Value-directed Compression of Large-scale Assignment Problems

Tyler Lu and **Craig Boutilier**
Department of Computer Science
University of Toronto
{tl,cebly}@cs.toronto.edu

Abstract

Data-driven analytics—in areas ranging from consumer marketing to public policy—often allow behavior prediction at the level of individuals rather than *population segments*, offering the opportunity to improve decisions that impact large populations. Modeling such (generalized) assignment problems as linear programs, we propose a general *value-directed compression* technique for solving such problems at scale. We dynamically segment the population into *cells* using a form of column generation, constructing groups of individuals who can provably be treated identically in the optimal solution. This compression allows problems, unsolvable using standard LP techniques, to be solved effectively. Indeed, once a compressed LP is constructed, problems can solved in milliseconds. We provide a theoretical analysis of the methods, outline the distributed implementation of the requisite data processing, and show how a single compressed LP can be used to solve multiple variants of the original LP near-optimally in real-time (e.g., to support scenario analysis). We also show how the method can be leveraged in integer programming models. Experimental results on marketing contact optimization and political legislature problems validate the performance of our technique.

Introduction

A variety of business and policy decisions can be formulated as generalizations of assignment problems, in which the treatment of different individuals in some target population is optimized subject to constraints on the resources used to implement these decisions. In marketing and advertising, a business with multiple campaigns to deploy in a target market must decide which customers to target with which campaigns to maximize business objectives, subject to constraints on campaign budgets, contact limits, etc. In planning public infrastructure (e.g., health-care facilities), one might strive to maximize societal satisfaction by determining which facilities to develop—and the individuals who have access to specific facilities—subject to budget and other feasibility constraints. This perspective applies to many decisions that span large populations.

The ability to make high quality decisions in such settings has improved considerably in recent years with the availability of rich data sources, advances in machine learning, and large-scale distributed computation. These have made it possible to accurately predict the impact of decisions on *specific individuals*. In marketing, for instance, transaction, survey, behavioral/usage and social media data allow marketers to predict behavior, value, and responses (e.g., to messaging or pricing) at the level of individual customers rather broad customer "segments." Online data collection may soon enable similar user-level predictions of behavior and satisfaction in traffic routing, education planning, political and public policy, and other areas.

While such granular predictive modeling enables better decisions *in principle*, in practice it often makes the corresponding optimization problems more difficult, since one can distinguish any individual from any other w.r.t *some* prediction. Modeling the impact of decisions on *individuals*, rather than larger population "segments," dramatically increases the number of decision variables in optimization formulations. Since such (generalized) assignment problems are often most naturally formulated as mathematical programs (e.g., linear (LP) or mixed integer (MIP) programs), this can induce significant computational bottlenecks, rendering problems virtually unsolvable at the individual level. At the same time, optimization over predefined, statistical segments can waste the value of detailed predictive models.

In this work, we develop a compression technique for LP-based assignment problems over large populations that addresses this challenge. *Dynamic cell abstraction (DCA)* is a dynamic segmentation technique that creates a small set of user segments (or *cells*) based on predictive models and optimization objectives. Our segmentations have a very small number of cells, rendering optimization tractable—indeed, able to support *real-time optimization*. At the same time, these cells are *provably optimal*; i.e., all users in a cell would be treated the same even if we had the computational power to optimally "personalize" decisions *without segmentation*. The method works by iteratively refining cells, solving intermediate LPs, until a (near) optimal cell set is constructed. DCA uses *variable aggregation* (Zipkin 1980) to model cells, and a form of *column generation (CG)*, a classic technique for tackling LPs and MIPs with large numbers of variables (Lübbecke and Desrosiers 2005), to evaluate the addition of new cells. Specifically, we modify CG using a method introduced by Walsh *et al.* (2010) in the context of online advertising to score the introduction of multiple columns/variables simultaneously. Our search process can ensure convergence to optimality for certain classes of problems, though it may reach a local optimum in general (but our empirical results suggest that near-optimality is achievable in practice).

We develop our model using a specific problem, that of *marketing optimization* over a large target market, for concreteness. However, we emphasize that the general approach can be applied to a wide variety of problems involving the constrained assignment of options/treatments or allocation of resources to individuals across a large population for whom personalized (known or predicted) values or utilities for specific options are available. (This includes, but is not limited to, generalized assignment problems.) To illustrate the broader applicability, we test our method on a legislature selection problem using a larger voter preference data set. We also develop our methods primarily for LP models. However, we will illustrate how it could be applied to MIP models (and include an empirical test of one such application). We outline a distributed computational paradigm that accelerates cell creation, derive error bounds on solution quality, discuss how the method supports real-time optimization for scenario analysis. Our empirical empirical results show that massive LPs—unsolvable in uncompressed form—can be compressed relatively quickly and in a way that can be effectively distributed, and once compressed, can be solved in milliseconds.

Problem Formulation

We focus on a formulation of the *multi-campaign, multi-channel marketing optimization problem (MMMOP)* that inspired this work. However, many assignment problems can be specified in the same fashion—we use MMMOP for concreteness only. The (abstract) problem we consider bears tight connections to both *generalized assignment problems (GAPs)* (Savelsbergh 1997) and *multi-dimensional knapsack problems (MDKPs)* (Puchinger, Raidl, and Pferschy 2010), but generalizes both.

Problem Components. An MMMOP is defined over a set of *customers* $\mathcal{S}$, a set of *marketing campaigns* $\mathcal{C}$, a set of *channels* $\mathcal{H}$, a family of *response sets* $R_{j,h}$ for each $j \in \mathcal{C}, h \in \mathcal{H}$, and data and constraints defined below. Each campaign $j \in \mathcal{C}$ is typically designed to meet a specific marketing objective (e.g., acquire new customers, cross-sell or up-sell new services or features to existing customers, retain customers, etc.) It is not unusual for large brands in financial services, telecom, or retail to run 75–200 simultaneous campaigns, while online marketplaces run thousands of campaigns for participating vendors.

$\mathcal{S}$ represents the potential customer targets of the marketing campaigns $\mathcal{C}$.[1] A customer can be *approached* with zero or more campaigns, each delivered through a specific channel $h \in \mathcal{H}$. (e.g., direct mail, email, inbound or outbound telemarketing, POS offers, etc.). We call a campaign-channel pair (j, h) an *approach*. Each approach (j, h) to customer i induces a *response* $r \in R_{j,h}$ indicating some customer behavior (e.g., if j offers an extension of the term for a discounted credit card interest-rate via telemarketing h, possible responses might be: no response answer; accepts offer, and 12mo. churn probability is reduced by 25%; rejects offer/churn probability reduced by 15%; etc.). We use "response" in a broad sense, indicating any immediate or long-term (change in) behavior.

Data: Models, Objective, Constraints. Given these basic elements, we assume the following data. The *unit cost* u_{jh} of approach $\langle j, h\rangle$ reflects the (response-independent) cost of approaching a single customer. Fixed costs are ignored for now (but see below). We assume a set of *predictive models* used to predict customer behaviors, campaign responses, and the value of those behaviors and responses to the marketer. A *response model* provides a probabilistic prediction $p_{ijh}(r)$ of customer i's response to approach $\langle j, h\rangle$. A *value model* $v_{ijh}(r)$ predicts the value of i's response r to approach $\langle j, h\rangle$ (incl. any response-specific costs). The *expected value* of $\langle j, h\rangle$ to i is:

$$v_{ijh} = \sum_{r \in R_{j,h}} v_{ijh}(r) p_{ijh}(r).$$

Generally, predictive models are learned from data, and depend only on the values of a (possibly small) set of customer attributes. Such models are sometimes further decomposed into more fine-grained models that predict specific customer behaviors that must be aggregated to determine comprehensive value and response models (e.g., i's response probability for $\langle j, h\rangle$ may be decomposed into a reachability and "received offer" conditional response probability). Similarly, values may be a function of several distinct responses and behaviors, and often require some calibration. For instance, if a business cares both about the total number of subscribers in addition to revenues, then some value must be associated with each customer acquired or retained that can be compared to (traded off against) revenue. We assume this calibration has been made, but we recognize the difficulty in doing so, and return to this point when discussing *scenario analysis* below.

We assume approaches do not *interfere*, or influence one another: if i receives multiple approaches, her response to each is independent. Interference can be modeled using, e.g., stochastic choice models (Louviere, Hensher, and Swait 2000), which require more complicated formulations due to (a) the combinatorics of choosing *sets of offers*; and (b) the nonlinear nature of these effects (e.g., fractional programming models can be used, a model we defer to future investigation).

Our goal below is to maximize expected campaign value, subject to certain natural constraints. A critical constraint is a *contact limit* L: no customer can receive more than L approaches.[2] In marketing, *over-contacting* is a key concern

[1]We assume a fixed set of *identifiable* customers, but our methods apply to settings in which customers are unknown *a priori*, but some joint distribution of customer attributes is available.

[2]The contact limit need not be constant, but can vary with customer attributes, channel or campaign.

since it can cause customers to ignore subsequent messaging. We also assume: *channel capacities* (or limits) L_h on the usage of each channel h; *campaign budgets* B_j limiting the cost incurred by campaign $j \in \mathcal{C}$; a *global budget* B limiting total cost; and *lead limits* L_j restricting the number of customers that can be contacted by $j \in \mathcal{C}$.

An LP Model. We consider the campaign optimization problem without fixed costs. Let binary variable x_{ijh} denote that customer i receives approach $\langle j, h \rangle$. We can formulate the MMMOP using the following MIP:

$$\max_{x_{ijh}} \quad \sum_{i,j,h} x_{ijh}(v_{ijh} - u_{jh}) \tag{1}$$

$$\text{s.t.} \quad \sum_{j,h} x_{ijh} \leq L \qquad \forall i \in \mathcal{S} \tag{2}$$

$$\sum_{i,h} x_{ijh} u_{jh} \leq B_j \qquad \forall j \in \mathcal{C} \tag{3}$$

$$\sum_{i,j,h} x_{ijh} u_{jh} \leq B \tag{4}$$

$$\sum_{i,h} x_{ijh} \leq L_j \qquad \forall j \in \mathcal{C} \tag{5}$$

$$\sum_{i,j} x_{ijh} \leq L_h \qquad \forall h \in \mathcal{H} \tag{6}$$

$$x_{ijh} \in \{0, 1\} \qquad \forall i \in \mathcal{S}, j \in \mathcal{C}, h \in \mathcal{H} \tag{7}$$

The very large number of integer assignment variables x_{ijh} makes this problem intractable. However, in many settings (including MMMOPs), one can relax these variables, forming an LP by allowing $x_{ijh} \in [0, 1]$. We can interpret this as allowing stochastic assignments (x_{ijh} is the probability of i receiving approach (j, h)), where constraints on budget, etc. are satisfied in expectation. MMMOPs resemble MDKPs, so even this relaxation, in practice, admits few fractional values in the optimal solution (only at the "edges" of the solution, where one "packs" a few lower-value customers into specific approaches (Puchinger, Raidl, and Pferschy 2010)).

Other constraints and modeling elements can be accommodated easily (e.g., limiting customers to one contact per campaign, varying contact limits by customer type or segment, allowing variable costs per customer). Our techniques can be applied *mutatis mutandis* to such extensions. If we include fixed costs f_j for using a campaign j (or channel h), then we require (non-relaxable) integer variables: a 0-1-indicator variable I_j for each $j \in \mathcal{C}$ (has j has been delivered to *any* customer). We discuss how to apply our method to the relaxation of such MIPs below.

Dynamic Segmentation

The LP model of an MMMOP has a huge number of decision variables: the x_{ijh} grow with the product $|\mathcal{S}||\mathcal{C}||\mathcal{H}|$ of the number of customers, campaigns and channels. As such, even this relaxed problem cannot be solved using state-of-the-art LP solvers for problems of realistic size (see below). We now describe a *dynamic segmentation* method that segments the customer population into *cells* of various sizes, and optimizes the approach for each cell rather than for individual customers. If the number of cells is kept small, scalability is no longer an issue. At the same time, unless cells are crafted carefully, significant value may be sacrificed. Our technique, which we call *dynamic cell abstraction (DCA)*, provides an *anytime approach* that gradually refines cells and will converge to a set of cells that admit an optimal solution in certain cases. DCA creates cells that distinguish customers based on specific attributes, and while any customer attributes may be used, here we focus on the cells that distinguish certain *values* v_{ijh} associated with specific approaches (j, h). In practice, DCA finds good or optimal solutions with a small number of cells by finding *just those customer distinctions required to make optimal decisions* and nothing more.

Approximate Optimization with Customer Cells. We first detail how the optimization model exploits customer cells. Suppose we partition the customer population $\mathcal{S}$ into a set of K covering and exclusive *cells* or *segments* $\mathcal{S} = \cup_{k \leq K} \mathcal{S}_k$, s.t. $\mathcal{S}_k \cap \mathcal{S}_{k'} = \emptyset$ for any $k \neq k'$. We call such a partitioning a *segmentation*. Let $z_k = |\mathcal{S}_k|$ denote the size of cell k. For the purposes of optimization, we treat all customers in a cell as if they have the same *expected value* to any approach $\langle j, h \rangle$, namely, the average value across all customers in the cell, i.e., as if we randomly picked a customer s_k from that cell to approach. Letting s_k be a generic customer from cell $\mathcal{S}_k$, we define:

$$v^k_{jh} = v_{jh}(s_k) = \frac{1}{z_k} \sum_{i \in \mathcal{S}_k} v_{ijh}.$$

To target customer segments, we replace individual targeting variables x_{ijh} in LP (1) with variables x^k_{jh} for each cell or segment, where $x^k_{jh} \in [0, 1]$ is the fraction of customers in cell k that receive approach $\langle j, h \rangle$. This can be viewed as *aggregating* the variables corresponding to the customers in a cell into a single aggregate variable for that cell, in a manner we discuss below. The *compressed LP* is:

$$\max_{x^k_{jh}} \quad \sum_{k,j,h} x^k_{jh} z_k (v^k_{jh} - u_{jh}) \tag{8}$$

$$\text{s.t.} \quad \sum_{j,h} x^k_{jh} \leq L \qquad \forall k \leq K \tag{9}$$

$$\sum_{k,h} x^k_{jh} z_k u_{jh} \leq B_j \qquad \forall j \in \mathcal{C} \tag{10}$$

$$\sum_{k,j,h} x^k_{jh} z_k u_{jh} \leq B \tag{11}$$

$$\sum_{k,h} x^k_{jh} z_k \leq L_j \qquad \forall j \in \mathcal{C} \tag{12}$$

$$\sum_{k,j} x^k_{jh} z_k \leq L_h \qquad \forall h \in \mathcal{H} \tag{13}$$

$$x^k_{jh} \in [0, 1]. \qquad \forall i \in \mathcal{S}, j \in \mathcal{C}, h \in \mathcal{H} \tag{14}$$

The compressed LP (8) assumes the value of an approach assigned to cell k is the expected value of that approach over all customers $i \in k$ (e.g., as if each approach is assigned uniformly at random to some i). This random assignment is feasible and by linearity of expectation attains the objective value of the LP in expectation.[3] If $L > 1$, the random

[3]If the fractions give non-integral approaches, small rounding

assignment must be over L-tuples of approaches with positive fractions in k to ensure the contact limit L is respected (e.g., if $L = 2$ and 3 approaches a, b, c are assigned fractions p_a, p_b, p_c (resp.) of k, then a random assignment of *pairs* $(a, b), (a, c), (b, c)$ is needed with *proportions* p_a, p_b, p_c; this requires solving a small linear system).

The compressed LP *underestimates* the value of the assignment by assuming a random allocation. If approach $\langle j, h\rangle$ is assigned m customers from cell k where m is significantly less than the cell size z_k, the allocation could be realized using customers $i \in \mathcal{S}_k$ that have higher value than the mean. With multiple approaches assigned to the same cell, the optimal assignment (i.e., the optimal packing) may have much greater value than the mean value for each approach. Hence we can often improve the objective value by splitting certain cells, as we now discuss.

Dynamic Cell Abstraction. To discover useful cell splits, we adapt the method of Walsh *et al.* (2010), originally developed in the context of display advertising. Similar to *column generation (CG)*, we use the *reduced costs* produced in the solution of the compressed LP to estimate the value of splitting a cell.

CG is used for LPs (and MIPs) with large numbers of variables (Lübbecke and Desrosiers 2005; Barnhart et al. 1998): since only a small subset of the variables are active in the optimal solution, one solves a relaxed problem using only a few variables, and iteratively estimates which variables will be active, gradually adding them to the LP, resolving a slightly larger LP at each iteration. CG assesses the value of a new variable using the *reduced cost* of a (missing) variable x, which is $rc(x) = v_x - \mathbf{c}\boldsymbol{\pi}$, where v_x is the objective function coefficient of x in the (unrelaxed) LP, $\mathbf{c}$ is the column vector of constraint coefficients for x in the (unrelaxed) LP, and $\boldsymbol{\pi}$ is the vector of dual variables at the optimal solution of the *relaxed* LP. The reduced cost $rc(x)$ corresponds to the marginal increase in objective value per unit increase in (nonbasic) variable x if one were to add x to the LP. One typically adds the variable that has maximum reduced cost and iterates. If all columns have non-positive reduced costs, then the solution of the relaxed LP is in fact the optimal solution to the full LP; hence this approach can be used to prove the optimality of the relaxed solution.

In our setting, when we split a cell k into two new cells k_1 and k_2, we do not add a single variable to the compressed LP. Rather we are: (a) adding the *set* of variables $x^{k_1}_{jh}$ and $x^{k_2}_{jh}$ for all (j, h); and (b) removing all variables x^{k}_{jh}. Removing variables doesn't impact the LP—every assignment realizable with cell k can also be realized with k_1 and k_2. But adding these new variables is somewhat problematic because certain constraints are not present in the compressed LP. Specifically, the "supply constraint" Eq. (9) associated with the new cells k_1 and k_2 are not present in the compressed LP. Since we haven't explicitly modeled these two new cells, we do not have dual variables for their constraints, which makes pricing of these columns difficult. However, it is not hard to show that the solution of the compressed LP is also an optimal solution to the LP that is obtained by adding the supply constraints for k_1 and k_2 to the compressed LP Furthermore, we can show there is an optimal solution in which the corresponding dual variables π_{k_1} and π_{k_2} are zero (see the Appendix: Pricing of Columns for details). Hence, we can accurately score a column corresponding to a new split cell k_1 or k_2 using only the dual values produced by the original compressed LP.

More precisely, suppose cell k' is a descendant (one side of the split) of a cell k that is under consideration. We define the reduced cost of the variable $x^{k'}_{jh}$ to be: $rc(x^{k'}_{jh}) = z_{k'}(v^{k'}_{jh} - u_{jh}) - \mathbf{c}^{k'}_{jh}\boldsymbol{\pi}$, where $\mathbf{c}^{k'}_{jh}$ is the column vector in the constraint matrix for $x^{k'}_{jh}$, and $\boldsymbol{\pi}$ is the vector of dual variables in the compressed LP. This gives us $rc(x^{k'}_{jh}) =$

$$z_{k'}(v^{k'}_{jh} - u_{jh}) - \pi_k - z_{k'}(u_{jh}\pi_{B_j} - u_{jh}\pi_B - \pi_{L_j} - \pi_{L_h}). \quad (15)$$

Finally, we must score the *split* of a cell k into k_1 and k_2, given that it introduces a *multiple* columns (which replace others). Suppose we ignore budget, lead and capacity constraints, and focus on the simple problem with only contact limits (Eq. 9). If we split k into k_1 and k_2, all customers in new cell k_1 will be assigned to the approach $\langle j^*, h^*\rangle$ that has highest expected value $v^{k_1}_{j^*h^*}$ for that new cell. Therefore the improvement in expected value will in fact be $rc(x^{k_1}_{j^*h^*})z_{k_1}$. Thus we score splits using $score(k, k_1, k_2) =$

$$\max_{j\in\mathcal{C},h\in\mathcal{H}}\{rc(x^{k_1}_{jh})z_{k_1}\} + \max_{j\in\mathcal{C},h\in\mathcal{H}}\{rc(x^{k_2}_{jh})z_{k_2}\}. \quad (16)$$

This score serves as a reasonable proxy for estimating the marginal improvement of a split, even with other constraints are present (as we see below empirically).

Searching for Splits. With the means to evaluate splits in place, we need a method to search through the space of potential splits to find one with maximal score. Of course, a cell of size z has 2^z possible (binary) splits, so we must restrict attention to a smaller number of reasonable splits. Much like decision tree induction, an ideal "split language" should be compact enough to allow all splits to be effectively evaluated, while also offering the expressiveness and flexibility to find near-optimal solutions with relatively few cells. Here we consider splitting cells using *value quantiles*. Specifically, we treat each approach (j, h) as a real-valued *feature* over $\mathcal{S}$, with value v_{ijh} for customer i, and allow a cell to be split at a specific *quantiles* $Q = \{q_1, \ldots q_T\}$ of each such feature. Focusing on binary splits, a cell k can be split into a pair of cells k^1 and k^2, where $k^1 = \{i \in k : v_{ijh} \leq \tau(q_t)\}$, $\tau(q_t)$ is the q_tth quantile of $\{v_{ijh} : i \in k\}$ for some $q_t \in Q$ and approach (j, h), and $k^2 = k \setminus k^1$. In our experiments below, we obtain excellent results using only *medians*, i.e., $Q = \{0.5\}$. If we have c cells, we need to evaluate scores (Eq. 16) for $c \cdot T|\mathcal{C}||\mathcal{H}|$ splits.

The evaluation of candidate splits for k requires computation of several sufficient statistics: first, the relevant value quantiles q_t of k for each approach; values $v^{k'}_{jh}$ for each resulting split k'. Each of these operations requires a pass over

errors may arise; but since cell sizes tend to be in the many thousands, these are negligible.

all "customer records" in cell k for each $q \in Q$.[4] This can be computationally intensive for large customer sets $\mathcal{S}$. Indeed, as we show below, this is the primary computational bottleneck in DCA. Fortunately, the computation of these sufficient statistics is effectively distributable,

To distribute the computation of quantiles and cell mean values v^k_{jh}, we partition customer records across M compute nodes, with (roughly) $|\mathcal{S}|/M$ records per node. The number of compute nodes can be chosen to meet specific performance demands (e.g., to allow all data to fit in memory). Computing the mean value v^k_{jh} of approach (j, h) for cell k can be accomplished in $O(|\mathcal{S}|/M)$ time, by passing sum and count data from each compute node to a master node. Computing quantiles in a distributed fashion is somewhat less straightforward, but various methods for approximating quantiles in distributed settings (including sampling-based methods) can be used (Shrivastava et al. 2004). Note that computing exact quantiles is not essential to the performance of DCA. We have implemented DCA as a series of map-reduce operations using *Apache Spark* (Zaharia et al. 2010), a general-purpose cluster computing engine, on top of Hadoop's distributed filesystem, with suitable caching, to allow effective in-memory, distributed computation.

Solution Quality. DCA may not converge to an optimal solution in general, in part, because of restrictions on the splits allowed. However, in the case of unrestricted splits, even if only binary splits are considered, we can show that in models with only contact limits (no campaign-specific budget or lead limits), DCA will converge to an optimal solution. (A brief proof sketch is provided in the Appendix: Convergence with Only Contact Limits). With budget or lead limits, DCA can reach a local optimum.

An attractive feature of DCA is its anytime character: a feasible solution to the uncompressed LP can be obtained at any point, using results from the current set of cells. Adapting a result of Zipkin (1980) on the quality of LPs with aggregated variables, we can show the following *a posteriori* approximation bound:

Theorem 1. *Let $\tilde{V}$ be the optimal objective value of the compressed LP (8) using segments S generated using DCA, and let V^* be the optimal objective value of the uncompressed LP (1). Let $\mathbf{c}_{ijh}$ be x_{ijh}'s column in LP (1). Finally, let $\boldsymbol{\pi}$ be the vector of dual values obtained when solving compressed LP (8), and $\widehat{\boldsymbol{\pi}}$ be the* extended dual vector *with the value π_k for each $k \in S$ replaced by the* set *of values $\pi_i = \pi_k/z_k$ for all $i \in k$. Then:*

$$V^* \leq \tilde{V} + \sum_{k \in S} \sum_{jh} \sum_{i \in k} [(v_{ijh} - u_{jh}) - \mathbf{c}_{ijh}\widehat{\boldsymbol{\pi}}]^+ \cdot L.$$

Here $[\cdot]^+$ denotes $\max(\cdot, 0)$. A proof is provided in the Appendix: Proof of Error Bound. The bound uses terms made available from the solution of the compressed LP. Computing it does require summing over a term for each variable in the original LP. However, this computation can be distributed (as above). The bound can also be estimated by sampling a small set of customers from each cell.

Scenario Analysis. In many decision problems, it may be difficult to specify objectives and constraints precisely. For instance, in MMMOPs, marketing groups are often unable to precisely articulate tradeoffs between different performance indicators that contribute to the objective function, or may be unsure of suitable global or campaign-level budget constraints. In such cases, it makes sense to solve multiple versions of the MMMOP, with varying (say) budget limits to allow one to explore the "sweet spot" in marketing spend. As we show below, the vast majority of computational effort in DCA is spent in generating cells (LP solve times are negligible). As such we present one way to amortize the costs of cell generation over multiple optimizations.

One way to do so is to use DCA to solve multiple "representative" LPs at each iteration. For instance, if we anticipate the need explore a variety of budget levels, we can solve an LP for several representative budget levels that "span" the anticipated range. We then use *the solutions of these multiple LPs* to assess the value of a split. Specifically, we score each split for *each* of the LPs using the usual method above. But since our goal is to find cells that support the solution of *all LPs*, we sum these scores to determine the overall score of a split (weighted sums could also be used). This creates a set of cells whose quality can be assessed *for all representative LPs simultaneously*. Furthermore, these cells can be used for other LPs (e.g., an LP with entirely new budgets distinct from any representative LP). Finally, the initial set of cells may simply be used as a starting point, and then *refined* using several additional iterations of DCA to attempt to fit a new LP. We provide preliminary evaluation of the first two approaches below.

Empirical Analysis

The experiments in this section illustrate the performance and value of DCA. We begin with MMMOPs on customer data sets of various sizes (250K, 500K, 750K, 1M, 2M, 4M, 6M, and 10M) using a set of 20 campaigns and 5 channels. Data are generated randomly with a simulator developed using real campaign predictive/value data. All DCA splits are binary and are restricted to medians (quantile 0.5). All LPs use Gurobi Optimizer 5.6 on a single high-performance, large memory compute node (dual Intel 2.6GHz processors, 244 Gb RAM); sufficient statistic computation is distributed across a variable number of compute nodes.

Fig. 1 shows solution quality as a function of the number of DCA iterations (up to 101 iterations, or 101 cells). For customer sizes up to 1M, solution quality is shown as a percentage of the optimal LP value (computed by solving the exact LP model Eq. 1). For the 2–10M instances, solution quality relative to the final value at iteration 101 is shown (the optimal benchmark is unavailable). The anytime profile on large problems is very similar to the smaller instances, suggesting convergence to a near-optimal result. We see that DCA is able to dynamically segment customers so that very few cells are required to achieve near-optimal results. This is true even with the restriction to median splits.

[4]A record for $i \in \mathcal{S}$ includes an ID, a cell indicator and the set of values v_{ijh} for all approaches (j, h).

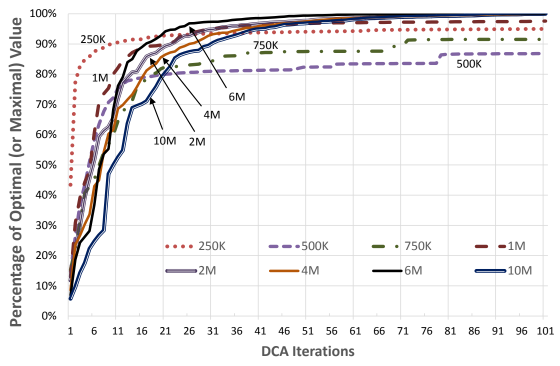

Figure 1: Solution quality vs. DCA Iterations.

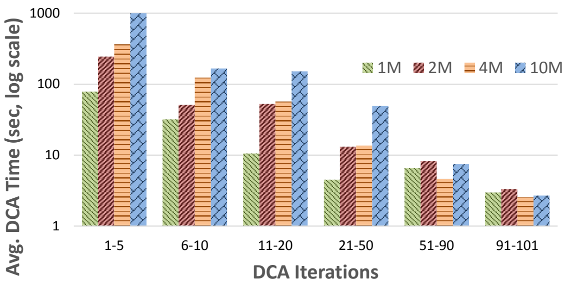

Figure 2: Avg. Time per DCA Iteration (binned, log scale).

Total DCA run time (in seconds) on a 10-node cluster is shown in the first row of the following table:

Size	.25M	.5M	.75M	1M	2M	4M	6M	10M
DCA time (s)	611	679	847	1437	3072	3636	6971	9129
DCA opt (s)	0.022	0.016	0.015	0.012	0.018	0.016	0.019	0.015
LP opt (s)	163	2458	3412	4434	–	–	–	–

Fig. 2 shows average time per iteration of DCA on larger instances binned by iteration number. Later iterations of DCA require significantly less time, since cell sizes decrease exponentially; indeed, noting the log-scale, we see that iteration times exhibit an exponential decrease (the final 10 iterations average 2-3s. each). This is critical for real-time cell refinement in scenario analysis.

DCA also scales effectively with the number of compute nodes used to assess cell statistics. Total DCA time on the 10M instance (101 iterations) reduces nearly linearly with the number of nodes (with a small 7–8% overhead):

Nodes	10	20	40
DCA time (min)	152.2 (100%)	87.2 (57.3%)	50.2 (33.0%)

Because the number of cells needed for high-quality solutions is so small, optimization time using DCA is negligible, on the order of hundredths of a second (see second row of table above). By contrast, the uncompressed LP (see third row) is only feasible for problems of up to size 1M, beyond which we hit memory limits—despite the high-memory platform used. The 1M-instance took about 74 min. to solve. Indeed, with the exception of the 250K instance, uncompressed LP times are significantly longer than the *entire DCA process*.[5]

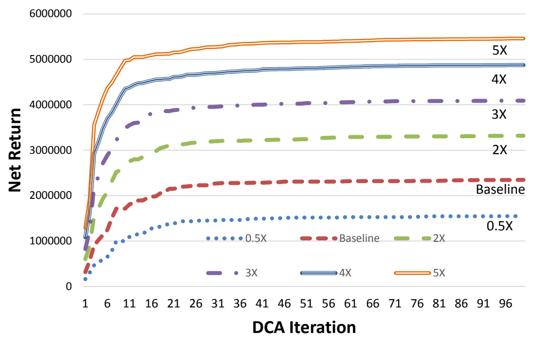

Figure 3: Solution Quality of Multiple LPs.

Of course, once DCA is run, cells can be used for *multiple* optimizations, as we now explore.

To test scenario analysis, we applied DCA to the simultaneous solution of 1M-customer instances using six lead limits: a base level as above, with five additional LPs solved with different lead limits (0.5, 2, 3, 4 and 5 times the base level for all global and campaign-specific limits). Note that *a single collection of cells* is produced to solve all six optimization problems. Fig. 3 shows the change in the objective value for *each* LP as the number of cells increases. Using only 101 cells (as with the single LP), we obtain near-optimal solutions *for all six LPs*: 96.0% (0.5X); 96.7% (Baseline); 98.0% (2X); 98.2% (3X); 98.5% (4X); and 98.5% (5X). The slight bias towards splits that favor the LPs with larger limits is due to the fact that we use an *unweighted* sum of objective values (larger limits offer greater total return on spend). The difference in DCA time/iteration w.r.t. solving a *single* LP is negligible. To test the robustness of these cells, we used them to compute solutions for *different lead limits* than those used to create the cells. With limits of 2.5X and 4.5X, the solutions produced using the cells above were 98.6% and 98.4% optimal, respectively. While a preliminary, these results suggest that cells produced for a well-chosen set LPs can generalize to new problems.

Finally, we tested DCA on a political preference data set derived from a questionnaire completed by 1.2M voters prior to the 2013 national election in Australia, each of whom responded to 59 poll questions, of which 29 were policy questions on eight broad issues.[6] We aggregate each voter's responses within each issue to compute her "ideal position" on that issue. Using a slate of 18 fictitious candidates with varying platforms, we compute a voter's satisfaction with each candidate using (negative) L_1-distance between her ideal point and the candidate's platform. We then used DCA to select a legislature of 5 candidates to best represent this population using the Monroe's (1995) proportional representation scheme—each voter is assigned to one of the 5 selected candidates, satisfaction is dictated by the assigned

[5]We tested attribute-based segmentation of the 1M-customer instance, using two sets of available customer attributes with 738 (resp. 26198) cells. These give solutions that are 51% (resp. 72.5%) of optimal, (soln. times 0.1s. and 4.3s.), compared to 97.6%-optimality attained by DCA with 100 cells.

[6]Administered by VoteCompass (http://votecompass.com/), a widely used interactive electoral literacy application.

candidate, and the assignment must be (roughly) balanced over the 5 candidates). Since integer variables are needed to ensure only 5 candidates are selected, we first solve the *relaxed problem* using DCA (with 24M matching variables and 20 selection variables) limiting ourselves to 400 cells. These cells are then used to solve the MIP with integrality re-imposed to ensure no fractional candidates are selected.

DCA requires 423s. for 400 iterations, with an anytime profile like those above (e.g., by iteration 100, no improvement is more than 0.1% of the first iteration). Interestingly, the cells produced using the LP work extremely well with the MIP—enforcing integrality causes only 0.56% loss in total voter satisfaction (MIPs solve in well under 0.5s).[7] This suggests using DCA on relaxed problems may be useful for generating abstractions into which strict combinatorial/integrality constraints can be re-introduced.

Concluding Remarks

We have developed DCA, a dynamic segmentation method for solving general "assignment" problems over large populations, modeled as LPs. These allow one to solve problems much larger than those that can be tackled using standard methods. DCA offers solution quality guarantees; fast, distributable computation that allows extreme scalability; excellent anytime performance; and strong support for real-time scenario analysis and reoptimization.

A number of important directions remain. These include incorporating richer response models in MMMOPs and methods for solving the resulting more complex programs (e.g., fractional programming can be used to model slates of offers that influence or interfere with one another using random utility or other models of influence, or using Markov decision processes to model cumulative, sequential influences). Apart from the heuristic use of DCA for MIPs as described here, we are especially interested in developing full branch-and-price algorithm for large MIPs. Further theoretical and empirical analysis of our methods is critical; we are especially interested in potential theoretical guarantees for our method for handling multiple LPs (e.g., different constraints or objectives). We are also exploring connections to recent work on abstracting LPs using symmetries (Mladenov, Ahmadi, and Kersting 2012) and (heuristic) distributed matching methods (Manshadi et al. 2013).

Acknowledgement's. This work was conducted in part at Granata Decision Systems. We thank the reviewers for their helpful comments, and Clifton van der Linden and the Vote-Compass team for making their dataset available.

Appendix: Pricing of Columns

We noted that the compressed LP for the current set of cells does not contain contact limit constraints on new cells that might be generated by splitting an existing cell. If we want to split cell k into two cells k_1 and k_2, we don't have contact limit constraints on the new cells, hence no dual values for these constraints (which play a role in determining the prices for any of k_1 or k_2's columns). We can add the following constraints to the compressed LP:

$$\sum_{j,h} x^k_{jh} \frac{z_{k^1}}{z_k} \leq L z_{k^1} \tag{17}$$

$$\sum_{j,h} x^k_{jh} \frac{z_{k^2}}{z_k} \leq L z_{k^2} \tag{18}$$

These express that the approaches assigned to cell k should not exceed the capacity of (new) cells k_1 and k_2, under the assumption that these approaches are allocated uniformly at random (hence, to the cells in proportion to their relative size, i.e., $\frac{z_{k^1}}{z_k}$ and $\frac{z_{k^2}}{z_k}$). These constraints are both multiples of Constraint 9, hence do not impact the solution of the compressed LP. Furthermore, their redundancy implies that some optimal solution of the dual of this extended LP is equivalent to the solution of dual of the compressed LP with dual values zero for to these new constraints. This means that reduced costs for any new "split cell" column can be determined without including these constraints (i.e., using dual values from the compressed LP itself).

Appendix: Convergence with Only Contact Limits

In practice, DCA must work with a restricted splitting language, which limits the cell splits that can be considered (otherwise, the number of splits is exponential in the number of customers). However, if one could split a cell arbitrarily finely, even using only binary splits, DCA will converge to an optimal solution for the uncompressed LP when we have only contact limits on customers (and no budget or lead limits on campaigns). We provide an informal argument here.

Assume an MMMOP with only contact limits of the form Eq. 2, and let $\mathbf{x}^*$ be its optimal solution with objective value z^*. For ease of exposition, we suppose the contact limit is $L = 1$. Let $\overline{x}$ be the optimal solution of a compressed LP with cells K, with objective value $\overline{z}$. Suppose this solution is not optimal for the original uncompressed LP, so $\overline{z} < z^*$. This means there must be some customer $i \in k$, for some cell k, such that $\overline{x}^k_{jh} < x^*_{ijh}$ (otherwise the objective values would be identical). For some such i, we must have $v^k_{jh} \leq v_{ijh}$ (again, otherwise we could not have $\overline{z} < z^*$). If we split cell k into cells $\{i\}$ and $k \setminus \{i\}$, the optimal solution with these new cells replacing k must be greater than $\overline{z}$ (e.g., the assignment that sets $x^{\{i\}}_{jh} = x^*_{ijh}$ and $x^{k\setminus\{i\}}_{jh} = \overline{x}^k_{jh}$, and is otherwise identical to $\overline{\mathbf{x}}$, must have greater objective value, and is feasible given only contact limits). By the properties of column generation on the extended LP consisting of these additional cell variables, the reduced cost of $x^{\{i\}}_{jh}$, hence the score of cell $\{i\}$, must be positive.

Appendix: Proof of Error Bound

We first outline a simple reformulation of our compressed LP, then prove our main error bound. Our compressed LP (8) above can be reformulated using variables $\dot{x}^k_{jh}$ denoting the

[7] A variant of the problem *without balance constraints* (Chamberlin and Courant 1983) can be approximated with a greedy algorithm (Lu and Boutilier 2011). DCA generates cells which, when used for the MIP, produce solutions within 0.38% of the value of the greedy solution (and 0.43% of the LP value).

total number of customers from cell k assigned approach (j,h) rather than the fraction. Allowing these variables to take fractional values, the following LP is obviously equivalent to LP (8), taking $x^k_{jh} = \dot{x}^k_{jh}/z_k$:

$$\max_{\dot{x}^k_{jh}} \quad \sum_{k,j,h} x^k_{jh}(v^k_{jh} - u_{jh}) \tag{19}$$

$$\text{s.t.} \quad \sum_{j,h} \dot{x}^k_{jh}/z_k \leq L \qquad \forall k \leq K \tag{20}$$

$$\sum_{k,h} \dot{x}^k_{jh} u_{jh} \leq B_j \qquad \forall j \in \mathcal{C} \tag{21}$$

$$\sum_{k,j,h} \dot{x}^k_{jh} u_{jh} \leq B \tag{22}$$

$$\sum_{k,h} \dot{x}^k_{jh} \leq L_j \qquad \forall j \in \mathcal{C} \tag{23}$$

$$\sum_{k,j} \dot{x}^k_{jh} \leq L_h \qquad \forall h \in \mathcal{H} \tag{24}$$

$$x^k_{jh} \geq 0. \qquad \forall i \in \mathcal{S}, j \in \mathcal{C}, h \in \mathcal{H} \tag{25}$$

We call LP (19) the *total formulation* of our compressed LP, and refer to LP (8) as the *fractional formulation*. While a trivial transformation, this will prove to be convenient in our derivation of approximation bounds. Since the fractional and the total formulations have identical optimal objective values—say z_f and z_t, respectively—and identical constraint vectors $\mathbf{b}$, they have the same dual solutions.

Zipkin (1980) analyzes the quality of the solution of LPs in which groups of variables are aggregated (and each aggregate variable corresponds to the sum of its constituent variables) and the solution of the aggregated LP is applied to the original using a *fixed weight disaggregation*. More precisely, suppose we have an LP with n variables $x_1, \ldots, x_n$, with A and $m \times n$ matrix:

$$z^* = \max \quad \mathbf{cx} \tag{26}$$

$$\text{s.t.} \quad A\mathbf{x} \leq \mathbf{b} \tag{27}$$

$$\mathbf{x} \geq 0. \tag{28}$$

Let $\sigma = \{S^1, \ldots S^k\}$ be a partitioning of the variables $\mathbf{x}$, with $|S^k| = n_k$. Let non-negative n_k-vector $g^k \geq 0$ be a *disaggregation vector* for partition S^k, with $\sum_i g^k_i = 1$. Letting A^k (resp., $\mathbf{c}^k$) be the submatrix of A (resp., subvector of $\mathbf{c}$) defined over variables in S^k, define $\overline{A}^k = A^k g^k$, $\overline{\mathbf{c}}^k = \mathbf{c}^k g^k$, $\overline{A} = (\overline{A}^1, \ldots, \overline{A}^K)$ and $\overline{\mathbf{c}} = (\overline{\mathbf{c}}^1, \ldots, \overline{\mathbf{c}}^K)$. Let $\overline{\mathbf{x}}$ for a vector of K variables (one per partition). The *aggregate LP* induced by σ, g is:

$$\overline{z} = \max \quad \overline{\mathbf{c}}\overline{\mathbf{x}} \tag{29}$$

$$\text{s.t.} \quad \overline{A}\overline{\mathbf{x}} \leq \mathbf{b} \tag{30}$$

$$\overline{\mathbf{x}} \geq 0. \tag{31}$$

This LP reduces the problem to one involving k (aggregated) variables. We use *origLP* and *aggrLP* to denote these two LPs, respectively.

The solution of *aggrLP* can be transformed into one for *origLP* by setting $x_i = g^k_{(i)}\overline{\mathbf{x}}^k$ for each $i \leq n$, where S^k is the partition containing x_i and (i) denotes the index of x_i within S^k. If $\overline{\mathbf{x}}$ is feasible for *aggrLP*, then its fixed weight disaggregation $\mathbf{x}$ is feasible for *origLP* and $\mathbf{cx} = \overline{\mathbf{c}}\overline{\mathbf{x}}$.[8]

Zipkin derives the following bound on the solution quality using the dual values for *aggrLP*:

$$z^* \leq \overline{z} + \sum_j [c_j - \mathbf{d} \cdot A_j]^+ \cdot u_j \tag{32}$$

Here $\mathbf{d}$ is the vector of dual variables obtained in the solution of *aggrLP*, A_j is the jth column of constraint matrix A, and u_j is any explicit or derivable upper bound on variable x_j in *origLP*. Here $[\cdot]^+$ denotes $\max(\cdot, 0)$.

The total formulation of our problem, LP (19), is an aggregated LP in Zipkin's sense—with one major difference outlined below—where for each approach (j,h) and cell k we aggregate variables x_{ijh}, for all $i \in k$, into x^k_{jh}, and use a fixed weight disaggregation (i.e., the uniform weighting). The difference is that we do not maintain constraints over individual customer contact limits, but aggregate those (i.e., aggregate rows) across cells as well to ensure further compression. We can use Zipkin's bounds, but must adapt the proofs to account for the fact that we aggregate rows as well. Specifically, our compressed problem does not have dual values for all constraints in *origLP* as required by Zipkin's bounds. Our proof shows how to derive such dual values directly from the solution of our compressed LP.

Theorem 1. *Let $\tilde{V}$ be the optimal objective value of the compressed LP (8) using segments S generated using DCA, and let V^* be the optimal objective value of the uncompressed LP (1). Let $\mathbf{c}_{ijh}$ be x_{ijh}'s column in LP (1). Finally, let $\boldsymbol{\pi}$ be the vector of dual values obtained when solving compressed LP (8), and $\widehat{\boldsymbol{\pi}}$ be the* extended dual vector *with the value π_k for each $k \in S$ replaced by the* set *of values $\pi_i = \pi_k/z_k$ for all $i \in k$. Then:*

$$V^* \leq \tilde{V} + \sum_{k \in S} \sum_{jh} \sum_{i \in k} [(v_{ijh} - u_{jh}) - \mathbf{c}_{ijh}\widehat{\boldsymbol{\pi}}]^+ \cdot L.$$

Proof. To simplify notation, we ignore channels h and assume variables x_{ij} in the original MMMOP; with channels, we simply duplicate each j with pairs (j,h).

We consider four different LPs:

- LP, our original uncompressed LP (1);
- LPF, the fractional formulation of our compressed LP (8);
- LPT, the total formulation of our compressed LP (19);
- LPE, an *extended* version of LPT in which we aggregate columns as in LPT, but do not aggregate rows, providing a direct Zipkin-style aggregation.

Let z denote the optimal value of LP, x its optimal solution, π the vector of dual values at the optimal solution, c its vector of objective coefficients, A and b its constraint matrix and vector. Let z_F, z_T, and z_E denote the same quantities for LPF, LPT, and LPE, respectively (and similarly for the other terms x, π, etc.). Note that $c_E = c_T$, $b_E = b$, $b_T = b_F$, and as observed above $z_F = z_T$ and $\pi_F = \pi_T$.

[8]This does not imply that *aggrLP* is feasible, even if *origLP* is, since the aggregation restricts solutions to a low-dimensional subspace of the original feasible set.

Part of Zipkin's bound Eq. 32 can be derived using LPE:

$$
\begin{aligned}
z &= cx && (33)\\
&\leq cx + \pi_E(b - Ax) && (34)\\
&= \pi_E b + (c - \pi_E A)x && (35)\\
&= z_E + (c - \pi_E A)x && (36)
\end{aligned}
$$

where the last equality holds by strong duality. From this the bound, one can derive bound Eq. 32 by some algebraic manipulation. Our difficulty is that we have not explicitly solved LPE so do not have dual values π_E. However, it is not hard to see that LPE is identical to LPT with the additional of redundant constraints—indeed, it consists of LPT with, for each cell k, one copy of k's contact limit constraint Eq. (20) for each customer $i \in k$. This means that $z_E = z_T$. Furthermore, the optimal solution π_T of the dual of LPT can be transformed into an optimal solution π_E of the dual of LPE by *any* (non-negative) division of the dual value $\pi_T(k)$ (for the contact constraint on cell k) among the dual values $\pi_E(i), i \in k$. In particular, we can create a dual solution π_E by setting $\pi_E(i) = 1/z_k \pi_T(k)$ for every $i \in k$. (We $\pi_E = \pi_T$ for all other constraints.)

We plug π_E into Eq. 32, and note that:

- We have an upper bound of L on the optimal values of any x_{ij} in LP, which plays the role of u_j in Eq. 32;
- V^* in the statement of the theorem is simply z^* in Eq. 32 and z in the argument above;
- $\tilde{V}$ in the theorem statement is z_F in the argument;
- and LPT and LPF have identical dual solutions, π_T and π_F (as observed earlier), so the extended dual vector π_E can be defined using π_T instead of π_F.

The theorem follows immediately. ■

References

Barnhart, C.; Johnson, E. L.; Nemhauser, G. L.; Savelsbergh, M. W. P.; and Vance, P. H. 1998. Branch-and-price: Column generation for solving huge integer programs. *Operations Research* 46(3):316–329.

Chamberlin, J. R., and Courant, P. N. 1983. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *The American Political Science Review* 77(3):718–733.

Louviere, J. J.; Hensher, D. A.; and Swait, J. D. 2000. *Stated Choice Methods: Analysis and Application*. Cambridge: Cambridge University Press.

Lu, T., and Boutilier, C. 2011. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11)*, 280–286.

Lübbecke, M. E., and Desrosiers, J. 2005. Selected topics in column generation. *Operations Research* 53(6):1007–1023.

Manshadi, F. M.; Awerbuch, B.; Gemulla, R.; Khandekar, R.; Mestre, J.; and Sozio, M. 2013. A distributed algorithm for large-scale generalized matching. *Proceedings of the VLDB Endowment* 6(9):613–624.

Mladenov, M.; Ahmadi, B.; and Kersting, K. 2012. Lifted linear programming. In *Fifteenth International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, 788–797.

Monroe, B. L. 1995. Fully proportional representation. *The American Political Science Review* 89(4):925–940.

Puchinger, J.; Raidl, G. R.; and Pferschy, U. 2010. The multidimensional knapsack problem: Structure and algorithms. *INFORMS Journal on Computing* 22(2):250–265.

Savelsbergh, M. 1997. A branch-and-price algorithm for the generalized assignment problem. *Operations Research* 45(6):831–841.

Shrivastava, N.; Buragohain, C.; Agrawal, D.; and Suri, S. 2004. Medians and beyond: New aggregation techniques for sensor networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys-04)*, 239–249.

Walsh, W. E.; Boutilier, C.; Sandholm, T.; Shields, R.; Nemhauser, G.; and Parkes, D. C. 2010. Automated channel abstraction for advertising auctions. In *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 887–894.

Zaharia, M.; Chowdhury, M.; Franklin, M. J.; Shenker, S.; and Stoica, I. 2010. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing*, 1–7.

Zipkin, P. H. 1980. Bounds on the effect of aggregating variables in linear programs. *Operations Research* 28(2):403–418.