I,m}$ = impressions of tier I available to satisfy future demand in week t, placed in segment m of the piecewise-linear expected revenue function,
x_k = impressions allocated to campaign k over the planning horizon,
x^t_k = impressions allocated to campaign k from week t,
x^t_i = impressions allocated from game i, week t,
x^t_{ik} = impressions allocated to campaign k from game i, week t,
$x_{(I,k)}$ = impressions allocated to campaign k from MixComponent I over the planning horizon,
$x^t_{(I,k)}$ = impressions allocated to campaign k from MixComponent I, week t,
$\lambda^t_{(I,k)}$ = nominal rate to serve impressions of MixComponent I to campaign k in week t.

Notes. Our model also makes use of some quantities not listed in the above table, but that are related to the quantities listed above. In particular, quantities which we denote with a dot superscript are analogs of their nondotted cousins. The dotted quantity measures the component of the nondotted quantity that applies from the current time (of solving the Weekly Plan LP) until the end of the relevant time period. For example, $\dot{x}^t_{ik}$ is the analog of x^t_{ik}. Whereas x^t_{ik} measures the number of impressions allocated to campaign k from game i, week t, the quantity $\dot{x}^t_{ik}$ is the number of impressions allocated to campaign k from game i in the *remainder* of week t. In the case that a portion of week t has elapsed at the time the Weekly Plan LP is being solved, the x^t_{ik} and $\dot{x}^t_{ik}$ do not coincide and in fact are related by the expression $x^t_{ik} = a^t_{ik} + \dot{x}^t_{ik}$.

Component I are allocated to campaign k in week t. Constraint (5) (spread impressions to all games within a MixComponent) models the following idea: "For each {MixComponent, Campaign, Week}, try to set the same service rate for all games." Defining the service rate for a {MixComponent, Campaign, Week} as the variable $\lambda^t_{(I,k)}$, serving all games in a MixComponent at the same rate requires satisfying $\lambda^t_{ik} = \lambda^t_{(I,k)}\ \forall i \in I$. By multiplying $x^t_{ik}/s^t_{ik} = \lambda^t_{(I,k)}$ through by s^t_{ik} and introducing the slack $v^t_{(I,k)}$, we obtain constraint (5). Note that $s^t_{ik}\lambda^t_{(I,k)}$ can be

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

interpreted as the nominal number of impressions to plan in game i, and so $v^t_{(I,k)}$ penalizes the maximum absolute deviation from the nominal number of impressions across all games in MixComponent I.

Constraint (6) (minimum revenue target for each game) is the only goal-type constraint that models publisher contract requirements. This constraint ensures that enough impressions are allocated to game i in week t to guarantee at least r^t_i dollars in revenue.

Constraint (7) (supply) states that for the *remaining* portion of week t in game i, the number of impressions planned, plus the number of impressions left unplanned, must equal the forecast supply.

Constraint (8) (saturation cap bound) partially enforces the saturation cap by bounding the number of impressions assigned to campaign k in game i, week t by the estimated nonsaturated supply $\hat{s}^t_{ik}$.

Constraint (9) (link total planned impressions with remaining planned impressions) ensures that for each {Game, Campaign, Week}, the total number of impressions planned, x^t_{ik}, must equal the actual number of impressions achieved up to the present, a^t_{ik}, plus the number of impressions planned from the present until the end of the week, $\dot{x}^t_{ik}$. For weeks $t \geqslant 2$, we always have $a^t_{ik} = 0$, and so this constraint reduces to $x^t_{ik} = \dot{x}^t_{ik}$.

Constraints (10)–(11) model a piecewise-linear component of the objective function and are described with the objective in §4.1. Constraints (12)–(17) link the various forms of x. In an actual implementation, only the variables x^t_{ik} and $\dot{x}^t_{ik}$ are required; the other forms of x can be written as linear combinations of x^t_{ik} and $\dot{x}^t_{ik}$. We use the other forms of x to keep constraints (1)–(11) readable.

Constraints (3) and (4) do not have upper bounds because MixComponents are not mutually exclusive: game i may be both a sports game and a racing game; in that case, impressions allocated to game i count toward *two* mix targets. Had upper bounds existed, they may need to be violated to satisfy the end-of-horizon goal (constraint (1)).

4.1. The Objective Function

The objective is to maximize revenues that can be affected by scheduling; it contains two terms.

Objective Term 1. Penalty costs for breaking contracts is:

$$-\sum_{k\in K}\pi_k y_k - \sum_{k\in K,\, t\in T_k}\beta_k y^t_k - \sum_{k\in K,\, I\in\Phi_k}\tau_k y_{(I,k)} - \sum_{k\in K,\, I\in\Phi_k,\, t\in T_k}\eta_k y^t_{(I,k)} - \sum_{k\in K,\, I\in\Phi_k,\, t\in T_k}\xi_k v^t_{(I,k)} - \sum_{t\in T,\, i\in\Upsilon}\zeta_i z^t_i.$$

We model the costs of falling short of the goals modeled by constraints (1)–(6) as negative revenues from the delivery of make-goods for broken contracts. Make-good costs include direct compensation for underdelivery, transaction costs, and loss of goodwill. Direct compensation is either the dollar amount refunded to the advertiser, or the (shadow) cost of assigning additional inventory to extend the campaign past its scheduled end date. Transaction costs include all manual processing to issue the make-good, including getting approval to extend the campaign. Loss of goodwill models the advertiser's displeasure with the quality of service they received and reflects the estimated loss of repeat business. Transaction costs and goodwill costs are likely significant, yet are extremely hard to estimate; fortunately, with regard to linear programming, exact estimation of the penalty costs is not as important as their relative ranking. Massive suggested the following penalty costs, which are proportional to campaign prices p_k: $\pi_k = p_k$ per impression short of the end-of-horizon goal for campaign k; $\beta_k = 0.1p_k$ per impression over/under the allowed deviation from the weekly goal; $\tau_k = 0.1p_k$ over/under the allowed deviation from the end-of-horizon goal for each MixComponent; $\eta_k = 0.05p_k$ over/under the allowed deviation from the weekly goal for each MixComponent; and $\xi_k = 0.05p_k$ for not spreading impressions evenly across all games in a MixComponent. We also used $\zeta_i \in \{0.5, 0.25, 0.1\}$ as the per-dollar penalties of falling short of the minimum revenue target of game i (penalties are differentiated by tier).

Note that in practice, make-good costs should also be increased for preferred customers (which are often given quantity discounts) to compensate for the fact that the above formulas assign low penalty costs to campaigns with low prices.

Objective Term 2—General Form. Revenue potential of unscheduled inventory is:

$$\sum_{I\in\Psi,\, t\in T} f^t_I(w^t_I).$$

Because supply and demand change over time, the scarcity of inventory, and therefore the *value* of unscheduled inventory, will change over time. By scheduling appropriately, we maximize the total dollar value of unscheduled inventory, i.e., the expected future sales revenue. The general form of objective term 2 is as listed above, where Ψ is the set of all tiers of games (premium, middle, discount), T is the set of all weeks, and f^t_I is a function that values the quantity of unscheduled inventory w^t_I for tier I in week t. Suppressing subscripts and superscripts, the value function for unscheduled inventory $f(w)$ of a particular {Tier, Week} is:

$$f(w) := p\mathbb{E}[\min(X, w)] = p\mathbb{E}[X - (X - w)^+] = p\int_0^w \bar{G}(x)\,dx,$$

where p is the fixed price for the type of inventory under consideration, X is a random variable that models market demand, $G(x)$ is the cumulative distribution function of demand, and $\bar{G}(x) = 1 - G(x)$. Thus, if we leave w units of inventory available, expected sales is $\mathbb{E}[\min(X, w)]$, yielding $f(w)$ dollars in expected revenue.

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Because $f'(w) = p(d/dw)\int_0^w \bar{G}(x)\,dx = p\bar{G}(w)$ and $\bar{G}$ is a nonincreasing function, $f(w)$ is concave increasing for all demand distributions of X. Therefore, we can approximate $f(w)$ by a piecewise-linear function with n segments of successively smaller positive slope. Denoting the marginal revenues (slopes) as θ_m and segment widths as ϑ_m for the $m = 1..n$ segments, the expected revenue from future sales is modeled by the linear form:

Objective Term 2—Linear Form. The Weekly Plan LP uses this linear form in practice.

$$\sum_{I\in\Psi,\, t\in T,\, m=1..n} \theta^t_{(I,m)}\dot{w}^t_{(I,m)}$$

and is accompanied by constraint (10) (bound the unplanned impressions in each segment) and constraint (11) (link unplanned impressions by {Tier, Week, Segment} to {Game, Week}).

Because the slope parameters $\theta^t_{I,m}$ are decreasing in m and we are maximizing, $\dot{w}^t_{I,m} = 0 \Longrightarrow \dot{w}^t_{I,m+1} = 0$. The bounds on $\dot{w}^t_{(I,m)}$, namely $\dot{\vartheta}^t_{(I,m)}$, ensure that impressions are accounted for in the correct segment, where $\dot{\vartheta}^t_{(I,m)}$ is defined as follows (ρ is the percentage of week 1 remaining):

$$\dot{\vartheta}^t_{(I,m)} := \begin{cases} \rho\vartheta^t_{(I,m)} & \text{for } t = 1, \\ \vartheta^t_{(I,m)} & \text{for } t \geqslant 2. \end{cases}$$

We assumed stationarity in the tests we performed, and computed the required cdfs from historical data. In practice, we expect the historical cdf to be used as a baseline that Massive can manually align with their sales projections (keeping the general shape of the cdf the same, but shifting its mean). For an example that uses a discrete demand distribution to compute θ and ϑ, see §EC.1 of the electronic companion to this paper, available as part of the online version at http://or.journal.informs.org/.

Alternately, when only $\mathbb{E}[X]$ is specified, we can use $G(x) = \{0 \text{ if } x < \mathbb{E}[X],\ 1 \text{ if } x \geqslant \mathbb{E}[X]\}$ to compute $f(w \mid \mathbb{E}[X]) = \{w \text{ if } x < \mathbb{E}[X],\ \ \mathbb{E}[X] \text{ if } x \geqslant \mathbb{E}[X]\}$. By Jensen's Inequality, we know $f(w \mid \mathbb{E}[X]) \equiv \mathbb{E}[\min(\mathbb{E}[X], w)] \geqslant \mathbb{E}[\min(X, w)] \equiv f(w)$; thus, $f(w)$ is a function that is bounded above by $f(w \mid \mathbb{E}[X])$, yet has similar asymptotic behavior; i.e., $f(w) \to w$ as $w \to 0$, and $f(w) \to \mathbb{E}[X]$ as $w \to \infty$. An appropriately chosen spline can be approximated with a few piecewise-linear segments to get the piecewise-linear form of $f(w)$ used by our model.

Finally, we point out that in theory, censored demand can be handled by the Kaplan-Meier estimator (Talluri and van Ryzin 2005); however, this is not necessary when working with tiers: individual games may get sold out, but because tiers are large aggregations of games, excess inventory usually exists.

5. Real-Time Algorithm

Each arrival to the ad server invokes the Real-Time Algorithm, which assigns campaigns to inventory elements in the player's game instance. The Real-Time Algorithm makes this allocation in accordance with the service rates λ^t_{ik} computed by the Weekly Plan LP, while also obeying campaign targeting, saturation, and competition constraints.

The Real-Time Algorithm serves the same purpose as the ISCI Rotator Algorithm for scheduling commercials in broadcast television (Bollapragada et al. 2004, Bollapragada and Garbiras 2004). A broadcast TV sales plan (analogous to our Weekly Plan) specifies the set of commercials to display during each airing of each show, leaving the ISCI Rotator to choose, for each ad, the commercial breaks and positions within those breaks where each ad should be aired. The ISCI Rotator allocates ads to slots such that saturation is managed (two airings of the same commercial are as evenly spaced as possible) and competition constraints are obeyed (competing brands are not displayed in the same commercial break). Although there are some high-level similarities, the Real-Time Algorithm operates differently than the ISCI Rotator. This is because targeting constraints add complexity to the Real-Time Algorithm and because the Real-Time Algorithm sacrifices optimality for speed: it is run millions of times per day compared to the ISCI Rotator's nightly execution.

The Real-Time Algorithm performs two operations: (1) it adjusts the service rates for the current game instance, and (2) assigns campaigns to inventory elements in accordance with the adjusted service rates. Notation for the Real-time Algorithm is summarized in Table 2.

5.1. Stage 1: Adjust Service Rates for the Current Game Instance

An arrival of type $a = \{\text{Zone, DMA, Weekday, DayPart}\}$ is initially assigned the service rate matrix Λ_a, the elements of which are

$$\lambda_{ck} := \begin{cases} \lambda^1_{i(a),k} & \text{if campaign } k \text{ matches the targeting} \\ & \text{of both IEC } c \text{ and the \{Zone, DMA,} \\ & \text{Weekday, DayPart\} specified by } a, \\ 0 & \text{otherwise,} \end{cases}$$

where $i(a)$ is the game that includes the zone specified in a, and $\lambda^1_{i(a),k}$ is the service rate for campaign k, game $i(a)$, week 1 from the Weekly Plan.

Algorithm 1 modifies the rate matrix Λ_a to satisfy (1) competition constraints and (2) the constraint $\sum_{k\in K_{ac}} \lambda_{ck} \leqslant 1$ $\forall\, c \in C_z$, which states that for each IEC c in the set of IECs C_z that intersect the current zone z, the sum of the service rates of campaigns K_{ac} eligible for service to this {Arrival, IEC} must not exceed 1. In Step 1, we compute a weighted average rate $\lambda^{WEIGHTED}_k$ for each campaign over all IECs in the zone. Next, we use the weighted average rates to assign one random campaign from each set of competing brands a nonzero service rate (Steps 2–5). We have assumed that the groups of competing brands are mutually exclusive; however, a straightforward extension of this algorithm applies to overlapping groups of competing brands. Finally, in Steps 6–12, the algorithm decreases the

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Table 2. Notation for the Real-Time Algorithm.

Indices

a = an arrival,
b, b', b^* = a bucket,
b_u = the "unpaid" bucket,
c = an IEC,
k, k', k'', k^* = a campaign,
p, p^* = a spot,
r = a region,
u = all nonpaying campaigns.

Functions of Indices

$b(c, k)$ = bucket for IEC c, paying campaign k,
$c(p)$ = IEC corresponding to spot p,
$k(b)$ = campaign corresponding to bucket b,
$e(p)$ = inventory element corresponding to spot p,
$r(p)$ = region corresponding to spot p.

Input: Sets

C_z = IECs in zone z,
E_z = inventory elements in zone z,
E_{cz} = inventory elements in zone z and IEC c,
K_a = campaigns with targeting matching arrival a,
K_{ac} = campaigns with targeting matching both arrival a and IEC c,
$P_b \equiv P_{c(b),z}$ = spots that can be placed in bucket b,
P_e = spots of inventory element e,
$P_z = \bigcup_{e \in E_z} P_e$ = spots in zone z,
$P_{cz} = \bigcup_{e \in E_{cz}} P_e$ = spots in zone z and IEC c,
R_z = regions in zone z.

Input: Collections (sets of sets)

$\mathcal{K}_a$ = groups of campaigns that are competing brands with targeting that matches arrival a.

Input: Parameters

s_e = expected adtime of inventory element e,
$s^p = s_{e(p)}/|P_{e(p)}|$ = expected adtime of spot p,
$s_{cz} = \sum_{e \in E_{cz}} s_e$ = expected adtime of IEC c in one game instance of zone z,
χ_k = contracted end date of campaign k,
ω_z = saturation cap of zone z.

Inputs/Variables (inputs that get adjusted)

λ_{ck} = service rate for campaign k in IEC c,
Λ_a = service rate matrix for arrival a.

Variables: Scalars

$n_{r,k}$ = number of times a spot from region r was allocated to campaign k,
v_b = unallocated expected adtime of bucket b,
x_p = unallocated expected adtime of spot p,
$y_{b,p}$ = expected adtime of spot p allocated to bucket b,
$\lambda_k^{WEIGHTED}$ = weighted average service rate for campaign k, computed over all IECs,
ϕ, ψ = used in local computations.

Variables: Sets

K^* = campaigns of competing brands,
K^U = campaigns strictly under the saturation cap,
B = buckets for paying campaigns,
P^* = spots p^* that can be placed in the chosen bucket b', *and* are in a region $r(p^*)$ where we have allocated campaign $k(b')$ the fewest number of times,
P^{**} = spots not completely allocated,
$\tilde{P}_k$ = spots in which campaign k should be served.

service rates of some campaigns (if necessary) in order for $\sum_{k \in K_{ac}} \lambda_{ck} \leqslant 1 \ \forall c \in C_z$ to hold; note that the campaigns with the latest end dates (which have the most time to catch up later) are chosen. See §EC.2.1 for a detailed example.

Algorithm 1 (AdjustServiceRates)

1: $\lambda_k^{WEIGHTED} \leftarrow \sum_{c \in C_z} s_{cz} \lambda_{ck} / \sum_{c \in C_z} s_{cz} \ \forall k \in K_a$
2: **for all** $K^* \in \mathcal{K}_a$ **do**
3: $k' \leftarrow \{k \in K^*$ with probability $\lambda_k^{WEIGHTED} / \sum_{k^* \in K^*} \lambda_{k^*}^{WEIGHTED}\}$
4: $\lambda_{ck'} \leftarrow \sum_{k \in K^*} \lambda_{ck} \ \forall c \in C_z$, and $\lambda_{ck} \leftarrow 0 \ \forall k \in K^* \backslash \{k'\}, \ c \in C_z$
5: **end for**
6: **for all** $c \in C_z$ **do**

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Label the campaigns $k \in K_{ac}$ such that $\chi_1 \geqslant \chi_2 \geqslant \cdots \geqslant \chi_{|K_{ac}|}$
$\phi \leftarrow (\sum_{k \in K_{ac}} \lambda_{ck} - 1)^+$; $k \leftarrow 1$
while $\phi > 0$ **do**
  $\psi \leftarrow \min\{\lambda_{ck}, \phi\}$; $\lambda_{ck} \leftarrow \lambda_{ck} - \psi$; $\phi \leftarrow \phi - \psi$; $k \leftarrow k + 1$
end while
end for
return Λ_a

5.2. Stage 2: Assign Campaigns to Inventory Elements

Let K_a be the set of paying campaigns with targeting that matches arrival a, and let u represent all unpaid campaigns. For each campaign $k \in K_a \cup \{u\}$, Algorithm 2 outputs a set of spots $\tilde{P}_k$ in which campaign k should be served. The algorithm begins by creating a set of buckets B; the size of bucket $b(c, k) \in B$ is the expected adtime for paying campaign k in IEC c, assuming campaign k is served at rate λ_{ck}. An infinitely large bucket $b_u \notin B$ called the "unpaid bucket" is also created; it is used when service rates are too low to assign all spots to paying campaigns. The main variables are: v_b = unallocated expected adtime of bucket b; x_p = unallocated expected adtime of spot p; $y_{b,p}$ = expected adtime of spot p allocated to bucket b; and $n_{r,k}$ = number of times a spot from region r has been allocated to campaign k. We will sometimes use index functions such as $r(p)$ = region corresponding to spot p; and $k(b)$ = campaign corresponding to bucket b. Recall from §3 that the expected adtime of any spot p of inventory element e is $s^p \equiv s^p(e) = s_e/|P_e|$. Steps 1–5 comprise the initialization phase.

Algorithm 2 (AssignCampaigns)

1: $v_{b_u} \leftarrow \infty$; $y_{b_u,p} \leftarrow 0\ \forall p \in P_z$; $B \leftarrow \varnothing$
2: **for all** $c \in C_z$, $k \in K_{ac}$ **do**
3: Create bucket $b \equiv b(c, k)$; $v_b \leftarrow \lambda_{ck} s_{cz}$; $y_{b,p} \leftarrow 0\ \forall p \in P_z$; $B \leftarrow B \cup \{b\}$
4: **end for**
5: $x_p \leftarrow s^p\ \forall p \in P_z$; $n_{r,k} \leftarrow 0\ \forall r \in R_z, k \in K_a$
6: **while** $\sum_{b \in B} v_b > 0$ **do**
7: $b' \leftarrow \{b \in B$ with probability $v_b / \sum_{b^* \in B} v_{b^*}\}$
8: $P^* \leftarrow \{p^* \in P_{b'} \mid n_{r(p^*),k(b')} = \min_{p \in P_{b'}} n_{r(p),k(b')}\}$; $p' \leftarrow \arg\max_{p \in P^*} x_p$
9: $\psi \leftarrow \min\{v_{b'}, x_{p'}\}$; $y_{b',p'} \leftarrow y_{b',p'} + \psi$; $v_{b'} \leftarrow v_{b'} - \psi$; $x_{p'} \leftarrow x_{p'} - \psi$; $n_{r(p'),k(b')} \leftarrow n_{r(p'),k(b')} + 1$
10: **end while**
11: $P^{**} \leftarrow \{p \in P_z \mid x_p > 0\}$
12: **if** $P^{**} \neq \varnothing$ **then**
13: $y_{b_u,p} \leftarrow x_p\ \forall p \in P^{**}$, $x_p \leftarrow 0\ \forall p \in P^{**}$
14: **end if**
15: $\tilde{P}_k \leftarrow \varnothing\ \forall k \in K_a \cup \{u\}$
16: **for all** $p \in P_z$ **do**
17: $b' \leftarrow \{b \in B \cup \{b_u\}$ with probability $y_{b,p}/s^p\}$
18: $y_{b',p} \leftarrow s^p$; $y_{b,p} \leftarrow 0\ \forall b \in B \cup \{b_u\} \setminus \{b'\}$; $\tilde{P}_{k(b')} \leftarrow \tilde{P}_{k(b')} \cup \{p\}$
19: **end for**
20: **for all** $k' \in \{k\colon |\tilde{P}_k| > \omega_z, k \neq u\}$ **do**
21: **while** $|\tilde{P}_{k'}| > \omega_z$ **do**
22: $p' \leftarrow \arg\min_{p \in \tilde{P}_{k'}} s^p$; $K^U \leftarrow \{k\colon |\tilde{P}_k| < \omega_z, k \neq u\}$
23: **if** $\sum_{k \in K^U} \lambda_{c(p'),k} > 0$ **then**
24: $k'' \leftarrow \{k \in K^U$ with probability $\lambda_{c(p'),k} / \sum_{k^* \in K^U} \lambda_{c(p'),k^*}\}$
25: **else**
26: $k'' \leftarrow u$
27: **end if**
28: $\tilde{P}_{k'} \leftarrow \tilde{P}_{k'} \setminus \{p'\}$; $\tilde{P}_{k''} \leftarrow \tilde{P}_{k''} \cup \{p'\}$
29: **end while**
30: **end for**
31: **return** $\tilde{P}$

In Steps 6–10, spots are fractionally assigned to buckets $b \in B$; i.e., a spot may be assigned to more than one bucket at this stage. At each iteration of the loop, we randomly draw a nonfull bucket b' and a noncompletely assigned spot p', and assign a portion (or all, if possible) of spot p' to bucket b'. We bias toward assigning large spots to large buckets, which achieves a tight packing; however, we do not always pick the largest bucket and spot: buckets are chosen randomly so the visual placement of ads within the game is randomized, and spots are chosen to spread multiple copies of the same campaign to spots in different regions. The chosen spot p' is the spot with the largest amount of unallocated expected adtime among those in the set P^*, where P^* is the set of all spots p^* that can be placed in chosen bucket b' *and* are in a region $r(p^*)$ where we have allocated campaign $k(b')$ the fewest number of times.

In Steps 11–14, any spots that did not fit into the buckets for paying campaigns B are assigned (in whole or in part) to the unpaid bucket b_u. It can be shown that when the algorithm is initialized, $\sum_{b \in B} v_b \leqslant \sum_{p \in P_z} x_p$, so all buckets $b \in B$ are always filled at this point.

In Steps 15–19, the algorithm executes randomized rounding (see Raghavan and Tompson 1987): each spot p gets assigned in whole to bucket b (and removed from all other buckets) with probability $y_{b,p}/s^p$ (the proportion of spot p currently assigned to bucket b). At this point, buckets may be perfectly filled, overfilled, or underfilled—all cases are feasible. A perfectly filled (respectively, overfilled/underfilled) bucket $b(c, k)$ corresponds to serving campaign k in IEC c exactly at (respectively, above/below) the service rate λ_{ck} in expectation.

In Steps 20–30, the algorithm adjusts the final solution $\tilde{P}_k$ (the set of spots in which to serve campaign k) if the saturation cap is exceeded (the same campaign appears in more than ω_z spots). We reassign the smallest spots p' to minimize the resulting change to the service rates; we first reassign spots in whole to paying campaigns that are under the saturation cap, drawn proportionally to $\lambda_{c(p'),k}$

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

when there are multiple such campaigns, and then to the unpaid campaign bucket. See §EC.2.2 for an example of Algorithm 2.

5.3. Properties of the Solution

At its conclusion, the Real-Time Algorithm has assigned a campaign to each spot such that:

1. DMA, Weekday, DayPart, and IEC targeting are satisfied because only campaigns that target the current arrival a receive nonzero service rates;
2. Competition constraints are satisfied, due to Algorithm 1;
3. The saturation cap is satisfied and local oversaturation is controlled, due to Algorithm 2;
4. Each campaign is served in expectation at the rate specified by the Weekly Plan LP (except perhaps for a few campaigns with later due dates that had their service rates reduced by Steps 6–12 of Algorithm 1). Thus, in expectation, our algorithm meets the impression goals, mix requirements, spread constraints, and revenue targets specified in the Weekly Plan while maximizing revenues affected by scheduling decisions. This follows because, excluding adjustments made to satisfy the saturation cap, Algorithm 2 fills buckets exactly to their fill lines in expectation; and
5. Variance in the service rates is low because buckets are packed close to their fill lines by using small spots to finish the packing. (These are also the spots that get reassigned by randomized rounding and to enforce the saturation cap.)

6. Experimental Findings

6.1. Primary Benefit

Using 26 weeks of historical data, we experimentally evaluate (backtest) the performance of our algorithm using a version of Massive's legacy algorithm that was provided as the benchmark. We set the horizon of the Weekly Plan LP at 13 weeks; therefore, at each point during the backtest we need 13 weeks of inventory supply forecasts (for games, as well as inventory elements, DayPart/Weekdays, and DMAs within games) as input. We generate forecasts from actual data by applying a random multiplicative noise term, the magnitude of which is parameterized by α. We backtest through weeks $t = 1..14$, using generated forecasts from weeks t through $t + 12$; the Weekly Plan LP is re-solved hourly.

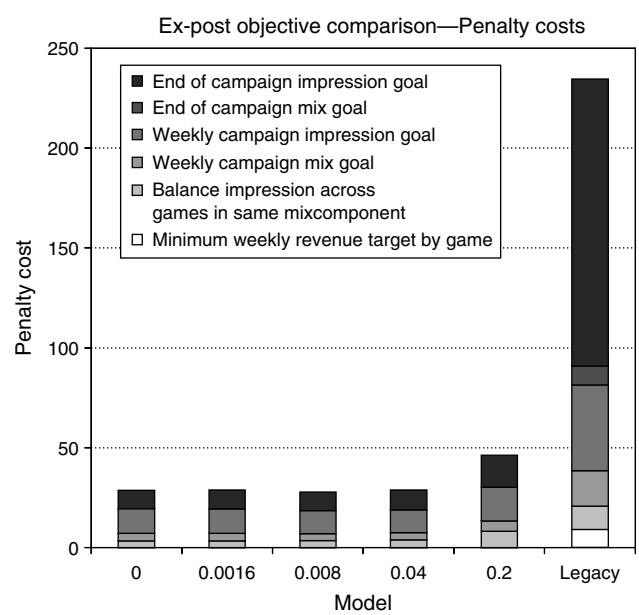
We use five forecast instances, which range from a perfect forecast ($\alpha = 0$) to a highly variable forecast ($\alpha = 0.2$). The full description of our forecast generation scheme, which produces adtime estimates $\bar{s}_i^t$, $\bar{s}_{id}^t$, $\bar{s}_{ie}^t$, and $\bar{s}_{iw}^t$, defined in §3, is given in §EC.3. To summarize, we generate multiplicative forecast errors for a forecast h weeks from today by taking the product of h lognormal random variables; the lognormal factors have median e^{μ}, where μ is a random variable dependent on α with $\mathbb{E}[\mu] = 0$. When $\mu = 0$, forecasts for one week into the future have a 95% confidence interval of $[1/\varsigma, \varsigma]$, where $\varsigma = e^{1.96\sigma}$ and $\sigma = 2\alpha$ (this confidence interval is symmetric on the logarithmic scale). For $\alpha = 0.04$ and $\alpha = 0.2$, this confidence interval is $[0.85, 1.17]$ and $[0.46, 2.19]$, respectively; i.e., when $\alpha = 0.2$ and $\mu = 0$, a one-week advance forecast for the actual value of 1,000 is between 460 and 2,190 with probability 0.95. Also, for the same α, confidence intervals grow wider as the forecasted period moves farther into the future.

Our tests on historical data show that we significantly reduce make-good costs and increase sales revenue potential relative to the legacy algorithm. Make-good costs (objective term 1 in §4.1) decrease by 80%–87%, depending on the magnitude of forecast error (see Figure 2). In the most conservative case of $\alpha = 0.2$ (it is expected that forecasts are at least this accurate), make-good costs decrease by 80%.

Revenues from the future sales of ad space (objective term 2 in §4.1) cannot be evaluated directly: because we are backtesting, the sales process in our actual data cannot be affected by our scheduling algorithm. Therefore, we use the distribution of the *age* of served impressions, the time between the sale of the campaign and when the impression was served, as a proxy.

A good scheduling algorithm will try to use the least costly inventory to satisfy existing campaigns, leaving premium inventory reserved for future sales. Therefore, a good scheduling algorithm should shift impressions from tier 1 (premium) games into tier 2 (middle) or tier 3 (discount) games when mix constraints allow. When impressions cannot be appreciably shifted between tiers, a good scheduling

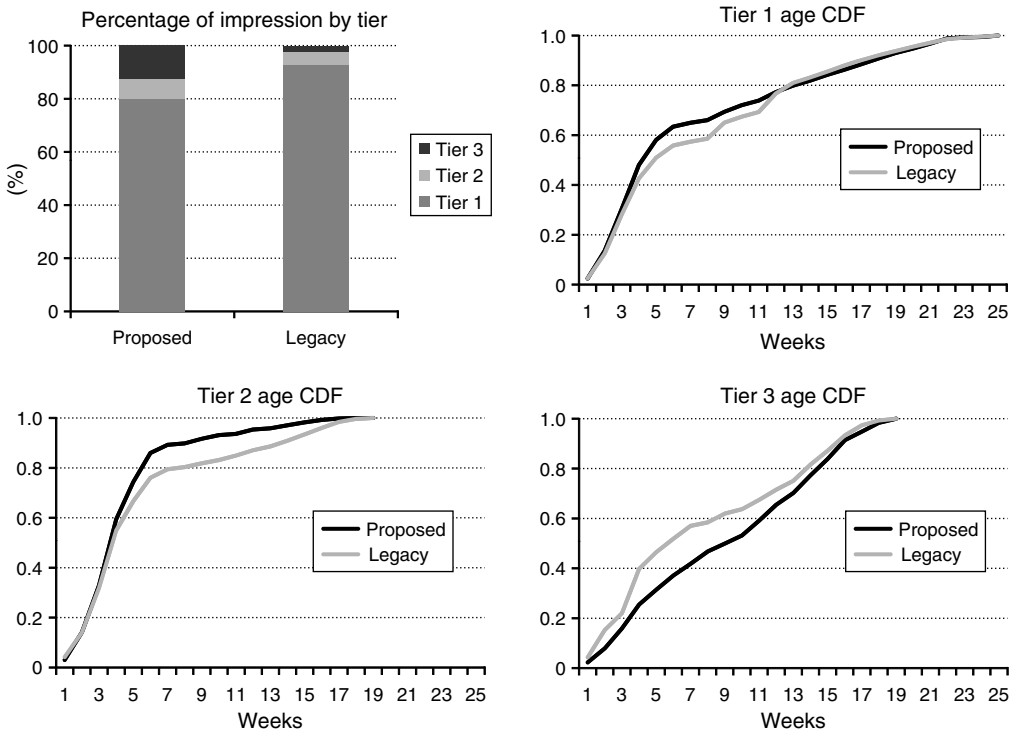
Figure 2. Performance of our algorithm compared to the legacy algorithm for different levels of forecast error α.



Note. The legacy algorithm does not use forecast data.

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Figure 3. The tier distribution of impressions served (top left) and age distributions of impressions served for each tier, for the legacy algorithm and our proposed algorithm (using $\alpha = 0.2$).



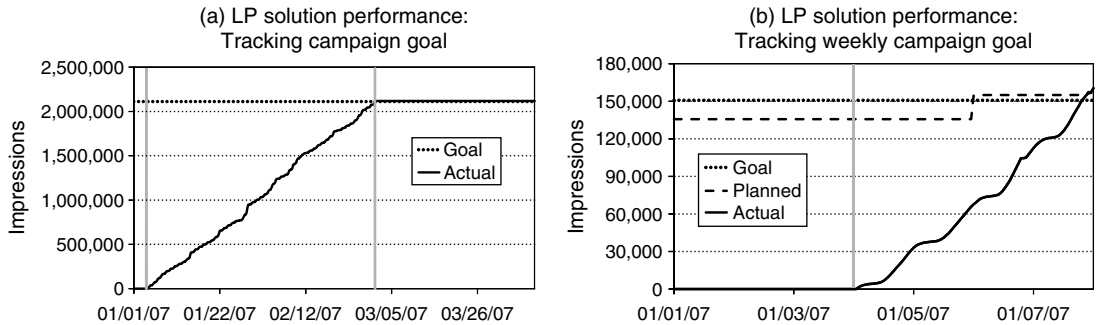
algorithm should try to serve tier 1 inventory as early as possible, and serve tier 2 and tier 3 inventory later on, again saving the most valuable inventory for future sales.

In Figure 3, we compare the age distribution from our algorithm with that of the legacy algorithm; in all cases, the forecast error parameter is set to $\alpha = 0.2$ (the most conservative scenario). As expected, our algorithm shifts a substantial portion of impressions from tier 1 into tiers 2 and 3. The legacy algorithm serves 92% of its impressions from tier 1 games, whereas our algorithm serves just under 80% from tier 1. Furthermore, our algorithm serves tier 1 and tier 2 impressions earlier in a campaign's life, whereas tier 3 impressions are served later: 63% of tier 1 impressions are 6 weeks old or younger using our algorithm, compared to 56% under the legacy algorithm.

6.2. Stability of Our Algorithm

Figure 4(a) shows that our algorithm makes steady, continuous progress toward meeting the end-of-campaign impression goals; the plot shows the cumulative number of impressions logged by a single campaign from its start date to its end date. Figure 4(b) plots the cumulative number of impressions logged by this same campaign in its first week. Notice that the LP's plan is revised midweek, at

Figure 4. The progress of the Real-Time Algorithm (labelled "actual") plotted over time for the largest campaign in the system.



Notes. The forecast error parameter is the most conservative ($\alpha = 0.2$). In (a) the goal is the end-of-campaign impression goal; in (b) we show the end-of-week impression goal and the number of impressions planned this week by the Weekly Plan LP.

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

which point the LP decides it is less costly to overrun the weekly goal a bit than to use inventory from future weeks to serve this campaign. Our algorithm successfully meets the end-of-week target, despite the fact that this campaign starts midweek.

6.3. Performance Under Higher Sell-Through and Tighter Targeting

We now evaluate our algorithm as the number of campaigns in the system increases and the supply constraints become tighter (or infeasible in some cases), thereby validating our approach as sales volume increases. We use *sell-through* and *targeting percentage* to measure the tightness of the supply constraints. As defined in §3, the mean and standard deviation of sell-through at the network level describe the tightness of supply: a high standard deviation of sell-through indicates that in the space of all impressions in the network, there are pockets of inventory with high sell-through, corresponding to intersections of games, DMAs, IECs, weekdays, and dayparts that are in high demand. The *targeting percentage* of campaign k, computed as $(\sum_{t\in T_k}\sum_{i\in I_k} s^t_{ik})/(\sum_{t\in T_k}\sum_{i\in\Upsilon} s^t_i)$, represents the percentage of impressions in the network that campaign k targets. A low targeting percentage (i.e., tight targeting) alone does not make a campaign difficult to serve, but when a tightly targeted campaign targets inventory that *also* has high sell-through, serving this campaign is difficult.

We generate six additional test cases with different levels of sell-through and targeting and benchmark the performance of our algorithm relative to the legacy algorithm. Each test case is built by augmenting the original problem instance from §6.1 (hereby called "status quo") with a set of 200 randomly generated campaigns. The test cases differ in two properties of the *added* campaigns: (1) the mean daily impression goal relative to the empirical distribution (which affects sell-through), and (2) the number of DMAs that match (which affects targeting). The other campaign parameters are consistent with the empirical distributions. The six test cases, which are listed in Table 3, have targeting percentages and coefficients of variation of sell-through in the same general range as the "status quo" test case, although some aspect of the instance (most often mean sell-through) is more highly constrained. (The precise values of these parameters for the status quo case are proprietary.)

Table 3. Test cases for tight supply constraint tests.

Instance	Impression goal multiplier	Number of matching DMAs	Mean sell-through	Standard deviation of sell-through	Targeting percentage
1	1.2	1	0.27	6.75	0.018
2	2.4	1	0.50	11.88	0.011
3	1.2	10	0.28	0.47	0.037
4	2.4	10	0.52	0.88	0.034
5	1.2	40	0.28	0.28	0.097
6	2.4	40	0.52	0.60	0.097

The test cases were constructed with mean sell-through at either (approximately) 27% or 52% and targeting percentages from 1% to 10%; as a result, there is a large range for the standard deviation of sell-through. We use $\alpha = 0.2$ in all test cases.

As can be seen by Figure 5(a), our algorithm outperforms the legacy algorithm in all of the six high-load test cases, confirming that our approach is robust. Relative outperformance, however, begins to decrease when the impression goal multiplier is high (i.e., mean sell-through is high) and the number of matching DMAs is low (i.e., targeting percentage is low). There is a nearly 60% make-good cost reduction for test case 5 (impression goal multiplier = 1.2, number of matching DMAs = 40), yet for test case 2 (impression goal multiplier = 2.4, number of matching DMAs = 1) the make-good cost reduction is only 5%. The relative performance of our algorithm diminishes because there are regions of the network in which the sell-through is above 100%—that is, demand exceeds supply, and therefore make-good costs are inevitable. We therefore compute an approximate lower bound on the make-good costs by solving the Weekly Plan with perfect forecasts ($\alpha = 0$) and only the make-good cost term in the objective. This is not a true lower bound because, as described in §3, we compute s^t_{ik} and $\hat{s}^t_{ik}$ using assumed independent adtime breakouts b^t_{id}, b^t_{ie}, and b^t_{iw}.

Using this bound (LB), we can measure the percentage of the gap that we close relative to the legacy algorithm, defined as 1 − (Our Algorithm Cost − LB)/(Legacy Algorithm Cost − LB). As seen in Figure 5(b), the percentage gap closed is 30% or better for most cases, dropping only when targeting gets very tight. This can be interpreted as a 30%+ cost savings on the part of the network that can be affected by good scheduling under reasonably high sell-through and tight targeting. In particular, if we order these test cases by the standard deviation of the sell-through as in Figure 5(c), we can see a clear relationship between relative outperformance and increasing variability of sell-through.

Even though the relative improvement is smaller, because the absolute magnitudes of the make-good costs in the highly constrained test cases are higher, a 5% cost savings for test case 2 translates into a *greater* absolute dollar amount saved when using our algorithm than the 60% cost savings for test case 5 (see Figure 5(d)). Thus, our algorithm significantly reduces make-good costs in all of the scenarios with high sell-through and tight targeting that we consider.

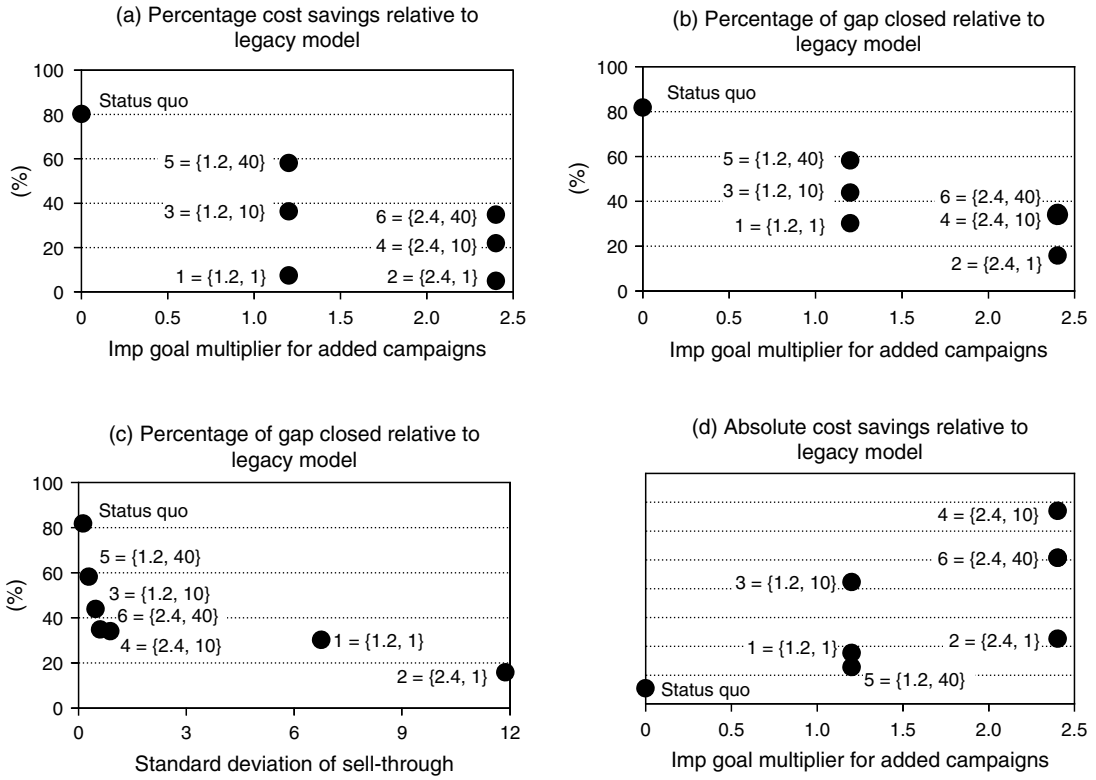
7. Implementation and Learnings

7.1. Implementation

Our collaboration has led to a paradigm shift at Massive: Previously, their ad-serving architecture was entirely execution based. After our project, Massive realized the benefits of hierarchical planning, where you first produce a plan and then you execute that plan. As of December 2009, a staged

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

Figure 5. Performance of all six high-load test cases + status quo.



Note. Subfigures are (a) top left, (b) top right, (c) bottom left, and (d) bottom right (vertical scale omitted for confidentiality).

implementation of our model has been underway. When complete, an LP-based planning module will be run nightly to produce *daily* targets for each campaign in each game, and an improved legacy algorithm that exploits structural ideas from our Real-Time Algorithm will track these daily targets. From a technology standpoint, Massive chose to use C# and the Microsoft Foundation Solver API.

Massive decided to switch from weekly to daily granularity because they discovered advertisers place a higher value on smooth delivery than previously thought, and Massive expects the benefits of explicit daily goals to outweigh the increase in forecast error caused by planning at a finer granularity. Moreover, due to our work, Massive knows that a granularity finer than {Campaign, Game, Day} is unnecessary. Recall from §6.3 that we tested how our model performs when many campaigns request tightly targeted inventory (i.e., Pittsburgh on Tuesday afternoon between 4–7 P.M.), and found that even though our Weekly Plan LP used the coarse granularity of {Campaign, Game, Week}, and did not explicitly resolve inventory conflicts for overcapacitated geographic regions and times of day, performance was quite good. We ascribe this to the fact that the schedule was being updated periodically, so shortfalls in a specific geographic region could be compensated for by increasing the overall service rate in that game.

In terms of the implementation's rollout, Massive decided the legacy algorithm should be adapted first, and the LP implemented afterward. Specifically, as a crucial intermediate step toward LP-based planning, Massive created a database to store impression goals for each {Campaign, Game, Day} and modified the legacy ad-serving logic to track these daily goals. Presently, a greedy heuristic computes the daily goals and populates this database; when the LP is complete, this database will instead be populated by the LP, and the real-time ad-serving logic will benefit from improved guidance without additional modification. This change to the legacy algorithm has been substantial: the explicit goals for each game have increased the *reach* of Massive's network; that is, by more effectively spreading impressions across many games, Massive has increased the number of unique individuals that see each campaign, on average by 26% per week.

Another substantial improvement made to Massive's legacy algorithm involves the use of adtime estimates for each inventory element. Originally, Massive's legacy algorithm did not use forecasts of any kind. As a result, the system tended to serve impressions erratically: the system would overserve, learn that overserving had occurred (because there is a significant delay in counting impressions), compensate by stopping the service of the campaign for awhile, and then repeat the process later as the campaign began to starve. Our Real-Time Algorithm does not suffer from this effect because it uses adtime estimates to track service rates that stay fixed between re-solves of the LP. To achieve smoother service rates in their existing framework, Massive adapted their legacy algorithm to

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.

use adtime estimates for each inventory element. In particular, Massive now assumes that when a campaign is placed into an inventory element, a number of impressions will be served in expectation. These "expected impressions" get accumulated, and are part of the service rate calculation that Massive uses to throttle the service rate of a campaign up or down. As a result, service rates are now much more stable: the standard deviation of hourly impressions served is 33% lower than before the use of adtime estimates.

In addition to the measurable benefits of increased reach and smoother campaign delivery, Massive has benefited from our collaboration in several intangible ways. First and foremost, the discovery process that we undertook with Massive to formulate the constraints of the linear program has been very valuable; in some cases, unwritten business rules were uncovered and communicated to business analysts and developers. The language of objectives, variables, and constraints has facilitated many engineering discussions and has been instrumental in building confidence in the new, more transparent, system. Furthermore, the visibility of sold inventory has been improved; sales associates can view the daily plan, allowing them to price games based on capacity and to encourage advertisers to purchase impressions from undersold games.

7.2. Learnings

Our learnings throughout the project led to several refinements of our model. For example, our Weekly Plan LP did not initially include constraints to guarantee a minimum amount of revenue per game. This posed a problem when a very popular game was added to Massive's network: the new game was allocated a sizable number of impressions that previously would have gone to incumbent games, thereby capturing their revenue. To be fair to all game publishers, we introduced minimum revenue targets for each game.

We also learned that Massive sells some campaigns called "share of voice" that contractually require a *percentage* of the total impressions generated by a game over a time period without having an absolute impression goal, e.g., an advertiser may request 25% of all impressions from Game A. This type of campaign is easy to serve in our framework; we just set the service rate to 25%.

Finally, we learned that the yield factor ϖ (see §3) used to convert between adtime and impressions varies from one game to the next, and also changes when the scheduling algorithm changes (i.e., from legacy to ours). We have recommended that yield factors be empirically measured over time in order to have accurate values of ϖ for each game.

With regard to hierarchical planning systems in general, we learned that it is usually not possible to have a complete separation of tasks from one stage to the next. For example, both the Weekly Plan LP and the Real-Time Algorithm deal with saturation: the Weekly Plan LP bounds service rates (constraint (8) in §4) so that with high likelihood the rates remain feasible when the Real-Time Algorithm enforces the saturation cap. This type of dependency between stages of a hierarchical planning system is common in other areas of practice as well; for example, in broadcast TV, commercials are spread over time (thus dealing with saturation) in both the high-level sales plan and by the lower-level ISCI Rotator Algorithm.

8. Conclusions

We studied the scheduling problem of Massive Inc., a network provider of dynamic in-game advertising and developed (1) an LP to compute service rates for each {Campaign, Game} pair, as well as (2) a Real-Time Algorithm that uses these service rates to assign campaigns to inventory elements whenever a gamer begins playing a new level. We benchmarked our algorithm against Massive's legacy algorithm and found that when tested on historical data (1) our algorithm reduces make-good costs by 80%–87%; (2) our algorithm reserves more impressions from premium games for future sales; (3) performance is good for various levels of forecast accuracy; and (4) results hold as sell-through increases and targeting becomes tighter, as is anticipated in the future.

Future work may include (1) explicitly modeling supply with random variables, (2) extending the model to plan at a finer granularity than {Campaign, Game, Week} using Bender's decomposition and/or column generation, (3) extending the model to incorporate reach goals by requiring a minimum number of unique game players to see a campaign, and (4) generalizing the model for other ad networks with dynamic display content, such as digital TV and the Web.

As a result of our work, Massive has increased the number of unique individuals that see each campaign by, on average, 26% per week, and achieved 33% smoother campaign delivery, as measured by standard deviation of hourly impressions served. Massive continues to improve their ad-serving technology based on the models and algorithms we developed and the insights we generated.

9. Electronic Companion

An electronic companion to this paper is available as part of the online version that can be found at http://or.journal.informs.org/.

Acknowledgments

The authors are grateful to Frank O'Donnell, David Sturman, Joe Ciaramitaro, and the rest of the team at Massive for numerous comments and support. This research was supported by Massive Incorporated and a William Larimer Mellon Fellowship.

References

Abrams, Z., O. Mendelevitch, J. Tomlin. 2007. Optimal delivery of sponsored search advertisements subject to budget constraints. *Proc. 8th ACM Conf. Electronic Commerce*, ACM, New York, 272–278.

Araman, V. F., I. Popescu. 2010. Media revenue management with audience uncertainty: Balancing upfront and spot market sales. *Manufacturing Service Oper. Management* **12**(2) 190–212.

Bollapragada, S., M. Garbiras. 2004. Scheduling commercials on broadcast television. *Oper. Res.* **52**(3) 337–345.

Bollapragada, S., M. R. Bussieck, S. Mallik. 2004. Scheduling commercial videotapes in broadcast television. *Oper. Res.* **52**(5) 679–689.

Bollapragada, S., H. Cheng, M. Phillips, M. Garbiras, M. Scholes, T. Gibbs, M. Humphreville. 2002. NBC's optimization systems increase its revenues and productivity. *Interfaces* **32**(1) 47–60.

Cai, Y. 2007. Electronic gaming in the digital home: Game advertising. Parks Associates Industrial Report, Parks Associates, Dallas.

Chambers, J. 2005. The sponsored avatar: Examining the present reality and future possibilities of advertising in digital games. Paper presented at DIGRA 2005, Simon Fraser University, Burnaby, British Columbia, Canada. Available at http://www.digra.org/dl/db/06278.01541.pdf.

Chickering, D. M., D. Heckerman. 2003. Targeted advertising on the Web with inventory management. *Interfaces* **33**(5) 71–77.

Gensch, D. H. 1973. *Advertising Planning: Mathematical Models in Advertising Media Planning*. Elsevier Scientific, Amsterdam.

Mehta, A., A. Saberi, U. Vazirani, V. Vazirani. 2007. AdWords and generalized online matching. *J. ACM* **54**(5) 22:1–22:19.

Nakamura, A., N. Abe. 2005. Improvements to the linear programming based scheduling of Web advertisements. *Electronic Commerce Res.* **5**(1) 75–98.

Raghavan, P., C. D. Tompson. 1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica* **7**(4) 365–374.

Rossiter, J. R., P. J. Danaher. 1998. *Advanced Media Planning*. Kluwer Academic Publishers, Norwell, MA.

Simon, H. 1982. ADPULS: An advertising model with wearout and pulsation. *J. Marketing Res.* **19**(3) 352–363.

Svahn, M. 2005. Future-proofing advergaming: A systematisation for the media buyer. *Proc. Second Australasian Conf. Interactive Entertainment*, Creativity & Cognition Studios Press, Sydney, Australia, 187–191.

Talluri, K. T., G. van Ryzin. 2005. *The Theory and Practice of Revenue Management*. Springer Verlag, New York.

Thompson, G. 1981. An optimal control model of advertising pulsations and wearout. J. W. Keon, ed. *Marketing Measurement and Analysis*. The Institute of Management Sciences, Providence, RI.

Zhang, X. 2006. Mathematical models for the television advertising allocation problem. *Internat. J. Oper. Res.* **1**(3) 302–322.

INFORMS holds **copyright** to this article and distributed this copy as a courtesy to the author(s).
Additional information, including rights and permission policies, is available at http://journals.informs.org/.