Machine Learning (2024) 113:2559–2573
https://doi.org/10.1007/s10994-023-06444-4

Learning to bid and rank together in recommendation systems

Geng Ji[1] · Wentao Jiang[1] · Jiang Li[1] · Fahmid Morshed Fahid[1,2] · Zhengxing Chen[1] · Yinghua Li[1] · Jun Xiao[1] · Chongxi Bao[1] · Zheqing Zhu[1]

Received: 20 March 2023 / Revised: 23 August 2023 / Accepted: 7 October 2023 /
Published online: 29 November 2023

Abstract
Many Internet applications adopt real-time bidding mechanisms to ensure different services (types of content) are shown to the users through fair competitions. The service offering the highest bid price gets the content slot to present a list of items in its candidate pool. Through user interactions with the recommended items, the service obtains the desired engagement activities. We propose a contextual-bandit framework to jointly optimize the price to bid for the slot and the order to rank its candidates for a given service in this type of recommendation systems. Our method can take as input any feature that describes the user and the candidates, including the outputs of other machine learning models. We train reinforcement learning policies using deep neural networks, and compute top-K Gaussian propensity scores to exclude the variance in the gradients caused by randomness unrelated to the reward. This setup further facilitates us to automatically find accurate reward functions that trade off between budget spending and user engagements. In online A/B experiments on two major services of Facebook Home Feed, Groups You Should Join and Friend Requests, our method statistically significantly boosted the number of groups joined by 14.7%, the number of friend requests accepted by 7.0%, and the number of daily active Facebook users by about 1 million, against strong hand-tuned baselines that have been iterated in production over years.

Keywords Recommendation systems · Contextual bandits · Policy gradients · Learning to bid · Learning to rank · Deep reinforcement learning

Editors: Emma Brunskill, Minmin Chen, Omer Gottesman, Lihong Li, Yuxi Li, Yao Liu, Zonging Lu, Niranjani Prasad, Zhiwei Qin, Csaba Szepesvari, Matthew Taylor

✉ Geng Ji
gji@meta.com

[1] Meta Platforms, 1 Hacker Way, Menlo Park, CA 94025, USA

[2] Department of Computer Science, North Carolina State University, Raleigh, NC 27695, USA

1 Introduction

Modern recommendation systems provide users with a personalized selection of diverse content (Pinterest, 2019; Facebook, 2022), such as followers' activities, suggested friends, and nearby businesses. Studies have shown that maintaining content diversity is able to enrich users' experience and avoid their overall fatigue (Clarke et al., 2008; Wilhelm et al., 2018). One important optimization problem in such recommendation systems is to identify the best service to present in each auction session, where we define each service as a type of content, such as the suggestions of friends. A common solution is called Real-Time Bidding (RTB), in which all services each makes its own bid for the opportunity, using a pre-allocated dollar-valued budget (Jansen & Mullen, 2008; Yuan et al., 2013). The free-market auction mechanism of RTB ensures a fair competition for different content providers.

Extensive studies on RTB have been done in the advertising and sponsored search domains (Jansen & Mullen, 2008; Yuan et al., 2013; Zhang et al., 2014; Cai et al., 2017; Ren et al., 2017), which mainly focus on the scenario where each service bids to present the single best candidate of the service in a RTB session. In this paper, we aim for a more general setup, where the service bids for presenting a list of multiple pieces of content in a slot, as illustrated in Fig 1. This requires a policy that simultaneously determines the bid price for the whole slot, and the rank order of the available content items.

A common strategy widely adopted in production environments is through the human-designed value formula with a set of adjustable coefficients. For example, a value formula could perform a linear combination of the user's predicted engagement signals for each piece of content. The output of the value formula is used to rank the candidate items and to decide the bid price. Tuning the parameters in the value formula back and forth in the online settings can take a large amount of time and human efforts. Additionally, the complexity of value formulas are heavily limited by the number of parameters that hand tuning

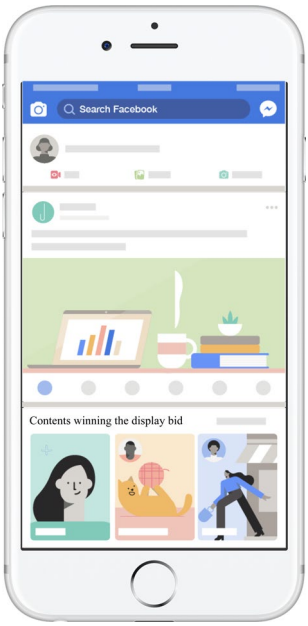

Fig. 1 The service that wins the content slot gets the chance to display multiple items to the user to interact with, like the three items shown at the bottom of the phone screen. These slots locate between regular feed contents, and recommend items such as groups or friends

can afford, and the simple weighting rules that human can think of, which draws questions to its capability of representing complex environments.

This paper proposes a novel framework to learn the optimal policy for solving Bidding And Ranking Together (BART), from the perspective of an individual service participating the auction. We resort to contextual bandit algorithms (Li et al., 2010; Dudik et al., 2011; Agarwal et al., 2012; Bottou et al., 2013; Agarwal et al., 2014; Swaminathan & Joachims 2015, 2015), which are known for its capability to derive an optimal policy from imperfect and sub-optimal past experience. Leveraging data collected through a random logging policy, we design a training algorithm that is able to learn more complicated policies than the rule-based model in production. The novel learning framework also greatly reduces the human efforts required to maintain the policy. Our contributions can be summarized in three folds:

- We formulate the problem of jointly deciding the bid price and rank order of candidates for a given service as a contextual bandit, which has not been explored in previous work to the authors' best knowledge.
- We propose an effective batch learning algorithm to train sophisticated policies from sub-optimal demonstrations. Our innovative design of top-K Gaussian propensity scores stabilizes training by removing unwanted variance in the policy gradients. We also propose a reward-shaping algorithm to automatically determine the best form of the reward function.
- We tested our algorithm on two major services in the Home Feed of Facebook. In online A/B experiments, our approach outperforms strong hand-tuned baselines used in production that have been iterated over years. Our work has been launched into the production of these services, and is helping billions of users to find communities and make friends online in a better way.

2 Related work

Real-Time Bidding (RTB) has been a crucial mechanism in many domains of web applications, such as advertisements (Yuan et al., 2013) and sponsored search (Jansen & Mullen, 2008). Most existing research in RTB revolves around the scenario where each bidder can win a display slot for at most one item, so there is no ranking of multiple candidates involved in the bidding process. RTB can typically be broken down into three stages: (1) utility prediction, such as click-through rate prediction based on advanced supervised learning models (McMahan et al., 2013; Cheng et al., 2016; Guo et al., 2017; Naumov et al., 2019); (2) bid price optimization, which utilizes information from the user, candidates, utility predictions, and other competitors (Li & Guan, 2014; (3) budget allocation (Lee et al., 2013; Xu et al., 2015), which ensures bidding costs satisfy desired constraints and smooths budget spending across time. People have proposed algorithms to tackle each stage separately or optimize multiple stages jointly (Zhang et al., 2014; Ren et al., 2017; Cai et al., 2017). In this paper, we focus on the second stage (i.e. bid price optimization) to determine the best bid price over multiple candidates of a given service, with the additional requirement to rank them, which is little researched to the best of our knowledge.

Our goal of learning the optimal personalized policy for bidding and ranking together can be abstracted into a process of learning from imperfect demonstration. In our case, the imperfect demonstration comes from adding noise to the current hand-tuned policy. This is

a suitable use case for reinforcement learning (RL) (Sutton & Barto, 2018). RL has shown its strong capability to improve the policy intelligently not only in simulation environments like board games (Silver et al., 2017) and video games (Mnih et al., 2013), but also in real-world applications (Gauci et al., 2018), such as device placement (Mirhoseini et al., 2017), ride sharing (Xu et al., 2018), notifications (Liu et al., 2021), and video recommendation (Chen et al., 2019).

In this paper, we assume that the outcome for one user session has no consequence in the future. Therefore, we focus on contextual bandit algorithms, a special domain of RL which only focuses on optimizing decisions for one time step. Contextual bandit applications can be categorized into two paradigms: online learning, where the policy can constantly interact with the environment and make responsive update to real-time feedback (Li et al., 2010; Dudik et al., 2011; Agarwal et al., 2012, 2014); and offline learning, where the policy is learned from the data collected in one-time fashion (Strehl et al., 2010; Bottou et al., 2013; Swaminathan & Joachims, 2015, 2015). The focal point of the former is usually in how to conduct efficient exploration about the environment, while the latter is mainly in counterfactual estimation of a policy's performance. Our work belongs to the offline learning paradigm as it is more amenable to existing offline training infrastructure already adopted by many companies in the industry. The core algorithm we used in our work is offline policy optimization (Strehl et al., 2010; Bottou et al., 2013), also known as Counterfactual Risk Minimization (Swaminathan & Joachims, 2015), which has been successfully applied to offline learning a video retrieval policy in a large-scale video recommendation system (Chen et al., 2019), a spelling correction algorithm in a search engine (Li et al., 2015), etc.

Our work learns a scoring function that outputs a scalar value for each candidate, which decides the order of the items when displayed in the content slot. This idea has also been widely used in learning to rank (Taylor et al., 2008; Liu, 2009; Agarwal et al., 2019) in the information retrieval literature. But one key difference is that instead of optimizing the ranking metrics which only depend on the relative magnitudes of the scores, our approach also uses the scoring function to control the bid price, so the absolute magnitudes of the scores matter as well.

Another topic relevant to our paper is called whole-page presentation optimization (Wang et al., 2016; Hill et al., 2017; Jiang et al., 2018; Bello et al., 2018; Ding et al., 2019). The task is to optimize the arrangement of different services in a personalized page as a whole. It is different from our settings because in these scenarios, the different sources of items all get presented together, instead of relying on the auction mechanism and only displaying candidates from the winning service.

3 Model formulation

3.1 Background and notations

To better explain our model, we first summarize how we assume the recommendation system works and introduce the related notations:

- Every time when a user opens the recommendation system app, an auction session is initiated to display a content slot to the user at a given position, which allows different services to bid against each other to compete for the opportunity to present.
- The service we work on retrieves I items from its pool of candidates. Features of each candidate are collected, including utility predictions from trained supervised learning models, such as click-through rates. The feature for each candidate $i \in \{1, \cdots, I\}$ is denoted as a vector $\boldsymbol{c_i}$. We use $\boldsymbol{c} \triangleq [\boldsymbol{c_1}, \cdots, \boldsymbol{c_I}]$ as the feature representation for all the candidates retrieved by the service in this auction session.
- We want to learn a policy to decide the score value x_i for each candidate i, a scalar that indicates how much the user is interested in this item. Let $\boldsymbol{x} \triangleq [x_1, \cdots, x_I]$, and $\tilde{\boldsymbol{x}} \triangleq [\tilde{x}_1, \cdots, \tilde{x}_I]$ sorts $\boldsymbol{x}$ in descending order.
- Our service's bid price $b(\boldsymbol{x})$ is a function of the scores. In the rest of the paper, we assume $b(\boldsymbol{x}) \triangleq \sum_{j=1}^{J} w_j \tilde{x}_j$, where $J \leq I$ is the number of items that can be displayed in the content slot. The weights $\boldsymbol{w}$ are fixed to be proportional to the empirical conversion rate of each position, which usually decreases rapidly because users do not tend to slide the horizontal scroll till the end when browsing the app. Note that the method we are going to introduce does not rely on the linear format of $b(\boldsymbol{x})$ described here.
- If our bid price $b(\boldsymbol{x})$ is higher than all other competing services, the top J items of our service, whose scores are $\tilde{\boldsymbol{x}}_{1:J}$, will be shown sequentially to the user in the content slot (see Fig 1 as an example where $J \geq 3$). Meanwhile, $b(\boldsymbol{x})$ will be deducted from the budget of our service.[1]

3.2 Contextual bandit setup

We formulate the problem of optimizing Bidding And Ranking Together (BART) as a contextual bandit (CB). CB takes actions based on the state input, and gets different reward values as the outcome. Below we define the state, action and reward of BART in details.

3.2.1 State: information about the user and candidates

A state $\boldsymbol{s} \triangleq \{\boldsymbol{u}, \boldsymbol{c}\}$ represents the context of our service in an auction session. It includes the features for each candidate $\boldsymbol{c}$ described in Sect. 3.1, and a user-only feature vector $\boldsymbol{u}$ containing information shared by all candidates, such as user age, time of the day, and the probability of being an active user.

3.2.2 Action: candidate scores

Because both the bid price and the rank order are determined by the candidate scores, the action of our problem reduces to $\boldsymbol{a} \triangleq \boldsymbol{x} = \{x_1, \cdots, x_I\}$. We treat the score of each candidate i given the state as a Gaussian random variable:

$$x_i \mid \boldsymbol{s} \sim Gaussian\big(\mu_{\boldsymbol{\theta}}(\boldsymbol{u}, \boldsymbol{c_i}), \sigma^2\big), \tag{1}$$

[1] This is known as the first-price auction. Our proposed method also works well in the second-price auction scenarios, as explained in Sect. 3.2

in which the mean value $\mu_{\theta}(\boldsymbol{u}, \boldsymbol{c}_i)$ is a function parameterized by $\boldsymbol{\theta}$, and the standard deviation σ is a hyper-parameter. The algorithm to learn the parameter $\boldsymbol{\theta}$ will be introduced in Sect. 4.

3.2.3 Reward: a mix of costs and benefits

Equation (2) summarizes our reward function using the indicator function $\mathbb{1}\{\cdot\}$, which takes value 1 when the condition is met and 0 otherwise:

$$r_{lose\_bid} \cdot \mathbb{1}\{lose\_bid\} + \left(r_{engage} \cdot \Delta_{metric} - b(\boldsymbol{x})\right) \cdot \left(1 - \mathbb{1}\{lose\_bid\}\right). \tag{2}$$

If our service loses the bid, we set the reward to r_{lose_bid}. Otherwise, we first give it a negative reward $-b(\boldsymbol{x})$, which is our bid price, to indicate the cost we need to pay. This setup is straightforward to understand for first-price auction, in which the winner pays the price they bid. But note that it also works for second-price auction where the winner pays the second-highest bid price, because if we still subtract $-b(\boldsymbol{x})$ when we win, Equation (2) becomes a lower bound of the (negative) cost to pay. Maximizing this lower bound leads to reasonable bidding and ranking strategies, as we will show in the experiments in Sect. 5.

Finally, we add $r_{engage} \cdot \Delta_{metric}$ to the reward, to include the metric gain resulted from the user's interactions with the displayed items. For example, if the metric goal of our service is to maximize the total number of user activities in this service, then Δ_{metric} could be the number of activities the user takes after clicking the suggested items in this content slot.

4 Policy optimization

In this section, we introduce our policy (Sect. 4.1), describe how to optimize it via offline training (Sect. 4.2), and build a reward shaping algorithm (Sect. 4.3) to automatically choose the hyper-parameters r_{lose_bid} and r_{engage} in the reward function defined in Eq. (2).

4.1 Top-*K* Gaussian policy

A policy $\pi(\boldsymbol{a} \mid \boldsymbol{s})$ defines the conditional probability distribution of taking an action given the state. According to the definition in Sect. 3.2, our action $\boldsymbol{a}$ is the scores of all the I candidates, each of which is a scalar Gaussian variable (Eq. (1)). Therefore, a natural choice for π would be a multivariate Gaussian. However, this is not ideal as it will introduce randomness irrelevant to the reward. This is because that only the top J candidates actually contribute to the bid price $b(\boldsymbol{x})$, and that only they get the chance to be shown to the users and to affect the Δ_{metric} term in the reward. To exclude the extra randomness caused by the values of other scores $\tilde{\boldsymbol{x}}_{K+1:I}$, we propose a top-$K$ Gaussian policy, with probability density function:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}) = \frac{1}{Z} \cdot \prod_{k=1}^{K} f_k\left(\tilde{x}_k; \boldsymbol{s}\right) \cdot \prod_{i=K+1}^{I} F_i\left(\tilde{x}_K; \boldsymbol{s}\right). \tag{3}$$

In Eq. (3), $f_i(\cdot; \boldsymbol{s})$ and $F_i(\cdot; \boldsymbol{s})$ denote the Gaussian probability density function and cumulative distribution function of the i-th largest score $\tilde{x}_i$. The last factor of Eq. (3) ensures that scores $\tilde{\boldsymbol{x}}_{K+1:I}$ are all smaller than $\tilde{x}_K$. The normalization constant

$$Z = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \prod_{k=1}^{K} f_k\Big(\tilde{x}_k; \boldsymbol{s}\Big) \cdot \prod_{i=K+1}^{I} F_i\Big(\min\{\tilde{x}_1, \cdots, \tilde{x}_K\}; \boldsymbol{s}\Big) \mathrm{d}\tilde{x}_1 \cdots \mathrm{d}\tilde{x}_K$$

can be computed using Monte Carlo approximations via the reparameterization trick (Kingma & Welling, 2014; Rezende et al., 2014).

Note that instead of forcing it to be equal to the number of display positions J, we make K a hyper-parameter and only require $K \leq J$. This treatment is to take into account the fact that users tend to only view the first few candidates placed at the beginning of the content slot, instead of taking the time to browse through all the J suggested items.

Also note that our top-K Gaussian policy is fundamentally different from the top-K off-policy correction in Chen et al. (2019), which independently computes the probability of choosing each item via a softmax policy, and uses sampling with replacement plus a de-duplicate process to construct a set of (at most) K items. Their computations are based on the assumptions that a user will interact with at most one item from the returned set, and that the expected return of an item is independent of other items in the set. In contrast, our top-K Gaussian policy is subject to neither of these constraints.

It is also worth mentioning that our top-K Gaussian policy setup is better than just adding a Gaussian noise to the final bid price $b(\boldsymbol{x})$, as the later formulation does not randomize the rank order of the candidate items, and thus provides less explorations to cover the action space of BART during data collection and policy training.

4.2 Batch learning objective

The trainable parameters in our policy $\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s})$ are $\boldsymbol{\theta}$, which decides the mean of the scores $\boldsymbol{x}$ defined in Eq. (1). To get the best value of $\boldsymbol{\theta}$, we optimize the policy by maximizing the expected reward

$$R(\boldsymbol{\theta}) \triangleq \mathbb{E}_{p(\boldsymbol{s})\pi_\theta(\boldsymbol{s}|\boldsymbol{a})}[r(\boldsymbol{s}, \boldsymbol{a})] = \mathbb{E}_{p(\boldsymbol{s})\pi_b(\boldsymbol{s}|\boldsymbol{a})}\left[\frac{\pi_\theta(\boldsymbol{a} \mid \boldsymbol{s})}{\pi_b(\boldsymbol{a} \mid \boldsymbol{s})} \cdot r(\boldsymbol{s}, \boldsymbol{a})\right], \tag{4}$$

in which $p(\boldsymbol{s})$ denotes the state distribution. Following the setup of offline policy optimization (Strehl et al., 2010; Bottou et al., 2013) and counterfactual risk minimization (Swaminathan & Joachims, 2015), $R(\boldsymbol{\theta})$ in Eq. (4) is rewritten into an expectation with respect to the behavior/logging policy π_b via importance sampling. By deploying π_b to the environment and collecting the feedback rewards, we build the empirical training objective $\hat{R}(\boldsymbol{\theta})$ based on the logged data triples $\{\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}, r^{(n)}\big(\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}\big)\}$:

$$\hat{R}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \frac{\pi_\theta\big(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\big)}{\pi_b\big(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\big)} \cdot r^{(n)}\big(\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}\big), \tag{5}$$

in which $\boldsymbol{s}^{(n)} \sim p(\boldsymbol{s}), \boldsymbol{a}^{(n)} \sim \pi_b(\boldsymbol{a} \mid \boldsymbol{s}^{(n)})$. However, one big issue of the estimator in Eq. (5) is that it has unbounded variance when $\pi_b\big(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\big)$ gets close to zero, making $\hat{R}(\boldsymbol{\theta})$ arbitrarily far away from $R(\boldsymbol{\theta})$. This problem becomes especially severe when π_b depends on too many random variables, making the density function of their joint distribution very small on some data points. Our top-K Gaussian policy introduced in Sect. 4.1 partially solves this problem by numerically integrating out the unwanted random noise (Casella & Robert, 1996; Ji et al., 2021) caused by the non-top-K candidate scores. To further stabilize the

training, we clip the importance sampling weight, as done in many previous work (Strehl et al., 2010; Bottou et al., 2013; Swaminathan & Joachims, 2015; Chen et al., 2019):

$$\hat{R}^M(\boldsymbol{\theta}) \triangleq \frac{1}{N}\sum_{n=1}^{N} \min\left\{\frac{\pi_{\boldsymbol{\theta}}\left(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\right)}{\pi_b\left(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\right)}, M\right\} \cdot r^{(n)}\left(\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}\right). \tag{6}$$

Equation (6) is the final objective we use for training. We optimize it via gradient-based algorithms such as Adam (Kingma & Ba, 2015) available in the PyTorch deep learning framework (Paszke et al., 2019). In this batch learning setup (Swaminathan & Joachims, 2015), the offline training data only needs to be collected once, which saves us the heavy infra burden of repeatedly publishing $\pi_{\boldsymbol{\theta}}$ online to collect new batches of data for re-estimating the gradients, and thus accelerates the overall training process.

4.3 Reward shaping

To ensure the trained policies can really improve the product metrics, the reward values assigned to the training data should accurately measure the gains and losses of each possible outcome. This is known as reward shaping in reinforcement learning (Ng et al., 1999). In our problem, it means how to best weigh r_{lose_bid} and r_{engage} against the dollar-valued bid price $b(\boldsymbol{x})$ in Eq. (2). We need a proper way to measure how good or bad it is to lose a bid and to get a user engagement that affects the metrics.

We propose a novel approach by reversely inferring r_{lose_bid} and r_{engage} from a simpler policy $\pi_{\boldsymbol{\phi}}$ defined in the same action space as $\pi_{\boldsymbol{\theta}}$ but with less parameters. In particular, we require the dimension of $\boldsymbol{\phi}$ to be equal to the number of hyper-parameters in the reward function, which is two in our case. Then we tune $\boldsymbol{\phi}$ in online experiments so that the user engagement metric is optimized.[2] At the best value $\boldsymbol{\phi}^*$, we compute the gradient of the expected reward $R(\boldsymbol{\phi})$ with respect to $\boldsymbol{\phi}$:

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}} R(\boldsymbol{\phi})\big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(s)\pi_{\boldsymbol{\phi}}(a|s)}\left[r(\boldsymbol{s},\boldsymbol{a})\right]\big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} \\
&= \mathbb{E}_{p(s)}\left[\int r(\boldsymbol{s},\boldsymbol{a})\nabla_{\boldsymbol{\phi}}\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})\mathrm{d}\boldsymbol{a}\right]\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} \\
&= \mathbb{E}_{p(s)}\left[\int r(\boldsymbol{s},\boldsymbol{a})\nabla_{\boldsymbol{\phi}}\log\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})\mathrm{d}\boldsymbol{a}\right]\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} \\
&= \mathbb{E}_{p(s)\pi_{\boldsymbol{\phi}}(a|s)}\left[r(\boldsymbol{s},\boldsymbol{a})\nabla_{\boldsymbol{\phi}}\log\pi_{\boldsymbol{\phi}}(\boldsymbol{a} \mid \boldsymbol{s})\right]\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*}.
\end{aligned} \tag{7}$$

Because $\pi_{\boldsymbol{\phi}*}$ optimizes the online product metric within this constrained family of policies, $\boldsymbol{\phi}^*$ must be an (at least local) optimum of $R(\boldsymbol{\phi})$, meaning that Eq. (7) should be equal to a vector of zeros, $\nabla_{\boldsymbol{\phi}} R(\boldsymbol{\phi})\big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} = \mathbf{0}$. We want to make use of these two equations to infer the values of r_{lose_bid} and r_{engage}. Because numerically computing the expectations is intractable, we collect data samples and build a Monte Carlo estimate of $\nabla_{\boldsymbol{\phi}} R(\boldsymbol{\phi})\big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*}$ to get

[2] We assume that manually tuning such a simple policy is easy to do, or one may even already have such a hand-tuned policy to start with, as we will see in Sect. 5.1.

$$\frac{1}{N}\sum_{n=1}^{N} r^{(n)}\left(\boldsymbol{s}^{(n)}, \boldsymbol{a}^{(n)}\right) \cdot \nabla_{\boldsymbol{\phi}} \log \pi_{\boldsymbol{\phi}}\left(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\right)\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*} \triangleq \mathbf{0} \tag{8}$$

using data samples $\boldsymbol{s}^{(n)} \sim p(\boldsymbol{s}), \boldsymbol{a}^{(n)} \sim \pi_{\boldsymbol{\phi}^*}(\boldsymbol{a} \mid \boldsymbol{s}^{(n)})$. Given the reward definition in Eq. (2), Equation (8) is actually a system of linear equations with respect to r_{lose_bid} and r_{engage}, which is easy to solve with the help of automatic differentiation tools (Paszke et al., 2017; Baydin et al., 2018) to compute the gradient terms $\nabla_{\boldsymbol{\phi}} \log \pi_{\boldsymbol{\phi}}\left(\boldsymbol{a}^{(n)} \mid \boldsymbol{s}^{(n)}\right)\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}^*}$. Compared to traditional inverse RL algorithms (Abbeel & Ng, 2004; Ziebart et al., 2008; Ho & Ermon, 2016), the reward shaping approach proposed here is much simpler to compute, without the slow back-and-forth iterations between policy optimization and reward function parameter update. As we will see in Sect. 5, this approach provides accurate reward settings to reflect the product environment, and helps to decide the strength of explorations in the logged training data.

5 Experiments

We test the performance of our method on two major services in the Home Feed of Facebook, Groups You Should Join (GYSJ) and Friend Requests (FR). Both services, among many others, bid for the same content slot, which shows the top $J = 20$ items of the winning service to the user in a horizontal scroll. To avoid affecting each other, we select different time periods to run the A/B experiment for each service. For both GYSJ and FR, we use $K = 3$ in the top-K Gaussian policies and $M = 100$ for the clipping threshold.

5.1 Service 1: groups you should join

The main goal of the GYSJ product is to encourage users to join more groups and interact with the contents in them, such as clicking the like button, commenting and sharing. The current value formula in GYSJ production is a linear function that combines the user's expected Click-Through Rate (eCTR, probability of the user joining a group after seeing the recommendation) and expected Post-Click Conversion Rate (eCVR, probability of the user interacting with the contents in the group after joining it) for each candidate, both of which are predicted by pre-trained deep supervised learning models. The linear combination is then multiplied by a predetermined function of p(active user), the probability of the user being an active group user in the current week. This probability comes from the prediction of another supervised learning model, and can be viewed as a user-only feature because it does not depend on any candidate information. The linear combination weights for eCTR and eCVR have been extensively tuned by hand to reach the highest user engagement metric in online A/B experiments. The dashed curves in Fig 2 visualize the product rule.

5.1.1 Behavior policy and reward shaping

We leverage the current value formula to construct a top-K Gaussian policy, and use it for both the behavior policy π_b in Sect. 4.2 and the reward-shaping policy $\pi_{\boldsymbol{\phi}}$ in Sect. 4.3. Because both policies need to be stochastic, we use the value formula as the mean of each

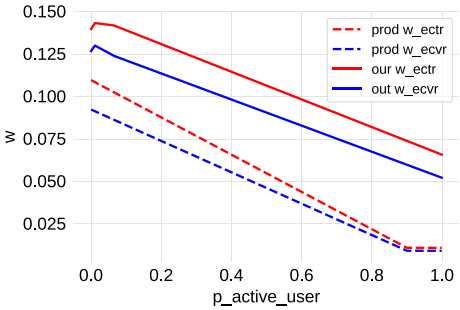

Fig. 2 The policies for the GYSJ service can be visualized by showing the linear combination weights for eCTR and eCVR as a function of the user-only feature, which is the probability prediction of the user being active in the current week

candidate score, and select the best standard deviation σ' through the reward shaping algorithm.

Concretely, we try several values of σ' to construct $\pi_{\boldsymbol{\phi}}$ of Sect. 4.3, where $\boldsymbol{\phi}$ corresponds to the two linear combination weights for eCTR and eCVR. We collect some data for each value of σ', and find that the results of the reward-shaping algorithm are very similar to each other when the values of σ' are relatively small. Concretely, r_{engage} tends to be a large positive value, and r_{lose_bid} a small value around zero. This makes intuitive sense because user engagement should definitely be encouraged. Losing bids does not boost user engagement, but is not too destructive either, because it also saves the budget, especially when the candidates available are not attracting enough to the users.

We also find that when σ' passes a threshold, the scores becomes too noisy for the reward shaping algorithm to return a positive value for r_{engage}. Therefore, our final choice for σ' is the biggest value before we first see a negative r_{engage} in the reward-shaping algorithm. We further use this policy as the behavior policy π_b in Sect. 4.2, as it strikes a good balance between exploration and exploitation. We set the standard deviation in Eq. (1) σ to $\frac{1}{2}\sigma'$, so that less uncertainty is included in the policy to learn than the behavior policy.

5.1.2 Model performance

To justify our method is able to learn a better policy than the value formula under the same family of linear scoring functions, we force the model architecture for $\mu_{\boldsymbol{\theta}}(\boldsymbol{u}, \boldsymbol{c}_i)$ in Eq. (1) to be similar to the hand-designed production rule. Concretely, we make it a fully connected neural network with two hidden layers and ReLU activation functions. It takes in the user-only feature $\boldsymbol{u}$, which is p(active user), as input and outputs a 2D vector as the linear combination weights for $\boldsymbol{c}_i$, which is [eCTR, eCVR]. For each data point, there are $I = 75$ candidates, provided by the upstream ranking and retrieval stages of the GYSJ product.

The daily results of a 22-day-long online A/B experiment are summarized in Fig 3. On top of the current hand-tuned value formula, our trained policy increased the top-line engagement metric by 0.44%, based on the average of the last seven days. More specifically, the number of impressions (groups shown to the users) increased by 9.8%, indicating that our bidding strategy has been improved. The number of group joins increased by 14.7%, even larger than the impression gains, meaning that our ranking strategy has also been improved. We confirmed that the same budgets were used for both the test and control versions on each day to ensure fair comparisons.

The solid lines in Fig 2 visualize the policy trained using our BART algorithm. It resembles the value formula, in the sense that they both put more weights to eCTR (red)

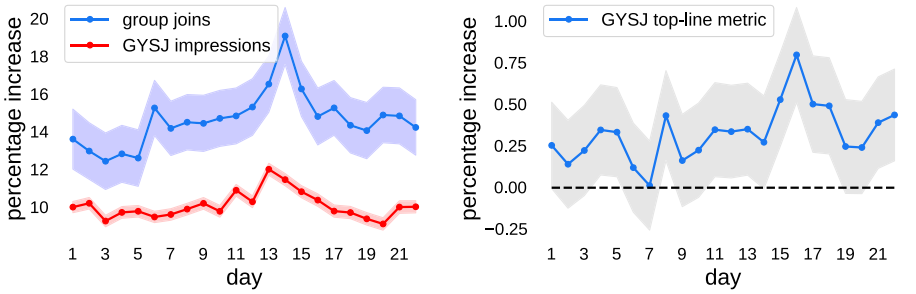

Fig. 3 Online performance of our method against the hand-tuned value formula on the GYSJ service. Both the test version and the baseline version contain 22.8 million users. Shaded areas indicate the 95% confidence intervals

than eCVR (blue), and that the more active the users, the lower their weights. However, our BART policy outputs higher weights, leading to more aggressive bid prices. Moreover, most of the model capacity of BART is used to learn the weights for the inactive users, instead of mimicking the value formula's hard clips of the weights for the most active users.

5.2 Service 2: friend requests

5.2.1 Training details

The main goal of the FR product is to encourage users to accept pending friend requests. The current value formula in FR is the probability of accepting the request, p(accept), which is obtained by training a binary classification model. Because there is no existing hand-tuned policy to construct π_{ϕ}, we directly set $r_{engage} = 100$, $r_{lose\ bid} = 0.01$, and the standard deviations of the candidate scores in π_{θ} and π_b to 0.1 and 0.2, respectively.

To make the problem more interesting, in addition to p(accept), we also select nine features used when training the supervised baseline model into our state representations. These features include user-only ones, such as the number of friends the user has and the time of the day the user opens Facebook. They also include user-candidate features, such as how long the request has been pending, and predictions from other machine learning models, like the cosine distance between the embedding vectors of the user and the candidate. Similar to the settings in GYSJ, we train a neural network with two hidden layers for μ_{θ}. But instead of outputting the linear combination weights, the FR BART model takes all the ten features as the input, and directly outputs the score for each candidate.

Another difference is that unlike GYSJ where we can always retrieve a fixed number of groups, the number of available candidates (pending friend requests) varies a lot across users. A nice property of the top-K Gaussian propensity scores in Eq. (3) is that it can handle different values of I for different data points, even when $I \leq K$. But to avoid cases where I is extremely large, we truncate its maximum value to $I_{max} = 30$.

5.2.2 Model performance

Daily results of a 30-day-long A/B experiment of our trained policy against the production baseline is shown in Fig 4. The learned policy increases the number of accepted friend requests by 7.0% in the Home Feed traffic of Facebook (blue curve), while the logging policy

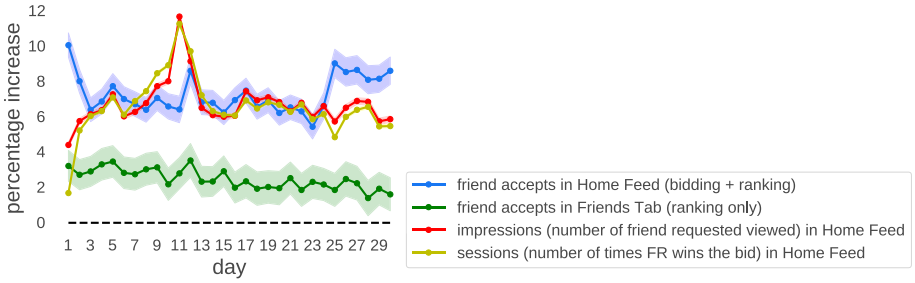

Fig. 4 Online performance of our method against the baseline production model for Friend Requests. Both the test version and the baseline version contain 126.3 million users. Shaded areas indicate the 95% confidence intervals. The BART model is trained using Home Feed data, but to get a better understanding of the improvement made for bidding and ranking separately, we also deploy the model to Friends Tab (green) to just control the ranking

that adds noise to the production baseline gets a −11.3% drop. Both the number of sessions (yellow curve) and the number of friend requests viewed (red curve) are increased by 6.6%, meaning that the increase in the impressions mostly comes from the improved bidding strategy. We confirmed the same budgets were used for both the test and control versions on each day to ensure fair comparisons. Compared to GYSJ, we observe less improvement of clicks (blue curves) relative to the increase in the impressions (red curves). We think this is because the limited supply of the total number of pending friend requests for each person, compared to the many good candidates that can be shown to the users in group recommendations.

To better understand the improvement contributed by bidding and ranking separately, we also deploy our trained model to the Friends Tab of Facebook, which always shows the pending friend requests at the top of the page, based on the order of the scores given by our BART model versus the baseline p(accept) model. In this pure ranking problem, our BART model reached an improvement of 2.4% in terms of the number of accepted friend requests (green curve in Fig 4), less than the improvement in Home Feed (blue curve), which involves both bidding and ranking. We think the performance gain in terms of ranking comes from three aspects. First, the stochastic behavior policy π_b enables explorations of different rank orders. Second, BART can properly use data points that lost the bid, while the supervised baseline can only train on data with accept or not accept labels. Finally, our method naturally handles the position of each candidate based on the relative magnitude of the scores, while the baseline p(accept) model does not consider the position information at all because how to fix the position bias remains an open problem (Craswell et al., 2008; Wang et al., 2018).

For both GYSJ and FR, we have launched the BART models we trained into the Facebook production. The backtest results matched what we observed in pretests, and showed that BART statistically significantly improved the number of daily active users by 928 K, as well as monthly active users by 1.64 M.

6 Conclusion

We have formulated the bidding and ranking problem of a single service with multiple candidates to display in a free-market recommendation system into a contextual bandit framework. Our top-K Gaussian policies are able to remove the noise independent of the reward in offline stochastic gradients. By leveraging existing policies tuned by hand, our lightweight reward-shaping algorithm accurately captures the reward and cost for each

possible outcome. Online A/B experiments on two major services in Facebook have shown the advantage of our work in increasing the top-line user engagement metrics. In the future, we plan to better learn and leverage the uncertainty of the policies, and work on solutions that can optimize the performance of multiple services jointly.

Author contributions GJ conceived of the presented idea and designed the algorithms, after discussing the problems to solve with WJ and JL. The first implementation of this method was done by GJ and ZC. Later when it got generalized for broader usage, YL, JL and FMF also contributed to the code refactoring. GJ, WJ, JL and FMF collected the data, conducted the experiments, and analyzed the results. JX, CB and ZZ supervised the project. GJ, ZC and ZZ wrote the manuscript, and all authors contributed to the reviewing of it.

Funding No funding was received to assist with the preparation of this manuscript.

Availability of data and materials Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors have no relevant financial or non-financial interests to disclose.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. In *International conference on machine learning*

Agarwal, A., Dudík, M., Kale, S., Langford, J., & Schapire, R. (2012). Contextual bandit learning with predictable rewards. In *Artificial intelligence and statistics*

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., & Schapire, R. (2014). Taming the monster: A fast and simple algorithm for contextual bandits. In *International conference on machine learning*

Agarwal, A., Takatsu, K., Zaitsev, I., & Joachims, T. (2019). A general framework for counterfactual learning-to-rank. In *Conference on research and development in information retrieval*

Baydin, A. G., Pearlmutter, B. A., Radul, A. A., & Siskind, J. M. (2018). Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research, 18*, 1–43.

Bello, I., Kulkarni, S., Jain, S., Boutilier, C., Chi, E., Eban, E., Luo, X., Mackey, A., & Meshi, O. (2018). Seq2Slate: Re-ranking and slate optimization with RNNs. arXiv preprint arXiv:1810.02019

Bottou, L., Peters, J., Quiñonero-Candela, J., Charles, D. X., Chickering, D. M., Portugaly, E., Ray, D., Simard, P., & Snelson, E. (2013). Counterfactual reasoning and learning systems: The example of computational advertising. *Journal of Machine Learning Research*, 14(101), 3207−3260.

Cai, H., Ren, K., Zhang, W., Malialis, K., Wang, J., Yu, Y., & Guo, D. (2017). Real-time bidding by reinforcement learning in display advertising. In *International conference on web search and data mining*

Casella, G., & Robert, C. P. (1996). Rao-blackwellisation of sampling schemes. *Biometrika, 83*(1), 81–94.

Chen, M., Beutel, A., Covington, P., Jain, S., Belletti, F., & Chi, E. H. (2019). Top-*K* off-policy correction for a REINFORCE recommender system. In *International conference on web search and data mining*

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., & Ispir, M. (2016). Wide & deep learning for recommender systems. In *Workshop on deep learning for recommender systems*

Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S., & MacKinnon, I. (2008). Novelty and diversity in information retrieval evaluation. In *Conference on research and development in information retrieval*

Craswell, N., Zoeter, O., Taylor, M., & Ramsey, B. (2008). An experimental comparison of click position-bias models. In *International conference on web search and data mining*

Ding, W., Govindaraj, D., & Vishwanathan, S. (2019). Whole page optimization with global constraints. In *International conference on knowledge discovery & data mining*

Dudik, M., Hsu, D., Kale, S., Karampatziakis, N., Langford, J., Reyzin, L., & Zhang, T. (2011). Efficient optimal learning for contextual bandits. arXiv preprint arXiv:1106.2369

Facebook: Facebook News Feed: An Introduction for Content Creators. https://www.facebook.com/business/learn/lessons/facebook-news-feed-creators. Online; accessed January, 2023 (2022)

Gauci, J., Conti, E., Liang, Y., Virochsiri, K., He, Y., Kaden, Z., Narayanan, V., Ye, X., Chen, Z., & Fujimoto, S. (2018). Horizon: Facebook's open source applied reinforcement learning platform. arXiv preprint arXiv:1811.00260

Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). Deepfm: A factorization-machine based neural network for ctr prediction. arXiv preprint arXiv:1703.04247

Hill, D. N., Nassif, H., Liu, Y., Iyer, A., & Vishwanathan, S. (2017). An efficient bandit algorithm for real-time multivariate optimization. In *International conference on knowledge discovery and data mining*

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. *Advances in neural information processing systems*

Jansen, B. J., & Mullen, T. (2008). Sponsored search: An overview of the concept, history, and technology. *International Journal of Electronic Business, 6*(2), 114–131.

Ji, G., Sujono, D., & Sudderth, E. B. (2021). Marginalized stochastic natural gradients for black-box variational inference. In *International conference on machine learning*

Jiang, R., Gowal, S., Mann, T. A., & Rezende, D. J. (2018). Beyond greedy ranking: Slate optimization via list-CVAE. arXiv preprint arXiv:1803.01682

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational Bayes. In *International conference on learning representations*

Kingma, D. P., Ba, J. (2015). Adam: A method for stochastic optimization. In *International conference on learning representations*

Lee, K.-C., Jalali, A., & Dasdan, A. (2013). Real time bid optimization with smooth budget delivery in online advertising. In *International workshop on data mining for online advertising*

Li, X., & Guan, D. (2014). Programmatic buying bidding strategies with win rate and winning price estimation in real time mobile advertising. In *Pacific-Asia conference on knowledge discovery and data mining*

Li, L., Chen, S., Kleban, J., & Gupta, A. (2015). Counterfactual estimation and optimization of click metrics in search engines: A case study. In *Proceedings of the 24th international conference on world wide web*

Li, L., Chu, W., Langford, J., & Schapire, R. E. (2010). A contextual-bandit approach to personalized news article recommendation. In *International conference on world wide web*

Liu, Y., Chen, Z., Virochsiri, K., Wang, J., Wu, J., & Liang, F. (2021). Reinforcement learning-based product delivery frequency control. In *AAAI conference on artificial intelligence*

Liu, T.-Y. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval, 3*(3), 225–331.

McMahan, H. B., Holt, G., Sculley, D., Young, M., Ebner, D., Grady, J., Nie, L., Phillips, T., Davydov, E., & Golovin, D. (2013). Ad click prediction: A view from the trenches. In *International conference on knowledge discovery and data mining*

Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S., & Dean, J. (2017). Device placement optimization with reinforcement learning. In *International conference on machine learning*

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602

Naumov, M., Mudigere, D., Shi, H.-J. M., Huang, J., Sundaraman, N., Park, J., Wang, X., Gupta, U., Wu, C.-J., Azzolini, A. G., et al. (2019). Deep learning recommendation model for personalization and recommendation systems. arXiv preprint arXiv:1906.00091

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *International conference on machine learning*

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch. In *NIPS autodiff workshop*

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., & Antiga, L. (2019). PyTorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*

Pinterest: New ways to control the ideas you see in your home feed. https://newsroom.pinterest.com/en/post/new-ways-to-control-the-ideas-you-see-in-your-home-feed. Online; accessed January, 2023 (2019)

Ren, K., Zhang, W., Chang, K., Rong, Y., Yu, Y., & Wang, J. (2017). Bidding machine: Learning to bid for directly optimizing profits in display advertising. *Knowledge and Data Engineering, 30*(4), 645–659.

Rezende, D. J., Mohamed, S., & Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., & Bolton, A. (2017). Mastering the game of Go without human knowledge. *Nature, 550*(7676), 354–359.

Strehl, A., Langford, J., Li, L., & Kakade, S. M. (2010). Learning from logged implicit exploration data. *Advances in neural information processing systems*

Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT Press

Swaminathan, A., & Joachims, T. (2015). The self-normalized estimator for counterfactual learning. *Advances in neural information processing systems*

Swaminathan, A., & Joachims, T. (2015). Batch learning from logged bandit feedback through counterfactual risk minimization. *Journal of Machine Learning Research, 16*(1), 1731–1755.

Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008). Softrank: Optimizing non-smooth rank metrics. In *International conference on web search and data mining*

Wang, X., Golbandi, N., Bendersky, M., Metzler, D., & Najork, M. (2018). Position bias estimation for unbiased learning to rank in personal search. In *International conference on web search and data mining*

Wang, Y., Yin, D., Jie, L., Wang, P., Yamada, M., Chang, Y., & Mei, Q. (2016). Beyond ranking: Optimizing whole-page presentation. In *International conference on web search and data mining*

Wilhelm, M., Ramanathan, A., Bonomo, A., Jain, S., Chi, E. H., & Gillenwater, J. (2018). Practical diversified recommendations on youtube with determinantal point processes. In *International conference on information and knowledge management*

Xu, J., Lee, K.-c., Li, W., Qi, H., & Lu, Q. (2015). Smart pacing for effective online ad campaign optimization. In *International conference on knowledge discovery and data mining*

Xu, Z., Li, Z., Guan, Q., Zhang, D., Li, Q., Nan, J., Liu, C., Bian, W., & Ye, J. (2018). Large-scale order dispatch in on-demand ride-hailing platforms: A learning and planning approach. In *International conference on knowledge discovery and data mining*

Yuan, S., Wang, J., & Zhao, X. (2013). Real-time bidding for online advertising: Measurement and analysis. In *International workshop on data mining for online advertising*

Zhang, W., Yuan, S., & Wang, J. (2014). Optimal real-time bidding for display advertising. In *International conference on knowledge discovery and data mining*

Ziebart, B. D., Maas, A. L., Bagnell, J. A., & Dey, A. K. (2008). Maximum entropy inverse reinforcement learning. In *AAAI conference on artificial intelligence*