Maximizing User Click - Optimizing Bid Advertising

Youssof Bendary, Keegan Veazey, Rajat Pratap Singh Bisht, Hamza Akmal Chaudary, and Sarang Ajay Patel

1 Project Overview

The landscape of optimizing online advertising predominantly relies on real-time bidding (RTB) [1]. To optimize advertisement placements, the companies have been consistently challenged with developing dynamic learning algorithms that enable this strategic bidding. Optimizing RTB to maximize key performance indicators such as user impressions within real-world constraints enables dynamic budget allocation based on both immediate and anticipated rewards. Traditional approaches have often treated bid decisions as static optimization problems where the value of each impression, such as a click, is independently set and assigned to some segment of ad volume [1]. Additionally, many existing bidding strategies assume a truthfulness to bid price: they assume each ad impression should be equal to the action value (e.g., click amount) multiplied by the action rate (such as clickthrough rate) [2]. Of course, the optimal bidding strategy based on this assumption concludes with largely untruthful results since RTB and bid amount also depend on dynamic changes in market competition and auction volume.

For this project, we draw inspiration from research papers exploring the constrained bidding optimization research space including Di Wu, et al. work "Budget Constrained Bidding by Model-free Reinforcement Learning in Display Advertising" published in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. We also draw inspiration from Han, et al. "Real-time bidding by reinforcement learning in display advertising" published in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. Our project goal is to build a simplified RTB environment simulator on which we train a single-agent Q-Learning approach to learn a bidding strategy that optimizes the number of impressions, i.e. number of ad clicks within a budget. While both papers will aid in guiding our work to construct our problem statement as a Markov Decision Process (MDP), we are not strictly following either. For example, the "Real-time bidding by reinforcement learning in display advertising" paper ultimately approaches the RTB problem space with dynamic programming and trains using two expansive datasets. The paper "Budget Constrained Bidding by Model-free Reinforcement Learning in Display Advertising" is more aligned with our Q-Learning approach, though they experiment with DQN and maximizing click-through rate rather than the number of impressions. See the related work section of this proposal for further information regarding these papers. Note if time permits, we plan on building a multi-agent Q-Learning system where we have agents act as competitor companies.

2 Related Work

In model-free ECPC [7], the authors propose a bidding strategy that leverages a combination of estimated conversion rate and batch reinforcement learning to optimize the ad bids dynamically on an hourly basis. The ECPC bidding is formulated as a Markov decision process. The method doesn't rely on the model exploration but rather focuses on a model-free RL where it tries to minimize the extrapolation error by aligning policy actions with historical data. The MDP is characterized by states representing the auction environment, including bid prices, impressions, and budgets. The agent's actions correspond to bid adjustments, while rewards are determined by the number of clicks or impressions. The problem is tackled using Q-learning, where the Q-function Q(s,a)Q(s,a)

estimates the expected future rewards for each state-action pair, updated iteratively via the Bellman equation:

Further work by Di Wu et al. [2] explores a model-free RL approach specifically aimed at optimizing budget-constrained bidding. They use a variant of Deep Q-Network (DQN), where the Q-function is approximated using deep neural networks rather than a tabular method. This allows the algorithm to handle the high-dimensional state space typical in online auctions. The authors employ experience replay and target network stabilization techniques to mitigate instability during training, and their reward function is designed to maximize the number of impressions while adhering to budget constraints. Formally, the reward function r_t in their setting is given as: $r_t = I_t - \lambda *(C_t/B_t)$ where I_t represents the impressions obtained at time step t, C_t is the cumulative cost up to t, B_t is the total budget, and λ is a regularization parameter that penalizes excessive budget consumption.

A notable challenge in RTB systems is the stochastic nature of the environment, particularly the large state and action spaces resulting from the combinatorial auction dynamics. Cai et al. [9] address this by leveraging dynamic programming for value function approximation, along with neural networks for policy learning. They use the policy iteration method, where the agent alternates between policy evaluation, updating the value function V(s) using the Bellman equation, and policy improvement by adjusting the bidding strategy to maximize expected future rewards. Their implementation benefits from techniques such as ϵ-greedy exploration to balance exploration and exploitation in learning optimal bidding strategies.

Since an ad impression has 2 states, the authors [8] consider clicks as binary variables and model clicks as Bernoulli trials, applying Bayesian inference with the beta distribution. This helps to update click probability with real-time performance. This method uses both network and campaign-level data to provide more accurate bid recommendations.

Limitations of model-free RL were shown by the authors [9] who showed that they can suffer from transition dynamics in large state spaces and its highly stochastic nature. Rather, the Markov decision process was used to learn the optimal bidding policy, combined with dynamic programming for value function approximation. They also incorporated neural networks to handle large state spaces.

3 Proposed Methods

Pay-Per-Click (PPC) is a revenue generation model that runs in parallel with Search-Engine-Optimization (SEO) algorithms, ever since the advent of search engines [7]. Even today a wide variety of online services including social media platforms and digital marketplaces employ the PPC model for earning profit. In this model, each keyword is associated with a cost based on the number of times that particular keyword was searched as well as the number of clicks on top results for that keyword. Small businesses make use of this PPC model to promote their products to new customers and increase the company's outreach, by bidding for these keywords. Biddings imply the cost per click (CPC) that business has to pay to these search and social media platforms for showcasing their ads as well only when the advertisements receive clicks.

An ongoing bottleneck with these PPC optimizations is to decide how much of the allocated budget should be invested in bidding for certain keywords and to decide which keywords are high priority and which ones are low priority for the business's welfare. Thus, this bottleneck simply

boils down to an Optimization problem where we are interested in minimizing the cost incurred by the business while maximizing the number of clicks that land customers to our advertised products. Therefore, we propose the idea of implementing an Agent trained using Reinforcement Learning (RL) due to the success of RL agents with combinatorial optimization problems. [8]

To implement our RL model, we will start by reshaping the optimization problem as a Markov decision Process (MDP). MDP 'states' for this Agent will include current CPC, keyword performance, performance with respect to the given budget, and competitor's bidding activity (more states can be employed as work progresses). MDP 'actions' associated with the Agent will be the decision to either make a bid or to hold off, as well as to decide what the bidding amount should be. The agent will receive feedback on how under or over the placed bid is as compared to the optima based on which policy will be adjusted. 'Policies' for the agent will include adjusting bidding amounts, choosing profitable keywords with higher priorities, and reallocating budgets. As of now, due to our limited knowledge in this field, we have decided to use Q-learning for implementing RL Agent, but this decision is susceptible to change as the coursework progresses.

Due to the real-world expense associated with such agents, we will develop a virtual environment for training and early deployment of this RL Agent. For an apt simulation of the real world, we will use the bidding data available for various keywords, which due to a large number of bids turns out to be a normal distribution histogram. We propose defining an optimal bid by the environment by removing the tail (outlier data) from the Normal distribution, then sampling the remaining values into a discrete value set and using a randomized probabilistic selection for choosing a certain bid for a particular keyword. Considering a shifted or skewed normal distribution of bidding data, we will use Z-scores (by removing data beyond 2 or 3 standard deviations) and percentile thresholding (for skewed normal distribution) to smooth the dataset before sampling.

Future work on this project will entail deploying two or more variations of the trained RL agent in the virtual environment and having them compete against each other. We can also take statistical data for the bidding values for a keyword from more than one advertisement platform, and check them to select only those that are empirically significantly similar (using t-tests for means [9][4], ANOVA[5] and Levene's tests for variances [6]; where $\alpha = 0.05$ or 5%) and generating a normalized normal distribution to feed into our Virtual environment to test for a more relatable real-world scenario.

4 Timeline

We expect the initial setup of our RTB environment simulator, including data processing and defining the MDP structure, to be completed within the first two weeks. This includes simulating bidding data and implementing keyword-performance tracking. In the following two weeks, we will begin training our single-agent Q-Learning model to optimize the bidding strategy. By the end of this phase, we aim to have preliminary results evaluating the agent's ability to maximize impressions while adhering to budget constraints.

After the project check-in, we will evaluate our Q-Learning agent's performance and, if time permits, explore implementing a multi-agent system where competing agents act as rival companies. We allocate two additional weeks to refining the agent's policy based on test outcomes and adjusting the bidding strategies. Final testing and the development of a comparison with

baseline approaches will take place in the last two weeks, with final adjustments to ensure the agent operates effectively within the simulated environment.

5 References

[1] Cai, Han, et al. “Real-time bidding by reinforcement learning in display advertising.” *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining,* 2 Feb. 2017, https://doi.org/10.1145/3018661.3018702.

[2] Wu, Di, et al. “Budget constrained bidding by model-free reinforcement learning in display advertising.” *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 17 Oct. 2018, https://doi.org/10.1145/3269206.3271748.

[3] S. Li, C. Yuan and X. Zhu, “Optimizing Enhanced Cost Per Click via Reinforcement Learning Without Exploration,” 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 2021, pp. 1-8, doi: 10.1109/IJCNN52387.2021.9534126.

[4] Skaik, Y. (2015). The bread and butter of statistical analysis “t-test”: Uses and misuses. *Pak J Med Sci, 31*(6), 1558-1559.

[5] Fisher, R. A. (1992). The arrangement of field experiments. In *Breakthroughs in statistics: Methodology and distribution* (pp. 82-91). New York, NY: Springer New York.

[6] Carroll, R. J., & Schneider, H. (1985). A note on Levene's tests for equality of variances. *Statistics & probability letters*, *3*(4), 191-194.

[7] Drolias, B. (2007). *Pay-Per-Click: The Complete Guide*. Lulu. com.

[8] Mazyavkina, N., Sviridov, S., Ivanov, S., & Burnaev, E. (2021). Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, *134*, 105400.

[9] Student. (1908). The probable error of a mean. *Biometrika*, 1-25.