End-to-End Inventory Prediction and Contract Allocation for Guaranteed Delivery Advertising

Wuyang Mao
wuyang.mwy@alibaba-inc.com
Alibaba Group
Beijing, China

Chuanren Liu*
cliu89@utk.edu
The University of Tennessee
Knoxville, United States

Yundu Huang
yundu.hyd@alibaba-inc.com
Alibaba Group
Beijing, China

Zhonglin Zu
zhonglin.zuzl@alibaba-inc.com
Alibaba Group
Hangzhou, China

M Harshvardhan
harshvar@vols.utk.edu
The University of Tennessee
Knoxville, United States

Liang Wang
Bo Zheng
liangbo.wl@alibaba-inc.com
bozheng@alibaba-inc.com
Alibaba Group
Beijing, China

ABSTRACT

Guaranteed Delivery (GD) advertising plays an essential part in e-commerce marketing, where the ad publisher signs contracts with advertisers in advance by promising delivery of advertising impressions to fulfill targeting requirements for advertisers. Previous research on GD advertising mainly focused on online serving yet overlooked the importance of contract allocation at the GD selling stage. Traditional GD selling approaches consider impression inventory prediction and contract allocation as two separate stages. However, such a two-stage optimization often leads to inferior contract allocation performance. In this paper, our goal is to reduce this performance gap with a novel end-to-end approach. Specifically, we propose the Neural Lagrangian Selling (NLS) model to jointly predict the impression inventory and optimize the contract allocation of advertising impressions with a unified learning objective. To this end, we first develop a differentiable Lagrangian layer to backpropagate the allocation problem through the neural network and allow direct optimization of the allocation regret. Then, for effective optimization with various allocation targets and constraints, we design a graph convolutional neural network to extract predictive features from the bipartite allocation graph. Extensive experiments show that our approach can improve GD selling performance compared with existing two-stage approaches. Particularly, our optimization layer can outperform the baseline solvers in both computational efficiency and solution quality. To the best of our knowledge, this is the first study to apply the end-to-end prediction and optimization approach for industrial GD selling problems. Our work has implications for general prediction and allocation problems as well.

*Corresponding Author

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599332

CCS CONCEPTS

• **Information systems** → **Online advertising**.

KEYWORDS

Guaranteed delivery, Contract allocation, End-to-end optimization

ACM Reference Format:
Wuyang Mao, Chuanren Liu, Yundu Huang, Zhonglin Zu, M Harshvardhan, Liang Wang, and Bo Zheng. 2023. End-to-End Inventory Prediction and Contract Allocation for Guaranteed Delivery Advertising . In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3580305.3599332

1 INTRODUCTION

Guaranteed Delivery (GD) advertising plays a pivotal role in online advertising and marketing platforms. Particularly, GD advertisers sign ad contracts with the publisher months/weeks before targeting dates to secure desired inventory of advertising impressions in advance. The contracts specify impression targets such as crowd, frequency, city, channel, and the amount of impressions. Here, impressions are ads being displayed to users and inventory consists of impression opportunities that can be sold to advertisers. Inventory overselling will result in under delivery (i.e., delivery of impressions less than targeting amount) and cause penalty for the publisher, while low usage rate of the impression inventory could also damage advertising revenue for the publisher. As shown in Figure 1, advertisers can query maximum available inventory for future advertising campaigns under certain delivery requirements on GD selling system before signing ad contracts. High-quality impressions tend to be sold out quickly for large-scale GD market places; and even a few percent improvement of inventory usage rate can serve more advertisers and increase the publisher revenue significantly. Therefore, the real-time GD selling system needs to return accurate inventory estimation and optimal contract allocations.

Recent work in GD advertising has focused on developing efficient allocation methods for real-time dispatching of user traffic to ad impressions, where the GD allocation is considered as a constrained optimization or bi-graph matching problem. Such

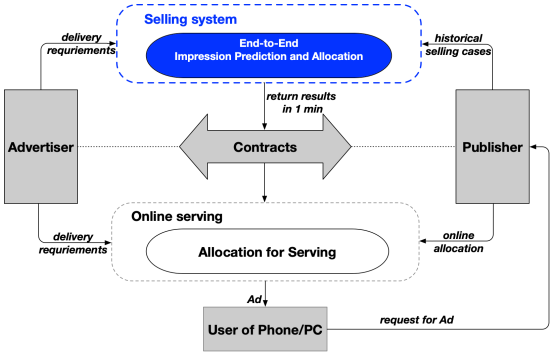

Figure 1: The system architecture for Guaranteed Delivery (GD) advertising. Our model supports the selling system, when advertisers and the publisher sign new GD contracts.

approaches, however, cannot meet all the operational requirements of the GD selling system as follows:

First, most studies [3, 5, 6, 10, 13] on GD advertising have only focused on online serving (as shown in Figure 1), though publisher needs the estimated predictive allocations when signing contracts with advertisers in advance. However, the theoretical models for online serving cannot fulfill the contracts if the signed target impressions exceed the impression inventory limit. The unfulfilled contracts will result in unsatisfactory marketing performance for advertisers and revenue loss for the publisher. Indeed, the publisher revenue is largely determined when GD contracts are singed during the selling period. To address these issues, we need to develop an intelligent decision support system for GD selling tasks.

Second, prior studies have not investigated the GD selling problem in an end-to-end manner. For instance, Zhang et al. [25] implemented an impression forecasting model and an inventory allocation method in a two-step framework. Although conducting impression forecasting followed by inventory allocation is straightforward, the forecasting uncertainty may impair performance in the contract allocation step. Two-stage methods assume that lower error of impression forecasting can automatically translate to better allocation quality. However, empirical evidences for this assumption is inconclusive in complex decision scenarios. Meanwhile, several recent studies have demonstrated that end-to-end predictive optimization can outperform two-stage methods on multiple tasks [8, 17, 22]. Integrating inventory prediction and contract allocation has great potential to improve performances of GD selling systems.

Third, existing end-to-end learning-based optimization framework like quadratic programming task loss (QPTL) [22] or interior point-based approach (IntOpt) [17] may not be directly applied for the GD selling problem. Particularly, GD selling problems have to incorporate a large number of constraints which can result in significant solution errors or infeasible memory requirement with common differentiable Lagrangian computing procedures. Our GD selling system must be designed with an efficient differentiable Lagrangian solver with less memory cost. Moreover, the learning-based optimization approach relies on effective capture of necessary features characterizing the optimization task. For GD selling systems, the solution must extract dynamic features from advertising contracts based on their complex connections, while previous studies are limited to static problems [8].

In this paper, we conduct inventory prediction and contract allocation in an end-to-end manner that can support GD selling tasks. Specifically, we propose the Neural Lagrangian Selling (NLS) method for large-scale and real-time GD selling problems. By analyzing Karush–Kuhn–Tucker (KKT) conditions and designing gradient descent procedures, we design a customized Lagrangian optimization layer to make GD selling problems differentiable and compatible with effective learning-based optimizers. The first benefit of our optimization layer is that only efficient matrix multiplication is necessary in the forward pass of the optimization iterations, while computationally expensive matrix factorization is involved in prior methods such as QPTL and IntOpt. Further, we adopt graph convolutional networks (GCNs) [15, 24] to capture contextual features and enable NLS to fit dynamic optimization objective and constraints in real-world GD selling systems. Finally, as shown in our results, we can effectively choose hyperparameters and achieve improved allocation error with our customized solver.

Our contributions can be summarized as follows:

1. We provide new insights into GD selling problems. Leveraging the power of deep learning and end-to-end optimization, our data-driven Neural Lagrangian selling (NLS) framework directly minimizes the final loss of advertising selling problem.

2. We derive a differentiable Lagrangian layer for inventory allocation problems. The Lagrangian layer can process general allocation constraints and has relatively low computational complexity for large-scale problems.

3. To tackle the complex and dynamic constraints of all advertisers, we design a flexible GCN module based on the GD allocation bipartite graph. With minimal adaption, our framework can be extended to other uncertain bi-graph matching problems.

4. With respect to prediction and allocation performances, our NLS significantly outperforms two-stage methods and alternative end-to-end methods. Our approach achieves better advertising delivery rate while simultaneously improving inventory rate, leading to increased GD revenue.

2 RELATED WORK

Guaranteed delivery advertising. GD advertising has inspired many research developments in recent years. Previous studies mainly focus on how to obtain optimal dispatching solution at serving stage. Chen et al. [5] proposed a compact allocation plan to match the ads and impressions. Bharadwaj et al. [3] developed a dual-based optimal algorithm using coordinate descent to approximate the optimal solution. Hojjat et al. [13] leveraged pattern learning to solve delivery optimization problem under reach and frequency requirements. Recently, Fang et al. [10] and Cheng et al. [6] investigated distributed serving allocation problem on user level. Generally, these methods are developed for the serving stage given a set of contracts, yet they cannot be directly adopted for GD selling and contract allocation tasks. For the selling optimization, Zhang et al. [25] designed a Lagrangian dual method, where prediction and allocation were solved as two steps separately. In this paper, we design an end-to-end solution by integrating inventory prediction and contract allocation into a unified learning problem.

End-to-end predictive optimization. Several recent studies have investigated the fusion of predictive algorithms and optimization problems, where a promising paradigm is to differentiate optimization layers with neural networks. Amos and Kolter [2] introduced a differentiable layer for Quadratic Programming (QP) optimization by differentiating the Karush–Kuhn–Tucker (KKT) conditions. Donti et al. [7] introduced the task-based end-to-end training process for QP problem. End-to-end training of Linear Programming (LP) problems have been studied by adding quadratic regularization terms [2, 22]. An interior-point approach (IntOpt) [17] was proposed with the homogeneous self-dual of LP problems to obtain backward gradients. Guler et al. [12] proposed a divide and conquer algorithm for *predict + optimize* with non-convex problems. This paper develops tailored predictive optimization solutions for GD selling tasks with dynamic objective and constraints, which haven't been extensively investigated in previous studies.

Representation learning for bipartite graph. The learning-based optimization relies on informative features, where learning representations of the GD bipartite graph is essential. To this end, Graph Neural Networks (GNNs) were demonstrated to be flexible and effective to process general graph structures [11, 21], and Graph Convolutional Networks (GCNs) were proposed to learn graph features directly and efficiently [15, 24]. For instance, Fan et al. [9] and Chen et al. [4] adopted GCNs to model bipartite graphs in recommendation systems. Inspired by these studies, we also adopt graph convolutional design to develop a customized representation learning network for the GD bipartite graph in our problem.

3 PROBLEM DEFINITION

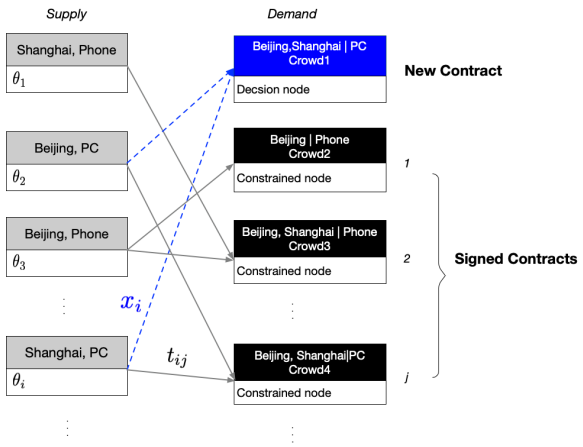

Figure 2: The Bi-graph of contract allocation problem. The supply nodes are impression inventory units. The demand nodes are GD contracts specifying the amount of impressions from different supply nodes.

The predictive allocation problem is illustrated in Figure 2. On the left-hand side, impression inventory $\theta \in \mathbb{R}^m$ represents the impression of m supply nodes where each supply node is the smallest unit for inventory forecasting and allocation. For example, one supply node could represent a specific combination of *city, app/channel, and device* to be targeted by various ads contracts. On the right-hand side, the demand nodes represent the contracts from advertisers.

Given a new contract, our task of allocation optimization is to maximize the allocation of impression supply to the new contract under customized advertiser constraints, without impacting the supply to previously signed contracts. We consider the new contract that queries about the rest advertise inventory as the decision node, and other contracts targeting the same supply nodes as the constrained nodes. We use x_i to denote the allocation rate from supply node i to the decision node and t_{ij} to denote the allocation rate from supply node i to the constrained node j.

3.1 Impression Forecasting

In traditional two-stage predictive allocation method, a predictive model (usually based on neural networks) is first trained for impression forecasting. Let θ represent the impression inventories of the supply nodes and $\{(z_i, \theta_i) : i = 1, 2, \cdots\}$ represent the training samples, where the features z_i consist of historical records and contextual information such as week, month, holidays, etc. The model $\theta \approx g(z; \omega)$ is trained via optimizing the parameters ω to minimize the prediction loss:

$$Loss = \|\theta - g(z; \omega)\|_2^2. \tag{1}$$

3.2 Inventory Allocation

The inventory allocation problem is formulated as:

$$\begin{aligned} \max_{x,t,u} \quad & (I \odot \theta)^\top x - p^\top u, \\ s.t. \quad & Gx + Bt - u \le h, \\ & u \ge 0, \\ & x, t \in [0, 1]. \end{aligned} \tag{2}$$

Specifically, we optimize the allocation decisions in x, where x_i represents the allocation ratio of the supply inventory θ_i to the decision demand of the new contract. Note that, $\theta \in \mathbb{R}^m$ contains the traffic inventory of all supply nodes, and $I \in \{0, 1\}^m$ represents whether the supply nodes are relevant to the decision demand. The optimization objective maximizes $(I \odot \theta)^\top x$, i.e., the total impressions from all the relevant supply nodes. We also need to reallocate the impression for the constrained demands of previous contracts, i.e., t_{ij}. For the sake of simplicity, the allocation matrix $T = (t_{ij}) \in \mathbb{R}^{m \times n}$ is flattened to a vector $t \in \mathbb{R}^{m*n}$, where n is the number of constraint nodes. The matrix $G \in \mathbb{R}^{k \times m}$, $B \in \mathbb{R}^{k \times (m*n)}$, and vector $h \in \mathbb{R}^k$ are used to define a set of allocation constraints in $Gx+Bt \le h$. To ensure we have a feasible solution with these constraints, we penalize $p^\top u$ in the objective function, where $p \in \mathbb{R}^k$ is the penalty coefficients (i.e., importance of different constraints) and $u \in \mathbb{R}^k$ is the slack variables for the constraints. These constraints can ensure both generality and feasibility of the optimization problem by encoding all the contract requirements, such as crowd, frequency, impression demand, and inventory capacity.

3.3 Two-Stage and End-to-End Allocation

The traditional two-stage approach first estimates the impression inventory by minimizing the prediction loss, and then solves the

inventory allocation problem using exact solvers to optimize allocation decisions. In this paper, our goal is to develop an end-to-end allocation approach by minimizing the allocation regret:

$$Regret = \|(I \odot \hat{\theta})^\top \hat{x} - (I \odot \theta)^\top x\|_2^2, \tag{3}$$

where $\hat{\theta} = g(z; w)$ is the predicted inventories and $\hat{x}$ is the corresponding allocation decisions. The end-to-end approach can learn from historical decision cases considering both impression prediction and inventory allocation. The regret definition implies that:

$$\frac{\partial Regret}{\partial \omega} = 2 \cdot \frac{\partial (I \odot \hat{\theta})^\top \hat{x}}{\partial \omega} ((I \odot \hat{\theta})^\top \hat{x} - (I \odot \theta)^\top x), \tag{4}$$

where

$$\begin{aligned} \frac{\partial (I \odot \hat{\theta})^\top \hat{x}}{\partial \omega} &= \frac{\partial (I \odot \hat{\theta})^\top}{\partial \omega} \hat{x} + (I \odot \hat{\theta})^\top \frac{\partial \hat{x}}{\partial \omega} \\ &= \frac{\partial (I \odot \hat{\theta})^\top}{\partial \omega} \hat{x} + (I \odot \hat{\theta})^\top \frac{\partial \hat{x}}{\partial \hat{\theta}} \frac{\partial \hat{\theta}}{\partial \omega}. \end{aligned} \tag{5}$$

Note that, $\frac{\partial \hat{\theta}}{\partial \omega}$ can be automatically computed for optimizing a deep learning model, and the challenge in directly minimizing the regret lies in backpropagating $\frac{\partial \hat{x}}{\partial \hat{\theta}}$. The gradients of the regret must be computed by differentiating the allocation optimization problem.

4 METHODOLOGY

Our approach is demonstrated in Figure 3. Specifically, the prediction module extracts the inventory related features. The prediction module can be implemented with the same neural network design as that in the two-stage approach. The GCN module extracts important information from the allocation bipartite graph. The Lagrangian solver module is the most challenging part, where we need to implement both forward and backward propagation of the inventory optimizer. Our approach is to relax the inventory allocation problem and derive the corresponding KKT (Karush–Kuhn–Tucker) conditions. We discuss more details in the rest of this section.

4.1 Lagrangian Dual Optimization

First, we derive a general algorithm suitable for the inventory allocation problem. Following previous works [10], we relax the allocation problem as a quadratic programming (QP) problem to maximize:

$$\ell(x, t, u) = (I \odot \theta)^\top x - p^\top u - \frac{\lambda}{2} \left(\|x\|_2^2 + \|t\|_2^2 \right), \tag{6}$$

subject to constraints:

$$\begin{aligned} Gx + Bt - u &\le h, \\ u &\ge 0, \\ x, t &\in [0, 1]. \end{aligned}$$

The corresponding dual problem is formulated as:

$$\min_{\alpha, \beta \ge 0} \max_{x,t,u} L(x, t, u, \alpha, \beta),$$

where the Lagrangian function is:

$$L(x, t, u, \alpha, \beta) = \ell(x, t, u) + \alpha^\top (h + u - Gx - Bt) + \beta^\top u. \tag{7}$$

According to the KKT conditions and solving the first-order derivatives $\frac{\partial L}{\partial x} = 0$, $\frac{\partial L}{\partial t} = 0$, and $\frac{\partial L}{\partial u} = 0$, we can derive the primal solutions:

$$\begin{aligned} x &= \min(\max(\frac{I \odot \theta - G^\top \alpha}{\lambda}, 0), 1), \\ t &= \min(\max(\frac{-B^\top \alpha}{\lambda}, 0), 1), \\ \alpha \odot (h + u - Gx - Bt) &= 0, \\ \beta \odot u &= 0, \\ \alpha + \beta &= p. \end{aligned} \tag{8}$$

These results describe the relationship between the primal decision variables and the dual decision variables. We can show that when $u > 0$, then $\beta = 0$ and $\alpha = p$; when $u = 0$, then the gradient of the dual variable α can be obtained as:

$$\begin{aligned} \frac{\partial L}{\partial \alpha} =& (I \odot \theta - \lambda x - G^\top \alpha) \frac{\partial x}{\partial \alpha} - (B^\top \alpha + \lambda t) \frac{\partial t}{\partial \alpha} \\ &+ (h - Gx - Bt), \end{aligned} \tag{9}$$

which further reduces to $\frac{\partial L}{\partial \alpha} = h - Gx - Bt$ given the KKT conditions. These results will be used in our end-to-end optimization algorithm for GD allocation using the gradient descent method.

In the Lagrangian problem, a proper selection of the relaxation hyperparameter λ is crucial. A value that is too small may cause instability in the convergence process, while a value that is too large may introduce errors. Considering the facts that $\theta x - \frac{\theta}{n} x^2$ and $-\frac{\theta}{n} t^2$ increase monotonically for $n \ge 2$ with respect to uni-variate $x, t \in [0, 1]$, our hyperparameter is chosen as $\lambda = \min(\theta)$.

4.2 Efficient Lagrangian Dual Layer

To solve the GD selling allocation problem with the Lagrangian algorithm, we integrate the Lagrangian dual optimization layer in the predictive deep neural networks as shown in Algorithm 1. Note that in Stage 3 we use the Adam gradient descent method [14], where b_1, b_2, μ, and $decay\_rate$ are learning rate hyperparameters.

Our Lagrangian dual layer can be flexibly embedded in any neural network structure and the back-propagation can be done automatically by deep learning frameworks such as TensorFlow [1] or Torch [19]. The computation of the optimization layer does not involve complex operations such as matrix inversion, and we can easily implement the optimization layer with batched input and parallel computing to improve training/inference efficiency.

4.3 GCN Module

To effectively learn the end-to-end model, we should extract meaningful features from the selling allocation problem. Particularly, the conditions for all GD contracts (e.g., the I in the objective function and G in the constraints) vary greatly for different cases. Therefore, we design the Graph Convolutional Network (GCN) module to capture adaptive features for diverse decision scenarios and improve accuracy and flexibility of the allocation optimization process.

In our problem formulation (shown in Figure 2), the number of impression supply nodes in the bipartite graph is represented by m, while the number of constraint demand nodes is n. The supply-constraint adjacency relationship is captured by the matrix $A \in \mathbb{R}^{m \times n}$. The supply-decision adjacency is represented by the

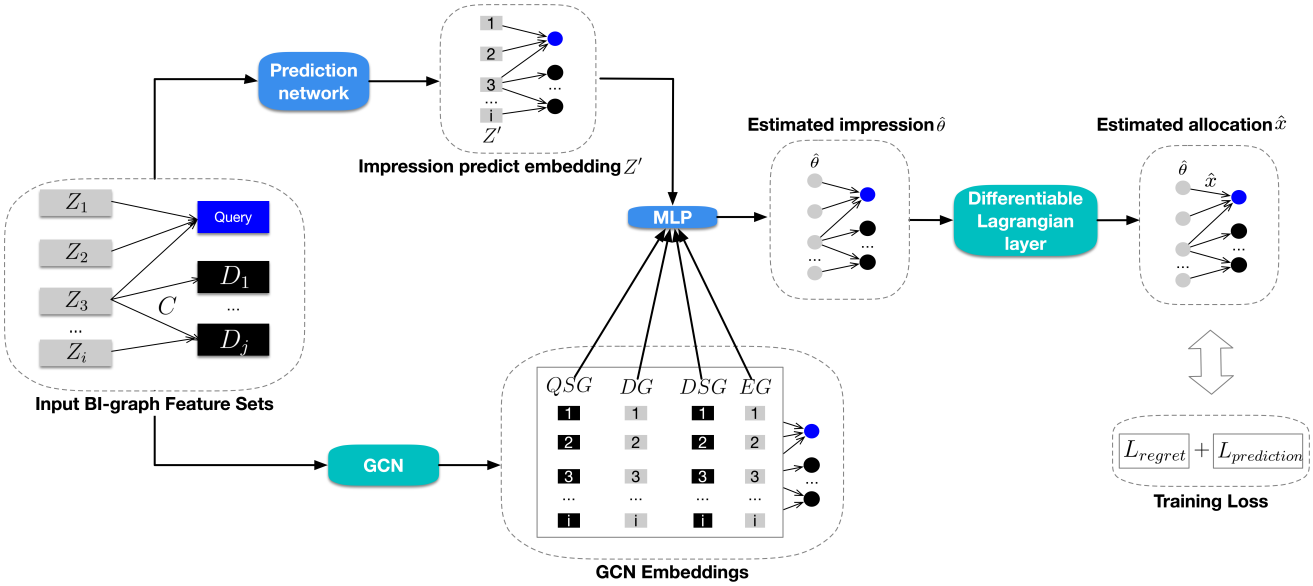

Figure 3: The overall architecture of the NLS model.

Algorithm 1 Forward Pass for Lagrangian Dual Layer (LDL)

Input: θ
Output: $x = LDL(\theta)$
Constants: G, B, h, p
Initialization: $\alpha = 0, mt = 0, vt = 0, \mu = 100, b_1 = 0.9, b_2 = 0.99, e = 10^{-9}, decay\_rate = 0.95$
$\lambda = \min(\theta)$
repeat
  Stage 1: Calculate original variable
    $x = \frac{I \odot \theta - G^\top \alpha}{\lambda}$
    $t = \frac{-B^\top \alpha}{\lambda}$
    $x = \min(0, \max(x, 1))$
    $t = \min(0, \max(t, 1))$
  Stage 2: Calculate dual variable gradient
    $\frac{\partial L}{\partial \alpha} = h - Gx - Bt$
  Stage 3: Update dual variable
    $mt = b_1 * mt + (1 - b_1) * \frac{\partial L}{\partial \alpha}$
    $vt = b_2 * vt + (1 - b_2) * (\frac{\partial L}{\partial \alpha} \odot \frac{\partial L}{\partial \alpha})$
    $\alpha = \min(\max(0, \alpha - \mu * \frac{mt}{\sqrt{vt+e}}, p))$
    $\mu = \mu * decay\_rate$
until Convergence or maximum iterations

vector $I \in \{0,1\}^m$. The input features for the GCN module are represented as follows, where f_1, f_2, f_3 are the feature dimensions:

- Supply node features $Z^0 \in \mathbb{R}^{m \times f_1}$ for supply nodes contain information such as holidays, dates, weeks, etc.
- Constraint node features $D \in \mathbb{R}^{n \times f_2}$ capture information such as targeting impressions and selling types.
- Edge features $C \in \mathbb{R}^{m \times n \times f_3}$ capture information such as lower bounds, upper bounds, and competition ratios between adjacent nodes and edges.

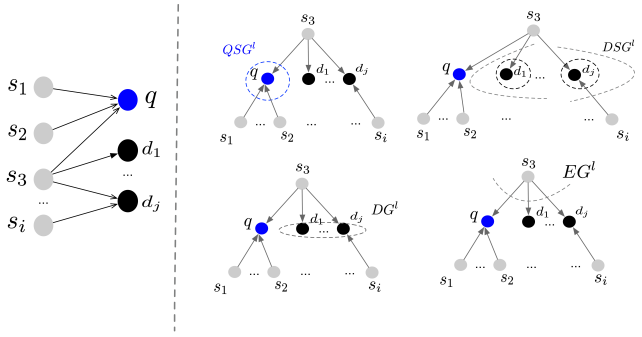

Figure 4: An illustration of the GCN feature extraction process for the supply node s_3.

It is nontrivial to simply apply existing GCN [23] networks to aggregate these information in our end-to-end process. To develop a suitable GCN operator, Figure 4 illustrates the the l-th GCN layer in our feature extraction process for the supply node s_3. Specifically, each layer consists of the following five steps, where W_Z, $(W^l_{QSG}, W^l_{DSG}, W^l_{DG}, W^l_{EG})$, $l = 1, \cdots, L$, represent the learnable parameters, and L is the number of layers:

(1) Extracting QSG (Query to Supply GCN) features: as our objective is to learn the available inventory of decision node, we aggregate the features(Z^{l-1}) of linked supply nodes into an embedding and broadcast this embedding to all corresponding supply nodes:

$$QSG^l = Broadcast(Relu(\frac{I^T Z^{(l-1)} W^l_{QSG}}{\sum_{i=0}^{m} A_{ij}})).$$

(2) Extracting DSG (Demand to Supply GCN) features: we first aggregate the supply features (Z^{l-1}) of the corresponding

constraint demands to obtain the impression supply embedding of the constraint nodes $\frac{A^T Z^{(l-1)}}{\sum_{i=0}^{m} A_{ij}}$, and then aggregate this embedding to the supply dimension to obtain:

$$DSG^l = Relu(\frac{A}{\sum_{j=0}^{n} A_{ij}} \frac{A^T Z^{(l-1)}}{\sum_{i=0}^{m} A_{ij}} W_{DSG}^l).$$

(3) Extracting DG (Demand GCN) features: we aggregate the original demand node features (D) to supply nodes to represent the competition for supply node impressions:

$$DG^l = Relu(\frac{ADW_{DG}^l}{\sum_{j=0}^{n} A_{ij}}).$$

(4) Extracting EG (Edge GCN) features: we aggregate the edge features (C) of each supply node to take into account bound conditions such as frequency control and crowd competition among edges that share supply nodes:

$$EG^l = Relu(\frac{\sum_{j=0}^{n}(A_{ij} \cdot C_{ij})}{\sum_{j=0}^{n} A_{ij}} W_{EG}^l).$$

(5) Finally, we concatenate $(QSG^l, DSG^l, DG^l, EG^l)$ to obtain:

$$Z^l = Relu((QSG^l, DSG^l, DG^l, EG^l) \cdot W_Z)$$

as the initial supply nodes features for the next layer.

4.4 End-to-End Allocation Optimization

The forward computation process of the overall end-to-end algorithm is shown in Algorithm 2. The process starts with the dense neural network DNN to extract features Z', which is then concatenated with the GCN embedding Z^L. This combined features are processed through a multilayer perceptron (MLP) with the Relu activation function to estimate the supply impressions $\hat{\theta}$. The estimated impressions ($\hat{\theta}$) are then input into our Lagrangian dual layer to produce the estimated allocation ratio $\hat{x}$. The final allocation regret can be minimized directly by this end-to-end process with the training loss:

$$loss = \|(I \odot \hat{\theta})^\top \hat{x} - (I \odot \theta)^\top x\|_2^2 + \epsilon \|\theta - \hat{\theta}\|_2^2. \quad (10)$$

The training loss consists of two parts: end-to-end allocation regret and prediction errors for inventory nodes. The hyperparameter ϵ can balance the weight between the two optimization objectives during the training process.

During the inference phase, our Lagrangian dual layer can directly compute the allocation decisions, so the inference process is the same as that in the training forward process. However, theoretically, if computational time requirements are met, the Lagrangian layer can be replaced with any exact linear/quadratic programming solver to obtain allocation solutions.

5 EXPERIMENTS

To assess the performance of our NLS end-to-end approach, we conduct extensive experiments utilizing both offline and online production data sets from a large GD advertising publisher. To the best of our knowledge, public and comprehensive data for studying GD selling tasks is unavailable, and we plan to release our data to ensure reproducibility of our results and facilitate future research on related topics.

Algorithm 2 Forward Pass for End-to-End Allocation Optimization

Input: Z^0, D, C
Output: x
Constant : G, B, h, p, A, I
$Z' = DNN(Z^0)$
for $l = 1, \cdots, L$ **do**
  $QSG^l = Broadcast(Relu(\frac{I^T Z^{(l-1)} W_{QSG}^l}{\sum_{i=0}^{m} A_{ij}}))$
  $DSG^l = Relu(\frac{A}{\sum_{j=0}^{n} A_{ij}} \frac{A^T Z^{(l-1)}}{\sum_{i=0}^{m} A_{ij}} W_{DSG}^l)$
  $DG^l = Relu(\frac{ADW_{DG}^l}{\sum_{j=0}^{n} A_{ij}})$
  $EG^l = Relu(\frac{\sum_{j=0}^{n}(A_{ij} \cdot C_{ij})}{\sum_{j=0}^{n} A_{ij}} W_{EG}^l)$
  $Z^l = Relu((QSG^l, DSG^l, DG^l, EG^l) \cdot W_Z)$
end for
$\hat{\theta} = Relu(MLP(Z^L, Z'))$
$x = LDL(\hat{\theta})$

5.1 Data Description

The data sets are summarized Table 1. We conduct experiments daily and split data records from each day into 10,000/1,000 sub-samples for training and testing, respectively.

Offline Dataset: We model online media impressions using a trigonometric function. The entire inventory is randomly divided into 100 supply nodes per selling case, with each node representing $city \times device$ impressions. The aim of the experiment with the offline dataset is to determine the feasibility of the end-to-end optimization theory. We compare the two-stage method and end-to-end approach in detail through experiments on three inventory allocation cases: full targeting(decision node targeting all supply nodes), single targeting(decision node targeting a single supply node), and random targeting(decision node targeting supply nodes randomly). We verify the NLS approach starting with simple problems, adding only basic constraints (overselling and underdelivery) to the offline inventory allocation cases. As illustrated in Table 1, the $supply \times demand$ dimension is 100×10 with only 110 constraints (100 overselling and 10 underdelivery).

Online Dataset: The experiments are performed on two online advertising selling datasets, the Pre-Video Ads (PVA) dataset and the Open-Screen Ads (OSA) dataset. The pre-video ads are delivered before an online video starts playing, while the open-screen ads are delivered when a mobile app is launched. Each supply node of PVA represents a combination of $Channel \times City \times Position$, while the supply node for OSA is $City \times App$. The corresponding selling problems for inventory allocation and allocation are more complex and dynamic compared to those in the offline datasets. Particularly, the dimension of constraints increases from $(m+n)$ to $(m*n+m+n)$, where m represents the supply nodes dimension and n represents the constraint demand nodes dimension as shown in Table 1.

Table 1: Summary of offline and online data sets.

Dimensions	Offline			Online PVA	Online OSA
	full targeting	single targeting	random targeting	real targeting	real targeting
$supply \times demand$	100×10	100×10	100×10	500×20	500×50
$constraints$	110	110	110	**10520**	**25550**

5.2 Baselines

We compare NLS with the following benchmark methods:

1) **Two Stage**: The aim of this study is to investigate if the end-to-end approach can reduce the optimization regret in solving the selling problem compared to the two-stage method [25]. To ensure a fair comparison, our prediction network module is kept consistent with that of the two-stage method, and no additional inventory-related features are added to our NLS.
2) **PF**: We implemented a Pure Fully-Connected (PF) [20] network as the baseline end-to-end approach. The purpose is to investigate if a simple black-box neural network can achieve better performance than other methods. The input features include concatenated impressions-related features and features of nodes in the GD bipartite graph.
3) **PPG**: The GCN [15, 24] networks have demonstrated significant advantages in solving graph-based prediction problems for many applications. Therefore we remove the Lagrangian layer from NLS and utilized GCN to predict allocation results. This allows us to compare our NLS with the Pure Prediction GCN (PPG) approach.
4) **PL**: The GCN module in NLS enables the learning of complex allocation cases. However, the Prediction Network + Lagrangian solver (PL) approach can also solve the end-to-end advertising selling problem without the GCN module. As another ablation study, we remove the GCN module from our NLS as a baseline.
5) **GCN+QPTL/GCN+InOpt (End-to-End)**: IntOpt[17] and QPTL[22] are two established differentiable LP solvers. We implemented these solvers on our problem and compared them with our Lagrangian layer in terms of solution quality and computational efficiency.

5.3 Evaluation Metrics

Due to the significant variation in impression inventory between different advertising selling problems, we evaluate the overall performance using the Normalized Deviation (ND) [16] metric in totally K selling problems. We denote $E2E_k = (I \odot \theta)^\top x$ to represent the final decision label calculated from observed impressions θ and allocation ratios x solved by the exact solver mindopt [18]. Similarly, $\widehat{E2E}_k (= I \odot \hat{\theta})^\top \hat{x}$ represents the estimated decision results computed from predicted impressions $\hat{\theta}$ and estimated allocation $\hat{x}$. Then, we define the following metrics:

(1) **End-to-End error:** We assess the performance of end-to-end selling allocation using the ND_{reg} (Regret Normalized Deviation) metric for different methods:

$$ND_{reg} = \frac{\sum_{k=0}^{K} \left| E2E_k - \widehat{E2E_k} \right|}{\sum_{k=0}^{K} |E2E_k|}.$$

(2) **First stage error:** The performance of the first stage impression prediction is evaluated using the ND_{pre} (Prediction Normalized Deviation) metric:

$$ND_{pre} = \frac{\sum_{k=0}^{K} |\theta_k - \hat{\theta_k}|}{\sum_{k=0}^{K} |\theta_k|}.$$

(3) **Second stage error:** The quality of the second stage allocation is evaluated using the ND_{allo} (Allocation Normalized Deviation) metric for various LP/QP solvers:

$$ND_{allo} = \frac{\sum_{k=0}^{K} \left| E2E_k - \widehat{AE_k} \right|}{\sum_{k=0}^{K} |E2E_k|},$$

where $\widehat{AE} = (I \odot \theta)^\top \hat{x}$ measures the performance in the second stage allocation using the observation inventory θ.

(4) **Publisher revenue:** We consider both the overselling penalty and loss from unsold inventory in our evaluation of publisher revenue, using the *ARPD* (Average Revenue Per Day) metric to measure the revenue performance:

$$ARPD = \frac{\sum_{k=0}^{K} (E2E_k - |E2E_k - \widehat{E2E_k}|) * Price_k}{nn},$$

where nn is the number of days and $Price_k$ stands for contract price.

(5) **Delivery rate:** Delivery Rate (DR) refers to the ratio of delivered impressions to the total guaranteed demand:

$$DR_k = \begin{cases} \frac{\widehat{E2E_k} - E2E_k}{\widehat{E2E_k}} & E2E_k < \widehat{E2E_k} \\ 1 & E2E_k \geq \widehat{E2E_k} \end{cases}.$$

It is desirable for DR to be as close to 100% as possible in order to fulfill customer contracts.

(6) **Usage rate:** Usage Rate (UR) represents the ratio of sold impressions to the total available impression inventory:

$$UR_k = \begin{cases} \frac{E2E_k - \widehat{E2E_k}}{E2E_k} & E2E_k > \widehat{E2E_k} \\ 1 & E2E_k \leq \widehat{E2E_k} \end{cases}.$$

A higher UR usually results in higher publisher revenue assuming the same delivery rate.

Table 2: Experimental Results on Offline Datasets

Methods	full targeting		single targeting		random targeting	
	ND_{pre}	ND_{reg}	ND_{pre}	ND_{reg}	ND_{pre}	ND_{reg}
Two Stage	$0.101_{\pm 0.003}$	$0.023_{\pm 2e-4}$	$0.101_{\pm 0.003}$	$0.125_{\pm 0.005}$	$0.101_{\pm 0.003}$	$0.045_{\pm 0.002}$
PF	$0.130_{\pm 0.010}$	$0.045_{\pm 0.005}$	$0.115_{\pm 0.008}$	$0.132_{\pm 0.010}$	$0.121_{\pm 0.010}$	$0.076_{\pm 0.007}$
PPG	$0.125_{\pm 0.010}$	$0.015_{\pm 1e-4}$	$0.127_{\pm 0.007}$	$0.112_{\pm 0.001}$	$0.135_{\pm 0.011}$	$0.036_{\pm 0.001}$
PL	$0.102_{\pm 0.001}$	$0.008_{\pm 1e-4}$	$0.103_{\pm 0.001}$	$0.113_{\pm 0.003}$	$0.101_{\pm 0.002}$	$0.047_{\pm 2e-4}$
NLS	$\mathbf{0.096}_{\pm 0.002}$	$\mathbf{0.007}_{\pm 2e-4}$	$\mathbf{0.097}_{\pm 0.001}$	$\mathbf{0.098}_{\pm 0.001}$	$\mathbf{0.095}_{\pm 0.001}$	$\mathbf{0.029}_{\pm 1e-4}$

5.4 Result Analyses

5.4.1 Offline Evaluation. The results of the offline evaluation are presented in Table 2. The two-stage results are noteworthy. The ND_{reg} is significantly lower than ND_{pre} in the full targeting case, indicating that the second-stage allocation task is easier when supply nodes are combined. In the case of single targeting, the selling problem is similar to the impression forecasting problem with the consideration of delivery constraints, and accordingly ND_{reg} is slightly higher than ND_{pre}. The ND_{reg} for random targeting falls between that of full targeting and single targeting as expected. Overall, **the two-stage results provide a glimpse into the potential of end-to-end learning**.

We highlight several interesting observations: 1) **PF**. The two stage method outperforms PF on all metrics, suggesting that a pure black box neural network is not sufficient for solving end-to-end selling problems. 2) **PPG**. PPG outperforms PF as GCN can learn information from the bipartite graph. Although the ND_{pre} of PPG is slightly higher than the two stage network, its ND_{reg} is significantly lower. However, PPG cannot be deployed in a production environment as it is difficult to guarantee allocation constraints without optimization solvers. 3) **PL**. PL has a lower ND_{reg} in both full targeting and single targeting cases compared to the two stage methods, but in the random targeting case, the model oscillates between different training samples with varying allocation objectives and constraints, leading to a worse random targeting ND_{reg}. However, compared to the PPG method, PL has a more stable ND_{pre}. 4) **NLS**. Finally, we can observe that NLS demonstrated a clear superiority over the two-stage method and other baseline end-to-end networks. **The improvement is most notable on the ND_{reg} metric where the NLS showed a relative decrease of 35.5% on random targeting case**. This highlights the effectiveness of the NLS in optimizing publisher revenue through high quality selling decisions. We can conclude that NLS combines the advantages of the GCN module and Lagrangian layer.

5.4.2 Online Evaluation. The findings from Table 3 demonstrate that the NLS method surpasses the two-stage method and other baseline end-to-end methods on both the PVA and OSA data sets. **The results specifically reveal NLS's exceptional performance in terms of decreasing ND_{reg} by** 39% **for the PVA dataset and that by** 56% **for the OSA dataset.** This highlights the advantage of NLS over two-stage methods in handling data sets with intricate and dynamic constraints. PF still performs the worst, while PPG performs better on the OSA data compared to the PVA data, indicating that larger selling problems have more optimization potential.

Table 3: Experimental results on two online data sets.

Methods	PVA		OSA	
	ND_{pre}	ND_{reg}	ND_{pre}	ND_{reg}
Two Stage	$0.069_{\pm 0.002}$	$0.068_{\pm 0.004}$	$0.067_{\pm 0.002}$	$0.132_{\pm 0.005}$
PF	$0.083_{\pm 0.015}$	$0.095_{\pm 0.020}$	$0.075_{\pm 0.015}$	$0.128_{\pm 0.021}$
PPG	$0.085_{\pm 0.005}$	$0.054_{\pm 0.002}$	$0.078_{\pm 0.003}$	$0.086_{\pm 0.004}$
PL	$0.065_{\pm 0.002}$	$0.061_{\pm 0.002}$	$\mathbf{0.065}_{\pm 0.001}$	$0.136_{\pm 0.004}$
NLS	$\mathbf{0.064}_{\pm 0.003}$	$\mathbf{0.041}_{\pm 0.001}$	$0.068_{\pm 0.003}$	$\mathbf{0.058}_{\pm 0.001}$

The PL method, on the other hand, does not have the ability to handle complex constraints.

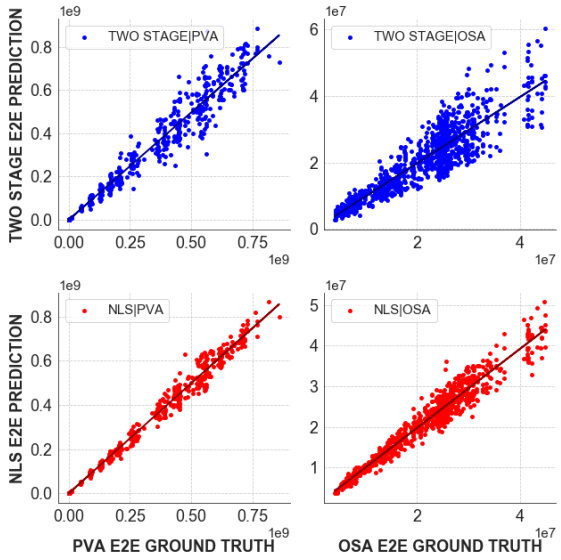

Figure 5: The E2E results for NLS and Two-Stage methods on PVA and OSA data sets. Our NLS has fewer outliers in comparison with two-stage methods.

Furthermore, the scatter plot in Figure 5 displays the $\widehat{E2E}$ results for NLS and two-stage methods on PVA and OSA datasets. The results suggest that NLS has fewer outliers compared to the two-stage methods, as NLS does not overfit the first stage prediction

Table 4: Comparison of NLS with other LP/QP solvers. Our NLS can handle complex constraints and achieve improved solutions.

Methods	Offline (random)			PVA			OSA		
	ND_{pre}	ND_{allo}	ND_{reg}	ND_{pre}	ND_{allo}	ND_{reg}	ND_{pre}	ND_{allo}	ND_{reg}
GCN + QPTL	$0.098_{\pm 0.008}$	$0.15_{\pm 0.02}$	$0.032_{\pm 0.005}$	-	-	-	-	-	-
GCN + IntOpt	$0.100_{\pm 0.003}$	$0.25_{\pm 0.03}$	$0.038_{\pm 0.005}$	$0.068_{\pm 0.006}$	$0.32_{\pm 0.04}$	$0.063_{\pm 0.002}$	$0.070_{\pm 0.008}$	$0.33_{\pm 0.04}$	$0.083_{\pm 0.006}$
NLS	$\mathbf{0.095}_{\pm 0.001}$	$\mathbf{1e\text{-}4}_{\pm 1e-5}$	$\mathbf{0.029}_{\pm 1e-4}$	$\mathbf{0.064}_{\pm 0.003}$	$\mathbf{7e\text{-}4}_{\pm 1e-5}$	$\mathbf{0.041}_{\pm 0.0031}$	$\mathbf{0.068}_{\pm 0.003}$	$\mathbf{6e\text{-}4}_{\pm 1e-5}$	$\mathbf{0.058}_{\pm 0.001}$

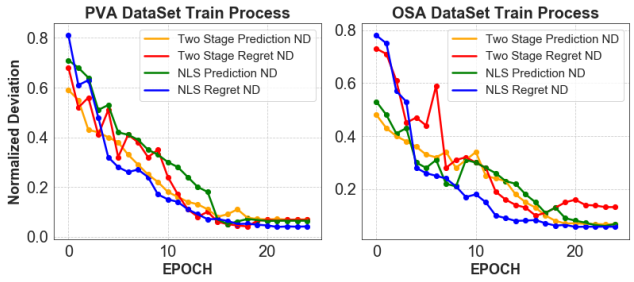

Figure 6: The training process of NLS and Two-Stage methods. Our NLS follows a more stable training trajectory.

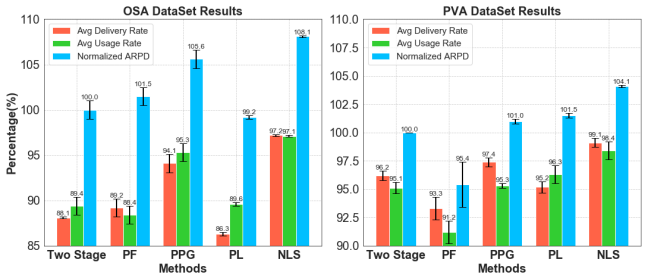

Figure 7: The revenue, delivery rate, and usage rate for different methods. Our NLS consistently performs better.

task as much. Figure 6 illustrates the training process of NLS and two-stage methods, supporting this theory. On the OSA dataset, the ND_{reg} of the two-stage methods increases after epoch 16, even though the first stage prediction ND_{pre} is still being optimized.

5.4.3 Revenue Analyses. Figure 7 compares the impact of different approaches on publisher revenue, with two-stage ARPD as the baseline. **NLS results in an 8.1% revenue increase on OSA and a 4.1% increase on PVA**. It also consistently outperforms other models in delivery rate and inventory usage rate, achieving 97.1% usage rate and 97.2% delivery rate on OSA, and 98.4% usage rate and 99.1% delivery rate on PVA.

5.4.4 Lagrangian Layer vs QPTL vs IntOpt. First, we evaluate second stage performance ND_{allo} of three solvers as shown in Table 4. The ground truth is calculated by the exact LP solver mindopt[18] using the simplex method. **The Lagrangian solver has nearly zero error**, while QPTL/IntOpt have relatively large errors. This is likely due to the fact that the Lagrangian solver was specifically designed and optimized for inventory allocation problems. Additionally, QPTL is unable to solve real-world inventory allocation problems due to memory exceeding issues as the number of constraints is multiplied compared to offline datasets. Next, we compare the performance of the three solvers in end-to-end selling problems as shown in Table 4. **NLS outperforms both baseline solvers.** But surprisingly, on the offline data, although the allocation error of QPTL/IntOpt is obviously significant, their end-to-end performance are relatively comparable. We believe the reason is that, although the forward pass allocation result of QPTL/IntOpt is not accurate enough, its optimization trajectory is relatively correct with respect to the optimal solution. On the two online data sets, the superiority of our end-to-end solver becomes more distinct to solve the GD selling problems. At last, the batch time consumption of each solver was compared, and the results are displayed in Figure 8. The performance comparison was conducted using a single Tesla p100 GPU and **it is evident that our Lagrangian solver has a clear advantage over QPTL/IntOpt**.

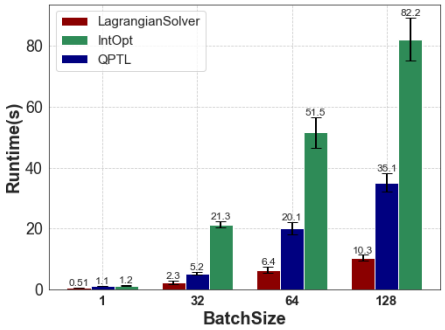

Figure 8: The runtime for different solvers.

6 CONCLUSION

This paper has studied the predictive selling problem for guaranteed delivery advertising. We developed and demonstrated the superiority of the end-to-end approach over traditional two-stage methods. Specifically, the Neural Lagrangian Selling (NLS) model can learn complex and dynamic GD selling cases and solve the contract allocation problem with a Lagrangian dual layer and and a GCN-based bi-graph embedding module. Experiments on both offline and online production data sets demonstrated that NLS improves selling performances and significantly increases publisher revenue compared to two-stage methods. Moreover, our approach has better computational efficiency and lower regret compared to baseline solvers. Our study provides a comprehensive investigation of end-to-end predictive optimization task for GD advertising sales management. Our results have important implications for future research on predictive bi-graph allocation problems.

REFERENCES

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 265–283. https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

[2] Brandon Amos and J Zico Kolter. 2017. Optnet: Differentiable optimization as a layer in neural networks. In *International Conference on Machine Learning*. PMLR, 136–145.

[3] Vijay Bharadwaj, Peiji Chen, Wenjing Ma, Chandrashekhar Nagarajan, John Tomlin, Sergei Vassilvitskii, Erik Vee, and Jian Yang. 2012. Shale: an efficient algorithm for allocation of guaranteed display advertising. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 1195–1203.

[4] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021. Structured graph convolutional networks with stochastic masks for recommender systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 614–623.

[5] Peiji Chen, Wenjing Ma, Srinath Mandalapu, Chandrashekhar Nagarjan, Jayavel Shanmugasundaram, Sergei Vassilvitskii, Erik Vee, Manfai Yu, and Jason Zien. 2012. Ad serving using a compact allocation plan. In *Proceedings of the 13th ACM Conference on Electronic Commerce*. 319–336.

[6] Xiao Cheng, Chuanren Liu, Liang Dai, Peng Zhang, Zhen Fang, and Zhonglin Zu. 2022. An Adaptive Unified Allocation Framework for Guaranteed Display Advertising. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 132–140.

[7] Priya Donti, Brandon Amos, and J Zico Kolter. 2017. Task-based end-to-end model learning in stochastic optimization. *Advances in neural information processing systems* 30 (2017).

[8] Adam N Elmachtoub and Paul Grigas. 2022. Smart "predict, then optimize". *Management Science* 68, 1 (2022), 9–26.

[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[10] Zhen Fang, Yang Li, Chuanren Liu, Wenxiang Zhu, Yu Zheng, and Wenjun Zhou. 2019. Large-scale personalized delivery for guaranteed display advertising with real-time pacing. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 190–199.

[11] Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE international joint conference on neural networks*, Vol. 2. 729–734.

[12] Ali Ugur Guler, Emir Demirović, Jeffrey Chan, James Bailey, Christopher Leckie, and Peter J Stuckey. 2022. A divide and conquer algorithm for predict+ optimize with non-convex problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 3749–3757.

[13] Ali Hojjat, John Turner, Suleyman Cetintas, and Jian Yang. 2014. Delivering guaranteed display ads under reach and frequency requirements. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.

[14] Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR '15)*.

[15] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR '17)*.

[16] Xiaoyang Ma, Lan Zhang, Lan Xu, Zhicheng Liu, Ge Chen, Zhili Xiao, Yang Wang, and Zhengtao Wu. 2019. Large-scale user visits understanding and forecasting with deep spatial-temporal tensor factorization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2403–2411.

[17] Jayanta Mandi and Tias Guns. 2020. Interior point solving for lp-based prediction+ optimisation. *Advances in Neural Information Processing Systems* 33 (2020), 7272–7282.

[18] MindOpt. 2022. MindOpt Studio. https://opt.aliyun.com

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. (Dec 2019). https://doi.org/10.48550/arXiv.1912.01703 arXiv:1912.01703 [cs, stat].

[20] Tara N Sainath, Oriol Vinyals, Andrew Senior, and Haşim Sak. 2015. Convolutional, long short-term memory, fully connected deep neural networks. In *2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. Ieee, 4580–4584.

[21] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE transactions on neural networks* 20, 1 (2008), 61–80.

[22] Bryan Wilder, Bistra Dilkina, and Milind Tambe. 2019. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1658–1665.

[23] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[24] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. 2019. A Comprehensive Survey on Graph Neural Networks. *CoRR* abs/1901.00596 (2019). arXiv:1901.00596 http://arxiv.org/abs/1901.00596

[25] Hong Zhang, Lan Zhang, Lan Xu, Xiaoyang Ma, Zhengtao Wu, Cong Tang, Wei Xu, and Yiguo Yang. 2020. A request-level guaranteed delivery advertising planning: Forecasting and allocation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2980–2988.