Automated Channel Abstraction for Advertising Auctions[*] [†]

William E. Walsh
CombineNet, Inc.
Fifteen 27th St.
Pittsburgh, PA, USA
wwalsh@combinenet.com

Craig Boutilier
Dept. of Computer Science
University of Toronto
Toronto, ON, Canada
cebly@cs.toronto.edu

Tuomas Sandholm
Computer Science Dept.
Carnegie Mellon University
Pittsburgh, PA, USA
sandholm@cs.cmu.edu

Rob Shields
CombineNet, Inc.
Fifteen 27th St.
Pittsburgh, PA, USA
shields@combinenet.com

George Nemhauser
ISYE
Georgia Inst. of Technology
Atlanta, GA, USA
gnemhaus@isye.gatech.edu

David C. Parkes
SEAS
Harvard University
Cambridge, MA, USA
parkes@eecs.harvard.edu

ABSTRACT

The use of auction mechanisms like the GSP in online advertising can lead to loss of both efficiency and revenue when advertisers have rich preferences: even simple forms of expressiveness like budget constraints can lead to suboptimal outcomes. This has led to the recognition of the value of (sequential and/or stochastic) optimization in ad allocation. Unfortunately, natural formulations of such optimization problems fall prey to *channel explosion*. Specifically, available ad inventory must be partitioned into subsets, or *channels*, of *indistinguishable supply*, each channel containing inventory that is interchangeable from the perspective of *each* active advertiser. The number of such channels grows exponentially in the number of features of interest. We propose a means for automatically abstracting these channels, grouping together channels so that irrelevant distinctions are ignored. Our approach, based on LP/MIP column and constraint generation, dramatically reduces the number of distinct channels over which ads are allocated, thus rendering optimization computationally feasible at practical scales. Our algorithms also allow revenue/efficiency to be sacrificed in a principled fashion by ignoring potentially relevant distinctions, but retaining the most important distinctions, ignoring only those that have low impact on solution quality. This allows tradeoffs to be made between tractability and solution quality. Numerical experiments demonstrate the computational practicality of our approach as well as the quality of the abstractions generated.

1. INTRODUCTION

Online advertising has radically changed both the nature of advertising and the technology used to support the development and deployment of ad campaigns. While ad targeting and campaign design is inherently complex, the variety of online advertising services has only increased this complexity. In particular, the ability to target ads to *specific individuals* based on detailed, personalized online information—information that is simply not available in broadcast media—presents compelling opportunities and tremendous technical challenges for ad delivery. For instance, the development of sophisticated matching and bidding algorithms for sponsored search, such as position auctions using the *generalized second price (GSP)* mechanism, can be viewed as a response to such opportunities [8, 17].

In contrast to sponsored search, the selling of *banner ads* (aka. *display ads*) is still largely approached through manual negotiation. There are some exceptions to this, with online exchanges for banner ads established by companies like Right Media (now part of Yahoo!) and DoubleClick (now part of Google); however, these exchanges largely deal with lower-value, “remnant” inventory on web sites. Premium display advertising space (e.g., slots near the top, or “above the fold,” of high traffic, high profile websites) is sold almost exclusively by non-automated means. The primary reason for this is a perception that auction/market mechanisms cannot be made to work for the types of campaign-level expressiveness required for display ads (e.g., as required by brand advertisers).[1]

Campaign-level expressiveness is addressed explicitly in [14, 6], where a variety of expressiveness forms are outlined (these include impression targets, smoothness of delivery, temporal sequencing, complements and substitutes, and many others). Although sophisticated bidding strategies [5, 9, 15] for some limited forms of expressive preferences (e.g., long-term budgets) can help extract greater value from an inexpressive auction, arbitrarily large inefficiencies can nevertheless arise [3]. Allowing richer languages in which advertisers can express their campaign preferences directly, rather than forcing them into standard per-event bidding models, is critical to admitting the automated matching and selling of banner ads. But a key bottleneck remains: the use of expressive bidding re-

[*]This work was sponsored by, and conducted at, CombineNet, Inc. Patent pending.

[†]Extended version of a paper that appeared in the *Fifth Workshop on Ad Auctions*, 2009.

[1]This parallels the situation in sourcing, where advances in modeling and optimization have led to the adoption of expressive bidding (and expressive bid-taking) for what had previously been widely viewed as “too valuable” to leave to auction mechanisms [16]. The expressive auction mechanisms are now used also for striking strategic long-term contracts on the most valuable parts of the sourcing spend.

.

quires *optimization* to match ad supply with advertisers' demand. The richer the expressiveness forms, the more complex the optimization. For example, in [6], a stochastic optimization model for rich, campaign-level expressiveness forms. However, even with very limited forms of expressiveness—as simple as per-impression value/pricing with budget constraints and bid expiration—that optimization is critical to extracting full value from one's ad inventory [14, 1]. Indeed, using simple myopic mechanisms like GSP can lead to significant loss in efficiency and revenue.

In this paper, we tackle one of the greatest impediments to the use of optimization in ad auctions, namely, that of *channel explosion*. A key advantage advertisers have in online settings is the ability to segment the target audience using an enormous variety of features: both static features (like user demographic) and dynamic features such as context (e.g., current browsing history, location) or historical data (e.g., past purchases, activity, etc.). This means that the number of features over which ad allocation must occur is extremely large. And the number of specific *ad channels* to which ads can be assigned—i.e., the number of distinct feature instantiations—grows exponentially in the number of features. Any optimization model must (usually quite explicitly) assign advertisers to explicit channels over time. Both simple linear programming (LP) models that use only budget constraints [1] and sophisticated mixed-integer (MIP) models [6, 14] use variables of the form x^i_j to denote the allocation of some amount of the supply of channel j to advertiser i.[2] These models simply cannot scale directly to problems involving more than a few thousand channels (e.g., on the order of 10-15 (Boolean) channel features).

We address the channel explosion problem through the use of *channel abstraction*. Intuitively, an abstract channel is any aggregation of these "concrete" channels (i.e., feature instantiations) into a single abstract channel for the purposes of optimization. During allocation optimization, ads are assigned to abstract channels rather than concrete channels;[3] hence with appropriate abstraction, we can obtain exponential reduction in the number of channels, thus rendering optimization practical. Furthermore, a well-chosen abstraction will often provide very little sacrifice of revenue or efficiency (often even providing an optimal, lossless abstraction). Such abstractions should be derived by considering their impact on *value* (e.g., efficiency or revenue) as opposed to clustering based on purely, say, statistical properties of the features in question.

We propose a suite of techniques for automatically generating abstractions and for optimization using a set of abstract channels. Our first algorithm uses a form of *column generation* to generate an abstraction: starting with a crude abstraction, we gradually refine the abstraction by introducing distinctions that have maximal impact on objective value. Unlike standard column generation, we must determine which *collection* of columns to add (and remove). We develop novel scoring techniques to do just this. We also develop a new *constraint generation* algorithm for optimizing an ad allocation MIP using a specific set of abstract channels (e.g., those generated by our column generation algorithm). This method incrementally refines the allocation of bids to abstract channels by posting constraints to ensure advertisers are assigned only relevant ad slots. This method will converge, in principle, to an optimal solution given enough time. However, we also discuss how the technique can be cut short with an approximate solution, and how it can be used to suggest further channel refinement for purposes of tractability.

[2]For example, in [1], each distinct keyword/query is a channel; and bids (or more precisely, slates of bids) are allocated to each query. Tractability is achieved by focusing on only the few thousand highest-volume queries. The MIP model of [6] uses assignment variables for losslessly "abstracted" channels consisting of (bid,attribute)-intersections, and is limited to a relatively small number of channels.

[3]As we discuss below, *dispatch* of ads assigned to an abstract channel will generally be sensitive to the actual channel, or full feature instantiation, in question.

The remainder of the paper is organized as follows. We briefly discuss the need for campaign-level expressiveness, optimization, and channel abstraction in Sec. 2. We present the basic ad allocation model in Sec. 3 and define our notion of abstract channels precisely in Sec. 4, along with its impact on optimization. Sec. 5 develops a novel and computationally effective column generation technique to generate useful abstractions, and provides empirical results demonstrating that near-optimal allocations can be determined using very few channels. We extend the approach in Sec. 6 with an iterative constraint generation algorithm to allocate bids to abstract channels that is sensitive to distinctions that are abstracted away. Empirical results demonstrate significant improvement in value when "IP expressiveness" (i.e., requiring binary variables) is involved. Sec. 7 addresses possible refinements of our techniques and key issues in implementation and deployment, such as data representation and uncertainty in supply. We conclude with suggested directions for future research in Sec. 9.

2. EXPRESSIVE ADVERTISING AND OPTIMIZATION

We consider the problem faced by an ad network selling and serving banner ads over a variety of web sites. Ads are served (*dispatched*) to specific locations on web pages as the pages are served by members of the network. Dispatch decisions can be based on a variety of *features* or impression attributes: features of the web page (e.g., page identity, page category, predicted demographic profile of users, page content, etc.), features of the user, if available (e.g., demographic properties such as gender, income level, geographic location), and transient contextual features (e.g., day-part, browsing history, past purchases, etc.).

In typical ad auctions, advertisers bid for ad slots satisfying specific features. Advertisers that match the features of the current ad slot are often allocated using GSP (more so for sponsored search that banner ads). Expressiveness is typically non-sequential and restricted to *per-item bidding* (e.g., a bid price is offered per-impression or per-click), time eligibility conditions, and simple budget constraints (often linking multiple bids/items, e.g., [13]).[4]

Even in such a simple setting, the need for optimization can be acute. Consider the following example, adapted from [6]:

> There are two sites A and B. Bidder b_1 bids \$1 per thousand impressions on A and \$0.50 on B, with a budget of \$55K. Bidder b_2 bids \$0.50 per thousand impressions on A, with a budget of \$45K. Suppose supply on A is 5 times that of B for the first 50K units, but is then exhausted (only B has supply from then on). In a typical per-item auction, b_1 will win all of A's and B's supply until its budget is exhausted. Specifically, b_1 would win 50K impressions of A and b_2 would win nothing. Total revenue is \$55K. The optimal allocation would collect revenue of \$100K by selling 50K units of A to b_2 and 110K units of B to b_1.

Optimization is also critical when one considers *slates* of ads (multiple advertisers shown on a single web page) [1].

[4]Structured (tree-based) languages have been proposed for specifying item prices over ad features [11]. These do not extend expressiveness beyond per-item, but allow compact, natural specification of a set of item prices that can be exploited in optimization.

The need for richer expressiveness in ad auctions is evident, especially campaign-level expressiveness for banner ads.[5] This point was emphasized in [14], where various forms of expressiveness are described, along with an *optimize-and-dispatch* architecture in which: (a) optimization is used to allocate ads over time at a coarse level of time granularity; and (b) a *dispatcher* assigns ads in real time to specific page impressions using parameters determined by the optimizer. Further forms on campaign-level expressiveness are detailed in [6], where algorithms for the online, approximate solution of the *Markov decision process* induced by the allocation model are developed. Specifically, given uncertain supply (in the form of web page hits) and demand (in the form of bids or contracts), the approach optimizes the allocation of (long-term) *expressive ad contracts* to *ad channels* (that is, groups of features satisfying specific properties) based on the distribution of predicted supply. Indeed, inexpressiveness can lead to arbitrary inefficiency in GSP for certain distributions of agent preferences (even with per-item preferences) [3].

In what follows, we assume that advertisers make *expressive offers* that articulate their preferences for *sequences* or *sets* of impressions (or clicks, conversions, etc.). These can include per-item bids, budgets, and other standard forms, but are extended to include much richer offer terms. We enumerate just a few examples of expressiveness that illustrate the power of our model:

- **Minimum targets/threshold preferences**: bidder pays a fixed amount only if a minimum impression threshold is met during a target period (e.g., $\$d$ for 300K impressions satisfying some condition φ). Multiple targets may be mixed, as may per-impression bids with *bonuses* for achieving specific targets. Maximums, even frequency capping at the site or individual level, can be imposed as well.
- **Temporal sequencing/smoothness**: bidder desires a minimum number of impressions satisfying condition φ in each of a set of time periods (e.g., 200K impressions per day for two weeks); or the bidder may make a threshold or per-impression offer that is only "valid" if the variance in the number of impressions per time period is no more than 10% (here validity may mean that the impressions outside that range are not counted, or that the entire contract is invalid).
- **Complements**: ads on site A and site B must appear in a 2:1 ratio (either over the life of the campaign, or during each relevant time period, e.g., hour, day-part, day, week).

While per-item expressiveness and budgets can usually be incorporated directly into an LP model [1], some of these richer forms of expressiveness require the introduction of binary variables (e.g., threshold preferences). Such MIP formulations are explicitly solved in [6]. However, existing LP/MIP models are unable to scale to practical problems involving a large number of *features* (impression attributes); yet it is precisely the ability to segment on very detailed attributes that explains the appeal of online advertising! The key bottleneck is the *channel explosion*: the number of specific *ad channels* to which ads can be assigned in an LP/MIP—i.e., the number of distinct feature instantiations—grows exponentially in the number of features, a problem to which we now turn.

[5]A similar need for campaign-level expressiveness is clear in other media as well, e.g., in TV advertising [4]. Our techniques apply directly to such problems, though the channel explosion problem is somewhat mitigated by the inability (or at least, current unwillingness) to segment individual impressions by very fine-grained features. The technology and willingness to monitor viewing habits and individual details to target ads as they are online would make the problem addressed here as acute in such settings.

3. ALLOCATION MODEL

We first outline a generic model for display ad allocation. A number of factors, such as the observability of impression features, stochasticity of supply, and data representation are set aside (but see Sec. 7). For now, we assume the ability to tractably reason with arbitrary logical formulae over multi-valued features and joint distributions over such features.

We assume a finite set of attributes or *features* $\mathcal{F}$, with each $F^i \in \mathcal{F}$ having finite domain $Dom(F^i) = \{f^i_1, f^i_2, \ldots, f^i_{n^i}\}$. Features describe attributes of an ad display such as web site, page location, user demographic, day part, etc. Each feature, possibly depending on the web property, is either *observable*—an ad display is known to satisfy that feature with certainty or not—or *stochastically verifiable*—an ad can be determined to satisfy that feature only with some specified probability.[6] To reduce notational clutter, we assume all features are observable (but see Sec. 7). Ad displays occur over some finite set of time periods $\{1, \ldots, T\}$.

Define the set of *concrete channels (c-channels)* $C = Dom(\mathcal{F})$ to be the instantiations or "possible worlds" over features $\mathcal{F}$. Intuitively, a c-channel $c \in C$ is a finest-grained chunk of supply to which an ad can be assigned. We often treat c as a model of the propositional language over variables $\mathcal{F}$ (e.g., writing $c \models \varphi$ for propositional formulae φ over $\mathcal{F}$). Let $s(c,t)$ denote the supply of c-channel c available at time $t \leq T$. We take supply to be deterministic (uncertain is addressed in Sec. 7.)

Potential advertisers have particular campaign objectives in mind, which will be expressed using a set of one or more bids, with bids potentially linked by shared variables, constraints, etc., reflecting the forms of expressiveness discussed above. While we allow all types of expressiveness that can be expressed as a MIP, some of our techniques below can be motivated by considering very simple bid structures, embodying "LP expressiveness" only. We present this special case here. Assume a bid set $\mathcal{B}$ consisting of a set of item-based, budget-constrained bids. Each bid $i \in \mathcal{B}$ has the form $\langle \varphi^i, v^i, g^i, w^i \rangle$, where φ^i is an arbitrary logical formula over the features $\mathcal{F}$, $v^i > 0$ is i's value/price per impression, $g^i > 0$ is its budget, and w^i is a time window $[s^i, e^i]$ with a start and end period between which impressions must occur ($1 \leq s^i \leq e^i \leq T$). Bid i reflects advertiser i's interest in impressions satisfying the condition φ^i. The (deterministic) allocation problem in this setting can be formulated as a simple LP that maximizes revenue by allocating $x^i_j(t)$ impressions of c-channel $c_j \in C$ to bid i at time t. To simplify notation, we formulate the optimization as if there were a single time period (the generalization to multiple periods is obvious). Let v^i_j be i's value for a c_j-impression: $v^i_j = v^i$ if $c_j \models \varphi^i$; $v^i_j = 0$ otherwise. Then we have:

$$\begin{array}{lll} \max\limits_{x^i_j} & \sum\limits_i \sum\limits_j v^i_j x^i_j & \\ \text{s.t.} & \sum\limits_i x^i_j \leq s(c_j) & \forall c_j \in C \\ & \sum\limits_j v^i_j x^i_j \leq g^i & \forall i \in \mathcal{B} \end{array}$$

Other forms of campaign expressiveness can easily be incorporated into this LP. For example, if a campaign has (partially) substitutable demands (e.g., it desires φ_1 or φ_2 with values v_1

[6]E.g., impressions to registered readers of the New York Times may be observable with respect to gender, while the gender of non-registered readers might only be probabilistically predicted given statistical data. Certain features may also be *inapplicable* under certain conditions, for example, on certain sites.

and v_2), two separate bids can be posted with a joint budget constraint. If φ_1 and φ_2 are complements, we can constrain the allocated impressions to meet some approximate ratio target (e.g, $cnt(\varphi_1) \leq (1+\varepsilon)cnt(\varphi_2)$, $cnt(\varphi_2) \leq (1+\varepsilon)cnt(\varphi_1)$, where $cnt(\varphi)$ is the number of impressions of φ). Smoothness constraints can also be encoded linearly (e.g., requiring at least 10% of total impressions to be allocated in each eligible time period). We refer to these and any other expressiveness forms that can be encoded in the LP as *LP expressiveness*. Notions such as threshold/bonus bids cannot be expressed in an LP, requiring the introduction of binary variables [14, 6]: we refer to these forms as *IP expressiveness*.

4. ABSTRACT CHANNELS

The number of c-channels $|C|$ grows exponentially in the number of features of interest. This number can be pruned by eliminating any features that interest no bidder. We can also provide a tighter bound on the number of required channels by aggregating c-channels that are indistinguishable to every bidder; this provides a *simple lossless abstraction* by grouping sets of c-channels corresponding to (logically consistent) formulae of the form $\wedge_{i\in\mathcal{B}} \pm \varphi^i$; i.e., conjunctions over all bid formulae or their negations.

However, such simple lossless abstraction is unlikely to render optimization (whether LP or MIP) practical: we still expect exponential growth in the number of channels, even when abstracted in this way. Instead, we must consider the use of "approximate" *abstract channels (a-channels)*. An abstract channel is any aggregation of c-channels, and can be represented as a logical formula α over $\mathcal{F}$. An *abstraction* is a partitioning of c-channels C into a set A of a-channels, i.e., a set of mutually exclusive and covering formulae $\{\alpha_1, \ldots, \alpha_{|A|}\}$. We treat an a-channel and its logical representation α indistinguishably, writing both $c \in \alpha$ and $c \models \alpha$ as appropriate.

Given an abstraction A, our optimization problem becomes one of assigning ads/bids to *a-channels* rather than c-channels. Define the supply of a-channel α to be $s(\alpha) = \sum\{s(c) : c \in C, c \models \alpha\}$. In the LP case with per-impression value, define the value of an α-impression to bid i:

$$v^i_\alpha = v^i \Pr(\varphi^i|\alpha), \quad \text{where} \quad \Pr(\varphi^i|\alpha) = s(\varphi^i \wedge \alpha)/s(\alpha).$$

This value reflects the (expected) value of a *random dispatch policy*: if i is assigned to an abstract channel α, it will be assigned randomly to the c-channels that constitute α.[7] The optimal allocation under the random dispatch assumption is given by the LP:

$$\begin{array}{lll} \max\limits_{x^i_{\alpha_j}} & \sum\limits_i \sum\limits_{\alpha_j} v^i_{\alpha_j} x^i_{\alpha_j} & \\ \text{s.t.} & \sum\limits_i x^i_{\alpha_j} \leq s(\alpha_j) & \forall \alpha_j \in C \\ & \sum\limits_j v^i_{\alpha_j} x^i_{\alpha_j} \leq g^i & \forall i \in \mathcal{B} \end{array}$$

With more general IP expressiveness, we do not associate value directly with impressions, but with properties of the entire allocation; specific impressions satisfying logical formulae φ^i "count towards" satisfaction of a bid's conditions. Thus we generally discount the *impressions that count* toward bid satisfaction by $\Pr(\varphi^i|\alpha)$ rather than discounting objective function value. The value discount in the per-impression LP is a special case of this.

[7]The dispatch of ads can be handled more intelligently: no ad for i will actually be assigned to a channel not satisfying φ^i; intelligent dispatch [14] can be used to reassign such wasted supply to ads that can exploit it. Thus, v^i_α will underestimate true value. We discuss this further below, and we develop methods to assign ads to abstract channels in a more refined fashion.

5. CREATING ABSTRACTIONS: COLUMN GENERATION

The solution of the abstract LP or MIP provides us with an optimal assignment of bids to a set of a-channels. This leaves the question of choosing a suitable set of a-channels: a set of computationally-manageable size, yet whose optimal solution provides an optimal or near-optimal solution to the original unabstracted MIP. Our first technique relies on column generation, and deals directly with LP expressiveness. We first describe the method using problems with only supply constraints, but then show how it applies more broadly to include arbitrary linear constraints (including budget constraints). We then show how to account for IP expressiveness.

The basic approach is as follows: we solve an abstract LP using some initial level of abstraction (e.g., aggregating all c-channels into a single a-channel $\top$). We refine the abstraction heuristically by choosing an abstract channel α to split into two by conjoining a formula β and its negation, thus replacing α by $\alpha \wedge \beta$ and $\alpha \wedge \overline{\beta}$. A new LP is solved with the new a-channels, and the process repeats until the improvement in LP objective value falls below some threshold or the number of channels reaches a specified limit.

Consider the following LP to allocate bids $\mathcal{B} = \{1, 2\}$ to a single abstract channel α (with no budget or other constraints):[8]

$$\begin{array}{llll} \text{Max} & v^1_\alpha x^1_\alpha & +v^2_\alpha x^2_\alpha & \\ \text{s.t.} & x^1_\alpha & +x^2_\alpha & \leq s(\alpha) \end{array}$$

Refining a-channel α requires introducing the bid columns (and supply rows) corresponding to $\alpha \wedge \beta, \alpha \wedge \overline{\beta}$ for some β.

Column generation [12] is used to solve LPs with very large numbers of variables by first solving a version of the LP with very few variables (columns), then adding new variables into the LP at each iteration and resolving. At each iteration, the new columns are chosen by solving a *pricing subproblem* which identifies columns that potentially improve the objective. We adopt this approach here, but with some significant enhancements that exploit the special structure of our problem, and account the introduction of multiple columns at once ($x^i_{\alpha\wedge\beta}$ and $x^i_{\alpha\wedge\overline{\beta}}$ for each bid i in the example) while simultaneously removing other columns (x^i_α).

5.1 Scoring Abstract Channel Splits

Assume we have the solution of the abstract LP above. We first determine the value, or *score*, of a potential split of α into two a-channels $\alpha \wedge \beta, \alpha \wedge \overline{\beta}$. This score allows us to compare candidate splits defined by different β. We score a split by: (a) scoring the new columns introduced by the split using a form of column generation scoring; and (b) combining the scores of these new columns in a way that exploits the special structure of our problem.

Standard column generation methods solve a pricing subproblem to identify columns absent from an LP with positive *reduced cost*, and typically add a column with maximum reduced cost (for maximization problems), terminating when no reduced costs are positive. We apply a similar technique. Let π_α be the value of the dual variable corresponding to the supply constraint for a-channel α in the dual of the abstract LP (i.e., the shadow price of the constraint). The reduced cost of variable $x^i_{\alpha\wedge\beta}$ is:

$$rc(x^i_{\alpha\wedge\beta}) = v^i_{\alpha\wedge\beta} - \mathbf{c}\pi$$

[8]We illustrate with a single channel to reduce notational clutter. Unless $\alpha \equiv \top$, this LP will have a set of a-channels α_j and allocation variables x^i_j for each bid i and a-channel α_j.

where $\mathbf{c}$ is $x^i_{\varphi_j \wedge \beta}$'s column and π is the vector of dual variables. The reduced cost of $x^i_{\alpha \wedge \overline{\beta}}$ is defined similarly. Unfortunately, the abstract LP does not include relevant supply constraints for $\alpha \wedge \beta$ or $\alpha \wedge \overline{\beta}$, meaning shadow prices for these constraints cannot be directly obtained from the LP. We consider adding two new rows to the original abstract LP, reflecting split channel supply, as follows:

$$\begin{array}{llll} \text{Max} & v^1_\alpha x^1_\alpha & +v^2_\alpha x^2_\alpha & \\ \text{s.t.} & x^1_\alpha & +x^2_\alpha & \le s(\alpha) \\ & \Pr(\beta|\alpha)x^1_\alpha & +\Pr(\beta|\alpha)x^2_\alpha & \le s(\alpha \wedge \beta) \\ & \Pr(\overline{\beta}|\alpha)x^1_\alpha & +\Pr(\overline{\beta}|\alpha)x^2_\alpha & \le s(\alpha \wedge \overline{\beta}) \end{array}$$

Since $s(\alpha \wedge \beta) = \Pr(\beta|\alpha)s(\alpha)$ (similarly for $\overline{\beta}$), these new constraints are multiples of the original $s(\alpha)$ constraint, leaving the optimal solution unaffected. This allows us to price the two new constraints: when we consider the dual of this LP, one optimal solution sets the dual variable π_α to its value in the original abstract dual LP, and sets the two new dual variables $\pi_{\alpha \wedge \beta} = \pi_{\alpha \wedge \overline{\beta}} = 0$. As a result, we can compute the reduced costs of the variables corresponding to the split channels using terms available from the solution of the original abstract LP:

$$rc(x^i_{\alpha \wedge \beta}) = v^i_{\alpha \wedge \beta} - \mathbf{c}\pi = v^i_{\alpha \wedge \beta} - \pi_\alpha$$
$$rc(x^i_{\alpha \wedge \overline{\beta}}) = v^i_{\alpha \wedge \overline{\beta}} - \mathbf{c}\pi = v^i_{\alpha \wedge \overline{\beta}} - \pi_\alpha$$

Reduced cost measures the increase in objective value per unit increase in the (nonbasic) variable, making maximum reduced cost a common, easily computable *heuristic* for variable introduction. (It can also be used to prove optimality when max reduced cost is nonpositive.) However, it can be misleading since it fails to consider how far the target variable can be moved until constraints are met. Furthermore, our aim is to introduce a *set* of new columns (all bid variables for the two new channels created by the split), and *remove* a set of columns (those corresponding to the original channel).

In our simple case, with only supply constraints, we can measure *exactly* the change in objective value resulting from a split. Without budget constraints, all supply of the new split channel $\alpha \wedge \beta$ will be allocated to the bid i that has maximum value $v^i_{\alpha \wedge \beta}$, giving total objective value improvement of $rc(x^i_{\alpha \wedge \beta})s(\alpha \wedge \beta)$. Here the reduced cost component reflects the precise difference in objective value if an α-impression to a current winning bid is replaced by an $\alpha \wedge \beta$-impression to bid i, while the supply component tells us exactly how much substitution is available. Applying the same argument to $\alpha \wedge \overline{\beta}$ gives us the following measure for scoring the split of any channel α into two subchannels $\alpha \wedge \beta$ and $\alpha \wedge \overline{\beta}$:

$$score(\alpha, \beta, \overline{\beta}) = \max_{i \in \mathcal{B}} \{rc(x^i_{\alpha \wedge \beta})s(\alpha \wedge \beta)\} + \max_{i \in \mathcal{B}} \{rc(x^i_{\alpha \wedge \overline{\beta}})s(\alpha \wedge \overline{\beta})\}$$

This scoring function has the desirable property that the score of a split is *exactly* the induced improvement in objective value when the only constraints are supply constraints. Of course, almost all natural problems will have other constraints: budget constraints most certainly, and other expressive forms as well. However, if we limit ourselves to LP expressiveness, the reduced cost calculation remains straightforward, requiring one vector product (using dual/shadow prices computed in the LP solution). The scoring function itself becomes heuristic—though it still provides a guarantee of optimality if the maximum score is nonpositive. It provides an upper bound on the possible improvement in objective value (e.g., consider the case where the maximizing bid i for split $\alpha \wedge \beta$ has a budget constraint that prevents it from consuming the entire split supply). Despite this, it provides much better performance than using reduced costs alone. One could envision more complex scoring functions that attempt to solve small optimization problems to better estimate the improvement in objective value for a given split.[9] However, a key advantage is that our scoring function requires no additional computation over standard reduced cost calculations (using terms readily available from the LP solve) apart from a trivial maximization. This is critical, since the number of potential splits is doubly exponential: we discuss this next.

5.2 Searching for Suitable Splits

Scoring a split is computationally simple, requiring at most $2|\mathcal{B}|$ reduced cost calculations.[10] However, the number of potential splits of an a-channel α is doubly exponential in n (i.e., 2^{k^n} formulae over n features with domain size k). In addition, we need to evaluate splits of each a-channel α_j in the current abstraction A.

To manage the complexity of this search, we adopt a simple myopic approach to determining the best split of an a-channel α_j. We build up the formula β_j on which α_j is split as follows. Denote $Dom(F^i) \setminus \{f^i_k\}$ as $\overline{f^i_k}$, i.e., the exclusion of the value k for attribute i. We first consider each β^1_j consisting of $\overline{f^i_k}$ for a single i and k. That is, at the first "level" we consider splits that exclude one attribute-value. We "commit" to a single attribute-value exclusion with the best score $score(\alpha_j, \beta^1_j, \overline{\beta}^1_j)$. We then consider refining β^1_j by conjoining with some new $\overline{f^i_k}$ or disjoining with some new f^i_k (conjoining tightens β^1_j, disjoining relaxes it). Each resulting β^2_j is scored in a similar fashion, and we again commit to the β^2_j with the highest score. This continues for m iterations, where m is either a fixed threshold or is determined dynamically by requiring a minimum score improvement be met. The best split of α_j is determined heuristically as $\langle \beta_j, \overline{\beta}_j \rangle$, where $\beta_j = \beta^m_j$.

Given a current abstraction A, the $\alpha_j \in A$ with the highest-scoring best split is adopted, creating a new abstraction A' with α_j replaced by $\alpha_j \wedge \beta_j$ and $\alpha_j \wedge \overline{\beta}_j$. The LP resulting from the new abstraction is solved, and the search for a best split repeated until the score of the best split of A falls below some threshold τ.

5.3 Using Abstractions in Ad Auction Optimization

One limitation of the column generation model as proposed is its focus on LP expressiveness. However, recall that the abstraction process is used to create the set of abstract channels to be *used* in MIP optimization; i.e., the intended output of this process is a set of a-channels, not (necessarily) the allocation itself. Given an allocation problem with IP expressiveness, we use column generation with a linear relaxation of the problem to generate abstract channels. Once the abstract channels are constructed, we then solve the "original" MIP using allocation to the abstract channels created, with appropriate discounting of impression values or count variables by the probability of a bid receiving a *relevant* impression within an a-channel (see Sec. 4).[11] To evaluate this approach, we experimented the column generation model on a collection of random problems, some with LP expressiveness only, others with IP expressiveness. All experiments were run on a machine with a 3.8GHz Xeon CPU, 2BM cache, and 16GB RAM.

5.3.1 *LP Expressiveness*

[9]Folklore in column generation suggests this is rarely worthwhile.

[10]This is in fact an overestimate, since any bid i that cannot use abstract channel α (i.e., $\alpha \models \neg\varphi^i$) will not have a variable x^i_j and will not contribute to the score.

[11]If the original problem uses only LP expressiveness, then the LP solution used to create the final refinement will be the optimal allocation and no re-solve is needed.

The first battery of problems involves bids that use only LP expressiveness; specifically, each bid has per-impression valuations for a particular set of attribute-values over a given time period, along with a total budget. Optimizations are performed over a time horizon of 30 periods. This battery contains multiple sets of problem instances, each set characterized by two parameters: m binary attributes and n bidders. We ran sets of instances with $n = 10m$ for $m \in \{10, 20, 30, \ldots, 100\}$.

Supply distribution. The probability of a unit of supply satisfying attribute-value f_1^i is drawn from $U[0, 1]$: since $Dom(F^i) = \{f_1^i, f_2^i\}$, $\Pr(f_2^i) = 1 - \Pr(f_1^i)$. Total supply of impressions, over all attribute-values, is 1,000,000 for each time period.

Bids. Each bid j has form $\langle \varphi^j, v^j, g^j, w^j \rangle$ and cares about a set of attributes A^j with size $|A^j| \sim U[0, 10]$. We assume bidders tend to have a lot of commonality w.r.t. the attributes they care about, so bid attributes are sampled from a Zipf distribution, with $\Pr(F^i \in A_j) = (1/i)/(\sum_{1 \leq k \leq m} 1/k)$, sampled without replacement. For any $F^i \in A^j$, bid j requires that impressions satisfy $f_{z_i}^i$, with $z_i \in \{1, 2\}$ chosen uniformly. The bid's formula is the conjunction of all required attributes, $\varphi^j = \bigwedge_{F^i \in A^j} f_{z_i}^i$.

Our bid valuation model reflects the intuition that bidders tend to place higher value on more specific bids (i.e., with more attributes), and higher value if the attributes in their bid formula are in greater demand. We determine bidder j's per impression value v^j as follows. We first draw a "base value" $\hat{v}^j$ from $U[0.1, 1]$ then adjust it by setting $v^j = \hat{v}^j(1 + 10 \sum_{F^i \in A^j} \Pr(F^i))$. That is, if the bid cares about no attributes, then $v^j = \hat{v}_j$, whereas if were to care about all m attributes, then $v^j = 11\hat{v}_j$. A bid's time window w^j is determined by sampling t_1 and t_2 from $U[-10, 40]$, setting $w^j = [\min(t_1, t_2), \max(t_1, t_2)]$, then truncating w^j to lie in $[1, 30]$. This incorporates the idea that some bids have windows that extend beyond the optimization horizon. A bid's budget is set to a fraction of the value of the total supply that it cares about. Namely, if σ_j is the total supply of formula φ^j during window w^j, then the budget is $g^j = \tau^j \sigma_j v^j$ with $\tau_j \sim U[0.1, 1]$.

In addition to the bids above, we include a "market" bid with value 0.1, unlimited budget, and no attribute preferences (i.e., $\varphi =$ *True*). This accounts for value that might be obtained from other sources (e.g., future bids or a spot market).

Optimization parameters. During an iteration of column generation, we continue searching for a suitable split so long as we can find a channel refinement that provides a score that offers a certain minimum improvement over the previous abstraction. Parameter *MI* sets this target: if some refinement offers at least an *MI* fractional improvement over the allocation value of the most recent LP, we continue; if there is no such refinement on any channel, we terminate column generation. Even if there is no *MI* improvement, it does *not* necessarily mean the the allocation value is within *MI* fraction of the true optimal value. Rather, it means there is no *myopic* improvement of at least *MI* that can be obtained within the restricted channel splitting space we consider: some sequence of channel refinements could effect greater improvement.[12]

Estimating an upper bound on the optimal value. To measure how good an allocation is, we need to estimate the true optimum value achievable if we generated all relevant columns. We compute an upper bound on the optimum as follows. When column generation is complete, we run another optimization using *undiscounted* values. That is, we remove all $\Pr(\varphi^i|\alpha_j)$ terms. This is clearly an upper bound on the optimum because it assumes that bids could actually make use of the entire amount of a channel it is allocated (rather than the only $\Pr(\varphi^i|\alpha_j)$ fraction it actually cares about for channel j). However, this is a very loose upper bound. We can tighten it significantly by ensuring that a bid's allocation does not exceed the supply that it actually cares about. That is, we add additional constraints of the form $x_j^i \leq s(\varphi^i \wedge \alpha_j)/s(\alpha_j)$ for all bids i and channels j. This is still an overestimate because it does not account for interactions between multiple bids. However, empirically, this bound is quite close to an even tighter upper bound that we generate via constraint generation (see Sec. 6). Since different optimization approaches and different optimization parameters can give different upper bounds for the same problem instance, we select the tightest (i.e., smallest) valid upper bound over all approaches tried on an instance.

		#	Frac		Runtime (sec)	
m	n	channels	UB	Improve	μ	range
10	100	12.0	0.899	0.500	11	$[4, 24]$
20	200	11.0	0.83	0.367	40	$[8, 74]$
30	300	10.2	0.843	0.381	75	$[35, 150]$
40	400	9.8	0.807	0.335	153	$[28, 556]$
50	500	10.0	0.818	0.397	212	$[23, 418]$
60	600	8.6	0.829	0.344	245	$[33, 470]$
70	700	8.3	0.825	0.304	314	$[26, 660]$
80	800	9.2	0.826	0.345	461	$[101, 940]$
90	900	8.6	0.807	0.323	566	$[75, 1211]$
100	1000	9.3	0.806	0.345	811	$[203, 1438]$

Table 1: Average results for column generation with LP expressiveness and $MI = 0.01$, m attributes, and n bidders.

Experimental results. Table 1 shows results from runs with parameter $MI = 0.01$, averaged over 20 instances for each $\langle m, n \rangle$ pair. The table shows several key measures including the number of a-channels generated. The fraction of the upper bound on the optimal value obtained by the abstract LP when column generation terminates ("Frac UB") is also shown (giving us a lower bound on the quality of the abstract allocation relative to the true optimal allocation). An estimate of the improvement in the degree of optimality is shown ("Improve"). This is reported as the average of $(\mathit{Final} - \mathit{Initial})/\mathit{UB}$, where *Final* is the final LP value, *Initial* is the LP value at the start of column generation (when a single abstract channel is used), and *UB* is the upper bound on the optimal value. Finally, the average and range of runtimes is presented.

Table 2 shows similar results, but for runs with $MI = 0.001$. We see that, with LP expressiveness, column generation can obtain a significant fraction of the upper bound value for problems in which it would be impossible to even enumerate the full unabstracted LP. Setting a lower value for the minimum improvement parameter MI allows us to obtain a greater fraction of the upper bound, but with a fairly significant increase in run time. This suggests adopting a more sophisticated technique that occasionally computes an upper bound during the course of column generation (using the current set of channels), then weighs the additional potential improvement against the amount of time already spent.

Fortunately, although the number of *potential* channels increases exponentially in m and n, our column generation procedure can obtain high value with very few channels. Indeed, the number of generated channels, and the resulting quality of solution, are com-

[12]The restricted space of channel splits we consider can obviously impact our ability to find a suitable refinement. Even without this restriction (i.e., even if splitting into arbitrary pairs of subsets is allowed), one can show that myopic splitting is insufficient in general when IP expressiveness is admitted. For certain forms of LP expressiveness, however, we can show that, if an abstraction is not lossless, there always exists a two-way split of some channel that improves value. Hence a myopic search (over an unrestricted split space) is sufficient to find an optimal, lossless abstraction.

		#	Frac		Runtime (sec)	
m	n	channels	UB	Improve	μ	range
10	100	32.4	0.965	0.515	53	$[10, 112]$
20	200	33.8	0.905	0.439	317	$[21, 758]$
30	300	27.1	0.899	0.438	538	$[112, 1384]$
40	400	28.6	0.871	0.399	1247	$[211, 4159]$
50	500	26.8	0.871	0.450	1543	$[153, 4027]$
60	600	22.7	0.877	0.392	1775	$[88, 4798]$
70	700	19.3	0.867	0.346	1959	$[66, 5878]$
80	800	24.2	0.873	0.393	3746	$[469, 8670]$
90	900	24.0	0.858	0.374	4956	$[807, 14534]$
100	1000	25.7	0.854	0.392	6687	$[1677, 17047]$

Table 2: Average results for column generation with LP expressiveness, $MI = 0.001$, m attributes, and n bidders.

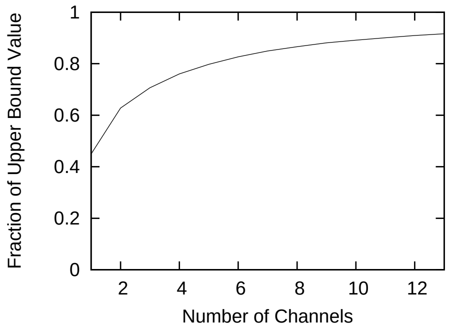

Figure 1: Fraction of upper bound vs number of channels for m = 10, $n = 100$.

parable across all m and n tested. Furthermore, on average, much of the improvement is obtained early in the procedure. Fig. 1 shows the fraction of upper bound obtained after a given number of channels has been generated, averaged over 20 instances, with $m = 10$, $n = 100$, and $MI = 0.001$. We obtain a high fraction of the upper bound from the first few channels generated, with additional channel splitting providing more modest improvement.

5.3.2 *Variants on bid distributions*

We have also run our column generation algorithm on variants of the bid distribution. Table 3 shows the average results for column generation with LP expressiveness and $MI = 0.01$, 100 attributes, 1000 bidders, for different variants. We vary the distribution of number of attributes per bid, the distribution for selecting bid attribute (either Zipf or uniform, both without replacement), the probability $p(z_i = 1)$ that a bid wants value $z_i = 1$ for a given attribute, and the distribution of the base bid value. For these runs, we show estimates both of the initial degree of optimality ("Initial frac UB") with a single abstract channel, as well as the final degree of optimality ("Final frac UB") after column generation.

We obtain a comparable estimate of the degree of optimality for all but the last variant, for which we achieve significantly higher optimality. On the first variant, which corresponds to the last row in Table 1, column generation requires the most time and produces the most channels. However, the initial degree optimality is lowest for this variant, suggesting it is harder than the others.

5.3.3 *IP expressiveness*

The second set of problems includes bidders with all-or-nothing bonus bids, as well as bidders with per-impression values and budgets. Since all-or nothing bids require binary variables, column generation on the LP relaxation offers only an approximation. All

		#	Frac		Runtime (sec)	
n_b	n_i	channels	UB	Improve	μ	range
10	40	7.8	0.873	0.271	41	[10, 103]
20	80	7.9	0.838	0.270	85	[17, 220]
30	120	9.1	0.806	0.295	148	[32, 500]
40	160	9.2	0.809	0.310	181	[47, 455]
50	200	9.8	0.833	0.331	222	[56, 539]
60	240	7.8	0.841	0.311	184	[49, 324]

Table 4: Average results for column generation with IP expressiveness, $MI = 0.01$, 100 attributes, n_b bonus bidders, and n_i per-impression bidders.

problems have 100 attributes, n_b bonus bidders, and $n_i = 4n_b$ per-impression bidders, with $n_b \in \{10, 20, \dots, 60\}$. The preferences of per-impression bidders is determined as before. A bonus bidder had φ^j and w^j chosen similarly. However, its per-impression value is $v^j = 0$, and instead the bidder pays b^j if it receives at least q^j impressions satisfying φ^j, but nothing otherwise. We select q^j to be a fraction τ^j of the total supply the bid cares about, namely, $q^j = \tau^j \sigma_j$, with $\tau_j \sim U[0.1, 1]$, and σ_j the total supply of formula φ^j during time window w^j. We then set $b^j = \hat{b}^j q^j$ where $\hat{b}^j$ is chosen as v^j for a flat bidder, but then multiplied by a factor chosen from $U[1.1, 1.5]$. We also include a "market" bid as above.

Table 4 shows results with $MI = 0.01$, averaged over 20 instances for each $\langle n_b, n_i \rangle$ pair. Shown are the number of channels generated, the fraction of the upper bound (on the optimum) obtained by when column generation terminates ("Frac UB"), the improvement over the fraction of the upper bound obtained before column generation ("Improve"), and the range of runtimes over the 20 instances. Although we use the LP relaxation to determine channel splits, we solve MIPs to determine the abstract allocation and value (hence fraction of the upper bound) obtained.

Although column generation operates on a relaxation of the true MIP, our scoring function is nevertheless effective in guiding our procedure to good channel splits. Indeed, the performance with IP expressiveness compares favorably to that with LP expressiveness. We emphasize that these campaign-level optimizations are run offline, and used to parameterize dispatch policies that are then implemented in real time. Thus the times reported here allow frequent, multiple optimizations (and reoptimization) of offline allocations (e.g., within a stochastic optimization framework [6]).

6. CONSTRAINT GENERATION FOR ABSTRACT OPTIMIZATION

The optimization above, using the abstraction generated by our column generation process, assumes that any ad allocated to an a-channel α will be randomly dispatched to the component c-channels that make up α. This is reflected in the MIP (or LP) objective by replacing the per-impression value v^i of bid i by $v^i_\alpha \Pr(\varphi^i|\alpha)$. With a well-crafted abstraction, this may produce an optimal allocation (e.g., consuming as much of each advertiser's budget as possible). However, if the number of a-channels is limited for computational reasons, the "pessimistic" assumption of random dispatch may leave revenue on the table. We consider another means of optimizing with a-channels that relies on constraint generation in the abstract MIP (or LP) to allocate the supply of abstract channels to bids non-uniformly, thus improving revenue.

6.1 Constraint Generation Procedure

Let A be an abstraction and M the *optimistic MIP* in which bids are assigned to a-channels, but where each impression to bid i is *assumed* to satisfy its formula φ_i. This assumption is realized by

# attributes per bid	Attribute selection	p($z_i = 1$)	Base bid value	# channels	Initial frac UB	Final frac UB	Improve	Runtime (sec) μ	Runtime (sec) range
$U[0,10]$	Zipf	0.5	$U[0.1,1.0]$	9.3	0.461	0.806	0.345	811	$[203,1438]$
$U[0,10]$	Zipf	0.75	$U[0.1,1.0]$	8.3	0.476	0.817	0.341	636	$[813,1509]$
$U[0,10]$	Zipf	1.0	$U[0.1,1.0]$	7.7	0.528	0.818	0.290	542	$[235,1007]$
$U[0,10]$	Zipf	0.5	$N(0.5,0.1)$	8.2	0.489	0.816	0.327	605	$[133,1439]$
$U[0,10]$	Zipf	0.75	$N(0.5,0.1)$	9.0	0.508	0.808	0.301	803	$[123,1839]$
$U[0,10]$	Zipf	1.0	$N(0.5,0.1)$	7.8	0.511	0.800	0.289	595	$[247,1503]$
$U[0,10]$	Uniform	0.5	$U[0.1,1.0]$	3.9	0.743	0.801	0.059	264	$[64,543]$
$U[0,10]$	Uniform	0.75	$U[0.1,1.0]$	4.3	0.752	0.804	0.052	306	$[133,706]$
$U[0,10]$	Uniform	1.0	$U[0.1,1.0]$	3.8	0.748	0.799	0.051	282	$[97,785]$
1	Uniform	0.5	$N(0.5,0.1)$	5.6	0.844	0.973	0.128	322	$[84,571]$

Table 3: Average results for column generation with LP expressiveness and $MI = 0.01$, 100 attributes, 1000 bidders.

replacing the per-impression value v^i_α for a-channel α by v^i itself: i.e., we assume that *every* ad for i assigned to α will be dispatched intelligently, thus guaranteeing that φ^i is satisfied. In a simple two-bid, two a-channel case, the resulting MIP (in this case, LP) is:[13]

$$\begin{array}{llllll} \text{Max} & v^1 x^1_{\alpha_1} & +v^2 x^2_{\alpha_1} & +v^1 x^1_{\alpha_2} & +v^2 x^2_{\alpha_2} & \\ \text{s.t.} & x^1_{\alpha_1} & +x^2_{\alpha_1} & & & \le s(\alpha_1) \\ & & & x^1_{\alpha_2} & +x^2_{\alpha_2} & \le s(\alpha_2) \end{array}$$

The optimistic assumption embodied in this formulation is unreasonable in general. There is no reason to believe the allocation of bids to α_1 permits feasible "packing" of their promised supply in such a way that each bid i gets only φ^i-impressions. However, we can test this assumption by solving an LP that determines whether there is enough supply to do just this: in our example, we want to determine if α_1 contains enough φ^1 and φ^2 supply to meet the "obligations" contained in the solution of the optimistic MIP; similarly, we wish to test a-channel α_2. More generally, let $\dot{\mathbf{x}} = \{\dot{x}^i_{\alpha_j}\}$ be the solution of the optimistic MIP with a-channels $\{\alpha_j\}$. Let $W(j) = \{i : \dot{x}^i_{\alpha_j} > 0\}$ denote the the "winners" of a-channel α_j. We solve the following LP for each α_j (with a constant objective, since our aim is only to determine feasibility):

$$\begin{array}{lll} \max & 1 & \\ \text{s.t.} & \displaystyle\sum_{c\in\alpha_j, c\models\varphi^i} x^i_c = \dot{x}^i_{\alpha_j} & \forall i \in W(j) \\ & \displaystyle\sum_{i\in W(j)} x^i_c \le s(c) & \forall c \in \alpha_j \end{array}$$

This LP determines a feasible allocation of bids i that share α_j to the c-channels that constitute α_j, thus guaranteeing that every impression given to i satisfies its bid condition φ^i. The first set of constraints ensures there is enough φ^i supply for each bid i—call these *bid adequacy constraints*—while the second establishes that no constituent c-channel is overallocated—call these *channel supply constraints*. If $LP(\alpha_j)$ is feasible for each α_j, then it provides an optimal dispatch policy that extracts the full objective value of the optimistic MIP. If not, we post constraints on the optimistic MIP and resolve. In particular, let $LP(\alpha_j)$ be infeasible. Then there must be some minimal set of constraints that are jointly infeasible. Let $S = S_a \cup S_s$ be such a minimal set, where S_a are bid adequacy constraints and S_s are channel supply constraints. We can show that the MIP solution violates the inequality:

$$\sum_{i\in S_a} x^i_{\alpha_j} \le \sum_{c\in S_s} s(c) \qquad (1)$$

[13]As discussed above, in general, we don't discount the *value* of an impression to a bid, but the number of impressions that *count* toward satisfaction of bid conditions. The optimistic MIP replaces all discounted counts by their undiscounted counterparts.

We can resolve the MIP by posting this constraint to ensure that overallocation of the channels in S_s does not occur for the purposes of maximizing value extracted from bids in S_a. A tighter version of this constraint can be employed: we can add to the sum on the lefthand side any bid i all of whose relevant channels are included in S_s, i.e., any i s.t. $\{c \in \alpha_j : c \models \varphi^i\} \subseteq S_s$. At each iteration, sets S leading to violated constraints are identified for each a-channel and posted.[14] The MIP is resolved until feasibility is attained (in which case full optimistic objective value is obtained), or computational or time bounds are reached.

Computationally, the most demanding aspect of this algorithm is the solution of the LPs used to generate constraints. While the solution of $LP(\alpha_j)$ could, in principle, require an exponential number of variables (i.e., the x^i_c corresponding to all c-channels $c \in \alpha_j$) and constraints, we use simple lossless channel abstraction to collapse this number. As such, the number of winners for each channel (and the interaction of their bids) determines the true complexity of the required LP solves.[15] The constraint generation algorithm can be used directly to solve the ad allocation MIP without relying on column generation. For example, it can be applied directly to the fully abstract MIP with a single a-channel ($\top$). It could also be used to optimize over *any* heuristically chosen abstraction.

6.2 Empirical Results

To evaluate the effectiveness of constraint generation we experiment with problems with bonus and per-impression bidders, as described in Sec. 5.3.3. We first perform column generation using $MI = 0.01$, then extend the solution using constraint generation. We initially seed the procedure with all constraints of type (1) involving single bids. Hence, all subsequently generated constraints involve multiple bids.

To avoid generating an unreasonable number of constraints, we use a tolerance ϵ (set to 0.01), whereby the feasibility LP allows the allocations from the MIP to decrease by up to ϵ. That is, we replace the first set of constraints in the LP by:

$$\begin{array}{ll} \displaystyle\sum_{c\in\alpha_j, c\models\varphi^i} x^i_c \le \dot{x}^i_{\alpha_j} & \forall i \in W(j) \\ \displaystyle\sum_{c\in\alpha_j, c\models\varphi^i} x^i_c \ge \dot{x}^i_{\alpha_j} - \epsilon & \forall i \in W(j) \end{array}$$

[14]These can be identified using the facilities of standard solvers, such as the CPLEX IIS (irreducible inconsistent set) routine. We use our own special purpose algorithm to identify such sets.

[15]The interaction is in fact even less when one accounts for time windows: a separate feasibility testing/generation process is invoked for each a-channel, time-period pair.

Thus, when constraint generation terminates, the allocation is guaranteed to be feasible, but may be suboptimal.

We found that, for larger problems, constraint generation did not always terminate within a reasonable amount of time. In our experiments, if constraint generation did not terminate within 600 seconds, we stopped generating constraints and generated a feasible allocation that minimized the maximum difference from the MIP allocation. We accomplish this with the following LP:

$$\begin{aligned} \min \quad & \epsilon & & (2) \\ \text{s.t.} \quad & \sum_{c \in \alpha_j, c \models \varphi^i} x^i_c \leq \dot{x}^i_{\alpha_j} & \forall i \in W(j) & \\ & \sum_{c \in \alpha_j, c \models \varphi^i} x^i_c \geq \dot{x}^i_{\alpha_j} - \epsilon & \forall i \in W(j) & \\ & \sum_{i \in W(j)} x^i_c \leq s(c) & \forall c \in \alpha_j & \end{aligned}$$

As discussed above, the feasibility LP could require an exponential number of variables. In practice, we find that if $W(j)$ is no greater than around 20, the size of the LP is reasonable (and *much* smaller than 2^{20}). If at any point the MIP gives $W(j) > 20$, we split channel α_j. However, rather than using the scoring function discussed above, we attempt to reduce the maximum, over the two new channels, of the bids that care about the channel. That is, we minimize $score(\alpha, \beta, \overline{\beta}) = \max(\{|\{i\}| : \beta \wedge \varphi^i \neq \mathit{False}\}, \{|\{i\}| : \overline{\beta} \wedge \varphi^i \neq \mathit{False}\})$.

When constraint generation is complete, we compute the value of the allocation based on the final feasible allocation generated by the LP (which might be different than that of the final MIP allocation, due to ϵ), but use the final (infeasible) MIP allocation as an upper bound on the true optimum value. This bound is close to, but somewhat tighter than the bound generated in Sec. 5.3.

Table 5 shows the results of experiments on the set of problems with bonus and per-unit bidders described in Sec. 5.3.3. Here we show the results only for the constraint generation portion. The table shows several key measures, including the number of constraint generation iterations, the number of additional channels generated and the number of constraints generated. The fraction of the upper bound on the optimal value obtained by the MIP when constraint generation terminates ("Frac UB") is also shown. An estimate of the improvement in the degree of optimality over the final column generation value is shown ("Improve"). Finally, the average and range of runtimes is presented. Clearly, the additional phase increases value to a high degree of optimality, although obtaining this improvement can be time consuming for larger problems.

We found in our experiments that, typically, little additional value is obtained by performing constraint generation beyond the initial single-bid constraints. We ran additional tests to determine the effectiveness of adding only static, single-bid constraints, without adding additional constraints. In these tests it was still necessary to run the relaxed LP (2) for each channel to determine a feasible allocation. As we see in Table 6, we can get nearly same level of optimality as from generating more constraints but at a significant time savings. In some cases, we obtain slightly higher optimality. This is possible because, even when we generate multi-bid constraints, we still run the relaxed LP in the final step. It is possible for the approximation to be worse, even when we generate the additional constraints.

6.3 Other Uses of Constraint Generation

One of the bottlenecks in the effective use of constraint generation is its tendency to scale poorly in the number of "winners."

		#	Frac		Runtime (sec)	
n_b	n_i	channels	UB	Improve	μ	range
10	40	0.2	0.986	0.113	11	$[4, 37]$
20	80	0.2	0.972	0.134	80	$[12, 545]$
30	120	0.3	0.992	0.186	168	$[17, 848]$
40	160	0.7	0.971	0.162	431	$[22, 2991]$
50	200	0.3	0.985	0.152	608	$[32, 7092]$
60	240	0.9	0.970	0.127	398	$[35, 2259]$

Table 6: Average results for constraint generation with only static, single-bid constraints, following column generation, with 100 attributes, n_b bonus bidders, and n_i per-impression bidders.

Specifically, if an a-channel, time-period pair has a large number of bids that are allocated to it in the initial abstract MIP solve, the procedure can generate hundreds of thousands of constraints, causing the MIP to slow down significantly and dominate runtime. The number of winners in the MIP can be used to suggest further channel refinements. The development of effective channel splitting heuristics that attempt to "separate" bids into different channels could make constraint generation much more effective. The quick identification of problematic a-channels during constraint generation is critical as well: whenever a channel is split, all constraints on the split channel must be discarded, and new ones must be generated on the new channels, further "wasting" computational effort. Thus problematic a-channels should be identified before significant constraint generation occurs.

Constraint generation can also be used selectively. The MIP can be solved by using the "optimistic" values on some channel-time pairs—requiring constraint generation to effectively carve up supply with those segments—while the random dispatch policy can be assumed in others (e.g., those where constraint generation cannot scale effectively). This offers a tractable means for improving on the abstract allocation problem without necessarily accounting for intelligent dispatch across the entire space.

7. DATA REPRESENTATION AND OTHER ISSUES

The implementation and practical deployment of our techniques bring to light a number of subsidiary issues that need to be addressed. We first discuss several ways in which our column and constraint techniques can be extended to further enhance scalability, then outline some additional challenges to practical deployment and how we address them.

7.1 Discussion of Techniques

The column generation procedure converges to an optimal allocation for LP expressiveness, even with our myopic search procedure. Successive conjoining of literals must eventually produce all c-channels; and since our scoring function overestimates improvement in LP objective achieved by any split, all worthwhile splits will be made. Of course, tractability requires that we do not split the channels too finely.[16] To this end, we consider complex splits by allowing both literal conjunction and disjunction during split search. Although complete search is impractical, more sophisticated techniques for constructing split formulae may lead to even better splits. For instance, dynamic programming may be used in special cases (e.g., under certain independence assumptions). Tech-

[16]Standard bounds from the column generation literature can be adapted to our problem to bound the degree of suboptimality should we stop generating channels when some split still has positive reduced cost [12].

		#	#	#	Frac		Runtime (sec)	
n_b	n_i	iterations	channels	constraints	UB	Improve	μ	range
10	40	13.9	0.2	210	0.983	0.110	162	$[12, 620]$
20	80	12.0	0.2	562	0.977	0.139	629	$[89, 1635]$
30	120	8.4	0.4	838	0.982	0.176	858	$[356, 1719]$
40	160	6.2	0.8	727	0.966	0.158	1433	$[625, 6417]$
50	200	5.7	0.3	706	0.978	0.145	1523	$[679, 8993]$
60	240	5.4	1.1	647	0.968	0.127	1257	$[663, 3773]$

Table 5: Average results for constraint generation, following column generation, with 100 attributes, n_b bonus bidders, and n_i per-impression bidders.

niques for constructing logical class and concept descriptions from the classification and concept learning literature—and more generally, methods for feature selection in learning [10]—may also be adapted to our setting.

However, we emphasize that our goal is not to identify the smallest set of channels per se, but rather a set of channels that leads to a high value from optimization while allowing the LP to remain tractable. Our approach obtains high value with a small number of channels. For our larger problems, search dominates runtime, requiring more than nine times as much time as the LP solves. Thus our primary focus is accelerating split search, rather than ensuring completeness. As we show in Sec. 8, simple heuristics can dramatically improve runtime performance of column generation.

Constraint generation can be used independently of column generation, but it is much more tractable if it starts with a good abstraction. While constraint generation can improve an allocation in the case of LP expressiveness, it is most beneficial with IP expressiveness, since column generation is applied to an approximation of the MIP (i.e., its LP relaxation). Since column generation is run on the LP relaxation at the root of the MIP search tree, it is not guaranteed to converge to optimality. Alternatively, we could employ a *branch-and-price* [2] approach, whereby column generation is applied at multiple points in the MIP search tree. This would allow convergence to an optimal allocation in the IP case, but is much more computationally expensive than standard (LP) column generation; it also leads to complications in the cutting plane algorithms needed to solve MIPs efficiently.

7.2 Data Representation

Our approach to channel abstraction requires manipulation of logical formulae describing both abstract channels and bids. Furthermore, the natural and compact description of both bids/campaigns and channel supply requires the use of logical formulae. In problems with dozens or hundreds of channel features, we cannot expect supply distributions to be explicitly articulated for each concrete channel. Nor should we expect bidders to specify their interests explicitly over such concrete channels.

Our data distributions make specific independence assumptions that allow them to be represented tractably. While more general models can be used (e.g., graphical models of distributions such as Bayesian networks), we adopt a simple clustering model. The channel feature set $\mathcal{F}$ is partitioned into a set $\mathcal{H} = \{H_i\}$ of subsets or *factors*, and we assume an explicit joint distribution (or *potential*) ψ_i is provided for each factor H_i (e.g., if $H_i = \{A, B, C\}$, then ψ_i is a joint distribution over $Dom(A, B, C)$). These potentials are independent, so the probability of any channel is $\Pr(c) = \Pi_i \psi_i(c[i])$, where $c[i]$ is the restriction of c-channel c to its feature values in H_i. The supply $s(c, t)$ of any channel at time t is then $s(c, t) = s(t)\Pr(c)$.[17] Our assumption above of complete feature independence is a special case of this model.

[17]If impression distributions are nonstationary, the potentials can be indexed by time, or by time "features" such as day-part.

Our implementation of channel abstraction uses *ordered Boolean decision diagrams (OBDDs)* [7] to represent logical formulae: this includes the logical representation of bid formulae φ^i and of abstract channels α_i. Given the specification of probabilities in terms of factors and potentials, we have devised efficient algorithms for: (a) computing the probabilities of a formula represented as an OBDD (e.g., to compute $\Pr(\alpha)$ for some a-channel α in order to determine its supply); and (b) computing the conditional probability of one OBDD given another (e.g., to compute the probability $\Pr(\varphi^i|\alpha)$ that a-channel α satisfies bid formula φ^i).

7.3 Channel Features and Stochastic Supply

The nature of useful channel features varies significantly from one web site to another. We capture this by aggregating c-channels into groups known as *base channels*, typically corresponding to particular sites (or subsections of sites). Each base channel (e.g., a specific web site) is characterized by its total amount of supply and by the set of features that are *observable* (i.e., features that are known with certainty to hold of a particular impression, such as day-part, gender of subscription users, etc.), *stochastically verifiable* (i.e., features for which a probabilistic estimate of satisfaction can be given), or *inapplicable* (features for which no information is available). The distribution of channels with a base channel is specified using the method above.

C-channels cannot be defined using inapplicable features or stochastically verifiable features: for any base channel, its c-channels are the instantiations of its observable features only. For instance, on a site A that has statistical data on gender, but no means of observing gender, no c-channel exists with features $site = A, gender = male$ (since one cannot assign an ad to such an impression with certainty). The distribution of gender is used only to predict the number of $male$-impressions (hence payment) when an ad is assigned to $site = A$. Similarly, if a feature is inapplicable, every feature value is assumed to go unsatisfied.

Our abstraction model is presented as if supply is deterministic. If supply is stochastic, our abstraction techniques can be generalized using the methods described in [6], where the results of deterministic optimization are used in a sampling and reoptimization framework to manage uncertainty and risk. Our data representation can easily be generalized as well: (a) we replace the point estimate of the supply $s(b)$ of a base channel b by a distribution (e.g., normal, or other parametric form that makes sense and can easily be sampled from); (b) instead of a simple multinomial for each observable attribute, we specify a Dirichlet, with hyperparameters for each domain value. This allows simple computation of expected values for deterministic optimization, and simple sampling for stochastic optimization.

8. COLUMN GENERATION HEURISTICS

Searching for the best channel split dominates the runtime of the column generation process. Indeed, with 100 attributes, 1000 bidders, and $MI = 0.01$, the search consumes over 90% of the runtime, compared to only 3% for the LP solves (with the remaining time devoted to management and bookkeeping). To decrease the runtime, heuristics can be employed in selecting the channel to split and selecting which attribute-values to split on. While such heuristics will result in suboptimal splits, they need not decrease the optimality of the final result so long as we continue column generation to the same minimum improvement tolerance. However, using heuristics may cause more channels to be generated before the tolerance is reached. This tradeoff can be beneficial, so long as the increase in channels is reasonable.

We tried three complementary heuristics that greatly speed up column generation, without sacrificing the optimality of the final allocation. The first two involve speeding up the search for a split on a given channel. The first, "single-value", is to consider splitting channels on only a single attribute-value (as opposed to conjunctions/disjunctions of multiple literals).

The second, "trigger", is to heuristically order the attribute values based on an estimate of their score, then select the first attribute value whose actual score (not the heuristic value) exceeds a threshold (in terms of the fractional improvement over the last LP solve). The key is to make the ordering measure much faster to compute than the actual reduced cost score. Since the most expensive part of computing the score is computing the conditional probability of a bid, given a channel split, we must avoid this computation in the heuristic. For channel α, we order the attribute-value f_k^j by decreasing $h(f_k^j)$, where $h(f_k^j) = \max_{i \in B^j} rc(x^i_{\alpha\wedge\varphi^i})s(\alpha \wedge \varphi^i)$ and B^j is the set of bids that indicate a preference on attribute j. This does not require the computation of any additionl conditional probabilities because $p(\varphi^i|\alpha \wedge \varphi^i) = 1$ and because $s(\alpha \wedge \varphi^i) = s(\alpha)p(\alpha \wedge \varphi^i|\alpha)$ and we already computed $s(\alpha)$ and $p(\alpha \wedge \varphi^i|\alpha)$ for the previous LP solve. For the "trigger" heuristic, we can choose any threshold that is at least as large as the *MI* threshold for determining when to stop searching for splits. In our experiments, we got the best results by setting the trigger threshold to *MI*.

Finally, we tried a heuristic for choosing which channel to split. The "queue" heuristic orders the channels by the maximum split score last computed for the channel. Channels are ordered by decreasing score. New channels that have not yet been scored are given a score of ∞. When deciding which channel to split, we find a split for the first n channels in the queue. We choose the best of those splits that exceeds a threshold. If none of the first n channels has a threshold that exceeds a threshold, we continue down the queue until we find a channel whose split exceeds the threshold. Whenever we choose not to split a channel, we replace the channel on the queue with it's queue value equal to the newly computed split score.

Table 7 shows average results for column generation with LP expressiveness, 100 attributes, 1000 bidders, $MI = 0.01$, and different combinations of heuristics. In the "heuristics" column, "S" refers to the single-value heuristic, "T(t)" refers to the trigger heuristic with threshold t, and $Q(n)$ refers to the queue heuristic with a minimum evaluation of n channels. The first row shows the results with no heuristics, and corresponds to the last row in Table 1.

We see that the heuristics can greatly speed up column generation with nominal impact on optimality (in some cases, even improving it slightly) and only a small increase in the number of channels generated. Furthermore, the most aggressive combination of heuristics (the last row) gives the best results. Note that, although increasing the threshold of the trigger heuristic and increasing the n of the queue heuristic increases the myopic optimality of the chosen split, it does not significantly affect the optimality of the final solution but does slow down the column generation process. Overall, it appears that, with good heuristics, it is beneficial to speed up the process by performing myopically suboptimal channel splits.

	#	Frac	Runtime (sec)	
Heuristics	channels	UB	μ	range
—	9.3	0.806	811	$[203, 1438]$
S	9.9	0.801	640	$[130, 1184]$
S,T(0.05)	9.8	0.802	566	$[117, 1143]$
S,T(0.03)	9.8	0.804	506	$[109, 841]$
S,T(0.02)	10.0	0.806	505	$[177, 952]$
S,T(0.01)	10.3	0.809	432	$[162, 869]$
S,Q(3)	10.0	0.801	470	$[130, 859]$
S,Q(2)	10.0	0.801	390	$[111, 646]$
S,Q(1)	10.0	0.800	344	$[90, 649]$
S,Q(1),T(0.01)	10.9	0.812	292	$[97, 551]$

Table 7: Average results for column generation with LP expressiveness, 100 attributes, 1000 bidders, $MI = 0.01$, and different heuristics.

Heuristics	Approach	#	Frac	Runtime (sec)	
—	col. gen	7.8	0.842	184	$[49, 324]$
—	constr. gen	1.1	0.968	1257	$[663, 3773]$
S,Q(1),T(0.01)	col. gen	10.7	0.847	31	$[7, 57]$
static constraints	constr. gen	1	0.969	612	$[28, 3794]$

Table 8: Average results for column generation, followed by constraint generation, with heuristics and without, with IP expressiveness, $MI = 0.01$, 100 attributes, 60 bonus bidders and 240 per-impression bidders.

We see in Table 8 that heuristics are also effective on problems with IP expressiveness. The first two rows show the results of column generation, followed by constraint generation, withtout any heuristics. These rows correspond to the last row of Table 4 and the last row of Table 5, respectively. The third and fourth rows show column generation with all heuristics applied, followed by constraint generation using only static, single-bid constraints. We see that heuristics greatly speed up both column generation and constraint generation, without sacrificing optimality.

9. CONCLUDING REMARKS

We developed a suite of techniques based on column and constraint generation that effectively tackle the channel explosion problem in the optimal allocation of online ads. Our techniques apply to both simple, current forms of expressiveness (e.g., simple budget constraints) and other, richer forms of campaign-level expressiveness that require the solution of large-scale integer programs. Our experiments demonstrate that high-quality allocations can be determined using very few abstract channels in optimization: this illustrates the desirable sensitivity of our methods to those channel distinctions that have the greatest impact on value (e.g., revenue or efficiency). Our techniques scale to problems with hundreds of attributes and bidders. Given the offline nature of the optimization problem we propose, our computational results suggest that our procedures can be run and rerun frequently to determine, say, (approximately) optimal allocations in stochastic models that require sampling [6].

There are a number of interesting directions in which this work can be extended, in particular, in directions that would enhance scaling to even larger problems. The search for channel splits in column generation, while effective for our problems, is still quite crude, and we suggested several avenues for improving it. The improvements to constraint generation discussed in Sec. 6.3 and exploring branch-and-price techniques remain a high priority as well. Finally, assessing the impact of approximate channel abstraction and/or optimization on incentives in ad markets is of interest.

10. REFERENCES

[1] Zoë Abrams, Ofer Mendelevitch, and John Tomlin. Optimal delivery of sponsored search advertisements subject to budget constraints. In *ACM Conference on Electronic Commerce*, pages 272–278, 2007.

[2] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W. P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46(3):316–329, 1998.

[3] Michael Benisch, Norman Sadeh, and Tuomas Sandholm. Methodology for designing reasonably expressive mechanisms with application to ad auctions. In *International Joint Conference on Artificial Intelligence*, 2009.

[4] Srinivas Bollapragada, Hong Cheng, Mary Phillips, Marc Garbiras, Michael Scholes, Tim Gibbs, and Mark Humphreville. NBC's optimization systems increase revenues and productivity. *Interfaces*, 32(1):47–60, 2002.

[5] Christian Borgs, Jennifer Chayes, Omid Etesami, Nicole Immorlica, Kamal Jain, and Mohammad Mahdian. Bid optimization in online advertisement auctions. In *Workshop on Sponsored Search Auctions*, 2006.

[6] Craig Boutilier, David C. Parkes, Tuomas Sandholm, and William E. Walsh. Expressive banner ad auctions and model-based online optimization for clearing. In *AAAI Conference on Artificial Intelligence*, pages 30–37, 2008.

[7] Randal E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, 1986.

[8] Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007.

[9] Jon Feldmann, S. Muthukrishnan, Pal Martin, and Cliff Stein. Budget optimization in search-based advertising auctions. In *Workshop on Agent Mediated Electronic Commerce*, 2007.

[10] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[11] Sébastien Lahaie, David C. Parkes, and David M. Pennock. An expressive auction design for online display advertising. In *AAAI Conference on Artificial Intelligence*, pages 108–113, 2008.

[12] Marco. E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1007–1023, 2005.

[13] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vaziran. Adwords and generalized on-line matching. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 274–273, 2005.

[14] David C. Parkes and Tuomas Sandholm. Optimize-and-dispatch architecture for expressive ad auctions. In *Workshop on Sponsored Search Auctions*, 2005.

[15] P Rusmevichientong and D. P. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *ACM Conference on Electronic Commerce*, 2006.

[16] Tuomas Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

[17] Hal R. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.