LBM: Hierarchical Large Auto-Bidding Model via Reasoning and Acting

Yewen Li*
Kuaishou Technology
Beijing, China
liyewen@kuaishou.com

Zhiyi Lyu*
Nanyang Technological University
Singapore, Singapore
zhiyi.lyu@ntu.edu.sg

Peng Jiang
Kuaishou Technology
Beijing, China
jiangpeng07@kuaishou.com

Qingpeng Cai†
Kuaishou Technology
Beijing, China
caiqingpeng@kuaishou.com

Fei Pan
Kuaishou Technology
Beijing, China
panfei05@kuaishou.com

Bo An
Nanyang Technological University
Singapore, Singapore
boan@ntu.edu.sg

Peng Jiang
Kuaishou Technology
Beijing, China
jiangpeng@kuaishou.com

Abstract

The growing scale of ad auctions on online advertising platforms has intensified competition, making manual bidding impractical and necessitating auto-bidding to help advertisers achieve their economic goals. Current auto-bidding methods have evolved to use offline reinforcement learning or generative methods to optimize bidding strategies, but they can sometimes behave counterintuitively due to the black-box training manner and limited mode coverage of datasets, leading to challenges in understanding task status and generalization in dynamic ad environments. Large language models (LLMs) offer a promising solution by leveraging prior human knowledge and reasoning abilities to improve auto-bidding performance. However, directly applying LLMs to auto-bidding faces difficulties due to the need for precise actions in competitive auctions and the lack of specialized auto-bidding knowledge, which can lead to hallucinations and suboptimal decisions. To address these challenges, we propose a hierarchical **L**arge auto-**B**idding **M**odel (**LBM**) to leverage the reasoning capabilities of LLMs for developing a superior auto-bidding strategy. This includes a high-level LBM-Think model for reasoning and a low-level LBM-Act model for action generation. Specifically, we propose a dual embedding mechanism to efficiently fuse two modalities, including language and numerical inputs, for language-guided training of the LBM-Act; then, we propose an offline reinforcement fine-tuning technique termed GQPO for mitigating the LLM-Think's hallucinations and enhancing decision-making performance without simulation or real-world rollout like previous multi-turn LLM-based methods. Experiments demonstrate the superiority of a generative backbone based on our LBM, especially in an efficient training manner and generalization ability.[1]

CCS Concepts

• **Information systems** → **Computational advertising**.

Keywords

Auto-bidding, Large Language Model, Generative Model, Offline Reinforcement Learning

ACM Reference Format:
Yewen Li, Zhiyi Lyu, Peng Jiang, Qingpeng Cai, Fei Pan, Bo An, and Peng Jiang. 2026. LBM: Hierarchical Large Auto-Bidding Model via Reasoning and Acting. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3774904.3792202

1 Introduction

The rapid digitization of commerce has significantly expanded the reach of online advertising platforms on the web, leading to a surge in advertisers competing for millions or even billions of impression opportunities through ad auctions [10, 44]. Previously, advertisers relied on experienced human experts to manually adjust bids, but it is becoming impractical due to the increasing competitiveness of auctions [30]. Therefore, auto-bidding has become essential for helping advertisers achieve their commercial goals in the face of competition [38]. Auto-bidding can be modeled as a sequential decision-making task, where the objective is to maximize conversions while adhering to specific economic constraints, such as cost-per-action (CPA), by adjusting bidding parameters over a bidding period [18, 29]. Current auto-bidding methods are mainly built on offline reinforcement learning (RL) or generative methods, leveraging bidding logs collected from online advertising platforms [13, 25]. These methods aim to learn optimal policies by stitching sub-optimal trajectories from datasets to achieve reward-maximizing trajectories [1]. Offline RL typically performs trajectory

*Both authors contributed equally to this research.
†Corresponding author.

WWW '26, Dubai, United Arab Emirates

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792202

[1]Code is available at https://github.com/yewen99/LBM-WWW26.

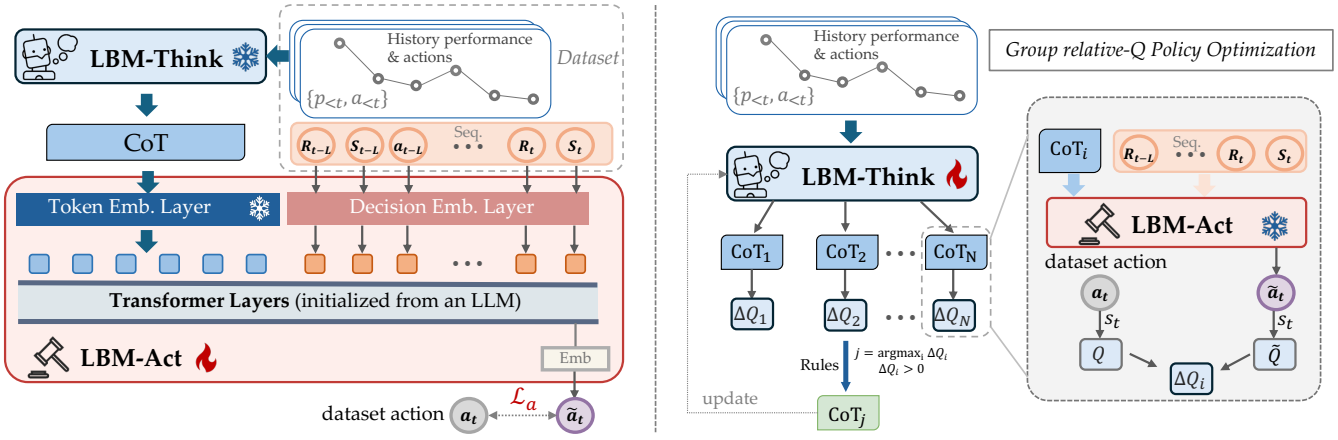


Figure 1: Overview of the training scheme for our hierarchical large auto-bidding model. Training Stage I (left): train the LBM-Act with language-guided decision training via a dual embedding mechanism to fuse two modalities; Training Stage II (right): reinforcement fine-tuning the LBM-Think via group relative-Q policy optimization.

stitching using dynamic programming with a value function [20]. Generative auto-bidding methods, which do not rely on a value function, perform stitching through a conditional generation approach utilizing the powerful generative models [1, 7, 19]. For instance, the Decision Transformer (DT) employs an auto-regressive transformer to generate bid actions based on historical sequences that encompass return-to-go, states, and actions [7, 26].

However, these RL and generative methods can occasionally act counterintuitively, such as increasing bidding parameters even when the CPA significantly exceeds the constraint, which is unlikely to occur in manual bidding. This issue arises because these methods primarily rely on reward design to acquire desired properties [1, 19, 24]. Consequently, it becomes challenging to ensure that the model fully understands the task status, resulting in a fully black-box approach. Additionally, since it's impossible to capture every corner case into datasets, these methods are inherently restricted by the mode coverage of the offline dataset and lack guarantees for generalization performance in dynamic advertising environments, especially in unforeseen situations [25]. These limitations present significant challenges and could be major obstacles to the progress and adoption of advanced auto-bidding services, as frequent abnormal model behavior can undermine the trust of advertisers [45]. Fortunately, large language models (LLMs) are revolutionizing decision-making by harnessing prior human knowledge to boost performance [3, 36]. These LLMs effectively follow human instructions and utilize their reasoning ability with pre-trained knowledge to comprehend the task status and reasoning for optimal actions, showcasing superior performance and generalization ability in tasks like math and code [16]. Thus, incorporating LLMs as a new generative backbone for auto-bidding presents promising opportunities for improvement.

While LLMs can be directly applied to auto-bidding by converting sequential numerical information into language token inputs and generating bidding parameters as a response, their performance is constrained via prompt engineering [9, 47] in large-scale auctions due to the generated suboptimal actions. This limitation arises from the intense competition among advertisers, where their bids can be very close, and thus, the need for precise actions is critical for optimality [17, 42]. A suboptimal bidding parameter could lead to budget waste or limited gains in impressions. For instance, if the bidding parameter is set too high, the budget may be exhausted fast, missing later opportunities; conversely, if set too low, it may result in no acquired impressions. To our knowledge, current public LLMs have not been specially pretrained or fine-tuned on auto-bidding data, as there is no public auto-bidding data in text format; thus, they could still suffer from the notorious hallucinations issue and lack optimal sequential decision-making ability [11, 54].

There have been many attempts to enhance LLMs for decision-making tasks, though they are not well-suited for the auto-bidding task. To leverage the reasoning capabilities of LLMs, some approaches, such as reinforcement learning with verifiable rewards (RLVR) [16], aim to enhance performance in tasks like mathematical reasoning, code generation, and computer control [11]. These methods primarily focus on reasoning abilities rather than continuous control, where the action space is typically discrete, such as in tool-use agents [2], whereas auto-bidding operates in a continuous numerical action space. For applications of LLMs in tasks with continuous state and action spaces, there is one challenge that representing numerical inputs as language tokens can be costly [40, 53]; for instance, a numerical state like "12.34" requires five tokens, potentially resulting in thousands of tokens when using extensive historical data in complex tasks. Even if it is done in this way, the attention mechanism of LLMs could be limited, leading to hallucinations and suboptimal performance [23, 49]. For addressing this, LLM-DT [40] fine-tunes the LLM with additional embedding layers that process long numerical sequences into more informative representations. However, this approach actually treats the LLM as a Decision Transformer (DT) initialized with transformer layers from the LLM, which limits the full exploitation of the LLM's knowledge and reasoning capabilities.

To leverage the reasoning capabilities of LLMs for developing an optimal auto-bidding strategy, we introduce a hierarchical **L**arge auto-**B**idding **M**odel (**LBM**), which comprises two modules: i) the LBM-Think module, which handles high-level reasoning in the

language space, and ii) the LBM-Act module, which focuses on low-level decision-making in the continuous action space. Specifically, the LBM-Think module is tasked with generating a chain-of-thought (CoT) [47] in language format that summarizes and reflects historical bidding status, while reasoning about high-level future actions like adjustment directions. Note that CoT can be pre-generated asynchronously ahead of the next decision timestep as shown in Fig. 4, making it suitable for industry applications. For the LBM-Act module, we introduce a **dual embedding** mechanism to encode both the numerical sequence of observations and language CoT. This hierarchical structure enables the LBM to comprehend task status while executing precise control efficiently. Additionally, it separates CoT generation and action execution into distinct LLM backbones, with LBM-Think utilizing a larger LLM and LBM-Act employing a smaller LLM. For training such a hierarchical LBM, we propose a two-stage training scheme, as illustrated in Fig. 1. In the first stage, we introduce a language-guided decision training approach with additional embedding layers to train the LBM-Act model, allowing it to effectively fuse two modalities of inputs and generate precise actions. In the second stage, we propose a Group relative-Q Policy Optimization (**GQPO**) method to fine-tune the LBM-Think model, reducing hallucinations and achieving better reasoning performance in a stable offline manner. This relative-Q serves as an assessment of the effectiveness of a CoT to fine-tune the LBM-Think entirely offline, unlike previous multi-turn LLM-based agents that require rollout in simulators or the real world, which is not feasible and can be risky for auto-bidding [2, 11]. To summarize, the contributions of this work are:

- To leverage the reasoning capabilities of LLMs for developing an advanced auto-bidding strategy, we propose a novel hierarchical **L**arge auto-**B**idding **M**odel (**LBM**) based on LLMs, including a high-level LBM-Think for reasoning and a low-level LBM-Act for precise action generation;
- We propose a **dual embedding** mechanism to efficiently fuse two modalities, including language and numerical inputs, for language-guided training of the LBM-Act;
- We propose a stable fine-tuning technique termed **GQPO** for mitigating the LLM-Think's hallucinations and enhancing decision-making performance in an offline manner;
- Experiments demonstrate the superiority of a generative backbone based on LLMs, especially regarding the efficient training manner and generalization ability.

2 Preliminary

2.1 Problem Statement

Consider a scenario where impression opportunities arrive sequentially, each identified by an index i. An advertiser secures an impression if its bid b_i exceeds those of other advertisers, resulting in a cost c_i. The objective is to maximize the cumulative value of the impressions won, expressed as $\sum_i o_i v_i$, where v_i represents the value of the impression and o_i is a binary variable indicating whether the advertiser wins impression i. Furthermore, it is crucial to account for budgetary and various economic constraints, such as limiting the unit cost for specific advertising events like CPC and CPA [18]. For simplicity, we focus on auto-bidding with cost-related constraints, which can be uniformly formulated as: $\frac{\sum_i c_{ij} o_i}{\sum_i p_{ij} o_i} \le C_j$, where C_j is the upper bound of j-th constraint provided by the advertiser. p_{ij} can be any performance indicator, *e.g.*, return or constant. c_{ij} is the cost of constraint j. Therefore, the objective of auto-bidding is:

$$\begin{aligned} \text{maximize} \quad & \sum_i o_i v_i \\ \text{s.t.} \quad & \sum_i o_i c_i \le B \\ & \frac{\sum_i c_{ij} o_i}{\sum_i p_{ij} o_i} \le C_j, \quad \forall j \\ & o_i \in \{0, 1\}, \quad \forall i \end{aligned} \tag{1}$$

A previous study [18] has already shown the optimal solution:

$$b_i^* = \lambda_0 v_i + \sum_{j=1}^{J} \lambda_j p_{ij} C_j, \tag{2}$$

where b_i^* is the optimal bid for impression i. The parameters λ_j represent the optimal bidding parameters. Please note that in industrial ad platforms, **current auto-bidding methods mainly focus on adjusting bidding parameters at a low frequency, such as every 30 minutes, rather than for every single impression that arises in millisecond** [42]. Therefore, there is ample time for an advanced auto-bidding method to respond to bidding parameter adjustment requirements.

2.2 Decision-making Process for Auto-bidding

Given the dynamic nature of the advertising environment, optimal bidding parameters must be regularly adjusted to maximize total value, framing the auto-bidding task as a sequential decision-making process. At each timestep t of a bidding period, the agent receives a state $s_t \in \mathcal{S}$ describing the real-time advertising status and then outputs an action $a_t \in \mathcal{A}$ for the final bid. The advertising environment has an underlying unknown state transition dynamic $\mathcal{T}$. The next state could be determined by both the historical information and current observations. After transitioning to the next state, the advertising environment would emit a reward r_t representing the value contributed to the objective obtained within the time period t. This process would repeat until the end of a bidding period, such as one day. A detailed description of our modeling is listed below:

- s_t: the state is a collection of information that describes the advertising status from the perspective of a campaign. The information should in principle reflect time, budget consumption and KPI constraints satisfying status, such as left time, left budget, budget consumption speed, current KPI ratio for constraint j (i.e. $(\Sigma_i c_{ij} o_i / \Sigma_i p_{ij} o_i)/C_j$), etc.
- a_t: adjustment to the bidding parameters λ_j, $j = 0, \ldots, J$, at the time period t, and modeled as $(a_t^{\lambda_0}, \ldots, a_t^{\lambda_J})$.
- r_t : a typical setting of the reward could be the conversion value contributed to the objective obtained within the time period from t to $t+1$.

3 LBM: Hierarchical Large Auto-Bidding Model

In this section, we first explain how we construct the hierarchical Large auto-Bidding Model (LBM) with two modules: high-level LBM-Think and low-level LBM-Act. Next, we propose a two-stage

training method for the model. Initially, we introduce a language-guided decision training method for LBM-Act. To further enhance the LBM's performance, particularly the reasoning ability of the LBM-Think, we then present the GQPO method, which utilizes an offline Q-value for fine-tuning it.

3.1 Model Structure and Inference

To improve reasoning capabilities, previous methods expand models to incorporate a large number of parameters, often surpassing 1 billion. This approach can be costly and result in slow inference for industrial applications, especially if each decision timestep requires immediate reasoning and action using LLMs. Even if we aim to create a model capable of both reasoning and generating optimal actions in a continuous action space, training such a model is challenging and may degrade the original model's general reasoning ability. This is illustrated in Fig. 6, where we use the GRPO to fine-tune an LLM for action generation with CoT. Therefore, a practical and flexible approach is to propose a hierarchical structure that decouples reasoning and acting into two separate modules. Reasoning capabilities can leverage existing LLMs to process historical information asynchronously, without needing a response at every decision timestep. The acting capability can be built on a smaller LLM that focuses on immediate decision-making using the pre-generated reasoning output and current state information.

For sequential inference procedure of auto-bidding, we can observe a sequence of historical information with horizon length H, *i.e.*, $\{p_i, a_i\}_{i=t-H:t-1}$, where p_i is the key performance indicators reflecting the bidding status like the achieved conversions, budget utilization ratio, and CPA. For every decision-making step, we have a state s_t representing the current detailed observation, like the impression value distribution and auction status. For the high-level LBM-Think model, denoted as h_θ, it summarizes and reflects the historical sequential information only and then reasons for a future high-level decision like the adjustment direction of the bidding parameters, producing a CoT denoted as c_t, *i.e.*,

$$\textbf{LBM-Think}: \quad c_t = h_\theta(\{p_i, a_i\}_{i=t-H:t-1}). \tag{3}$$

Note that this high-level process could be done asynchronously before timestep t, and we could simplify the performance indicator p with only the necessary elements from states, thus leaving sufficient time for the LLM to generate. After receiving CoT c_t and current state s_t, the LBM-Act model f_ϕ generates an action $\tilde{a}_t$, *i.e.*,

$$\textbf{LBM-Act}: \quad \tilde{a}_t = f_\phi(s_t, c_t). \tag{4}$$

To enhance the decision-making performance, we can extend the s_t to include more information in a DT manner, including *return-to-go* $R_{\le t}$, states $s_{\le t}$, and actions $a_{<t}$. Therefore, the LBM-Act needs to fuse two modalities of inputs, including language and number. A detailed illustration of LBM could be seen in Fig. 1 and Fig. 4.

Given that different modules have distinct objectives, where LBM-Think h_θ focuses on understanding the bidding status and LBM-Act f_ϕ concentrates on low-level action generation, we propose a stable two-stage training approach. Specifically, this involves first training the LBM-Act through language-guided decision training using a dual embedding mechanism, followed by reinforcement fine-tuning of the LBM-Think with the trained LBM-Act.

3.2 Language-guided Decision Training for LBM-Act via Dual Embedding

While it's possible to convert numerical states into language format, this approach is inefficient and can consume a significant number of tokens. For example, representing "12.34" in language format can require 5 tokens, and thus handling sequences of high-dimensional states could result in the use of thousands of tokens. Additionally, the performance of attention mechanisms is often constrained when processing long sequences of token inputs, particularly when using smaller language models for LBM-Act, which can lead to suboptimal performance in auto-bidding scenarios [23, 49]. To address these challenges, we propose a **dual embedding** mechanism to fuse these two modalities, rather than unifying them solely in the language modality. Dual embedding involves employing two distinct embedding layers to process data: one for language modality using a pretrained token embedding layer for the CoT generated by the LBM-Think model, and another for numerical data, like observed numerical sequences, using a *decision embedding layer*. Each numerical state is projected into a single embedding using an additional MLP, termed "decision embedding", which matches the size of a token embedding and encapsulates decision-related information. By integrating these two embeddings, LBM-Act then employs transformer layers initialized with the pretrained weight of an LLM to project them to the final layer's output representation, and a final action is determined via an MLP processing this representation. Therefore, with this dual embedding mechanism, the LBM-Act can effectively comprehend language instructions and manage complex decision-making tasks.

To enhance the LBM-Act model's ability to handle two modalities, we propose a language-guided decision training method. Specifically, we first utilize the LBM-Think module to generate a CoT, which involves summarizing historical bidding performance information and reasoning for optimal future high-level decisions, such as determining the direction for bidding parameter adjustments. Additionally, we obtain a corresponding numerical sequence $\{R_{t-L:t}, s_{t-L:t}, a_{t-L:t-1}\}$ of length L, representing more comprehensive and detailed information. In previous DT methods, the model learns to generate a_t based on this sequence, aiming to achieve the current return-to-go R_t during the training stage, rather than learning to generate an optimal action like offline RL methods such as IQL. However, there is uncertainty in achieving the goal represented by R_t, expressed as $p(a_t) = f(R_{\le t}, s_{\le t}, a_{<t}) = a_{t-1} + \epsilon$, where $\epsilon \in \mathbb{R}$ represents the uncertainty. It is not always clear which action aligns with the direction of the optimal action. For example, a favorable R_t could be achieved with a bad a_t but good subsequent actions $a_{>t}$. To address potential conflicts between the two modalities during training, we use the direction of the dataset action a_t compared to a_{t-1} as an anchor direction. If a reasoned high-level direction in the CoT conflicts with this anchor direction, it will be disregarded. The accepted CoT component is concatenated with the numerical sequence $\{R_{t-L}, s_{t-L}, a_{t-L}, \ldots, R_t, s_t\}$ and sent to the dual embedding to obtain their corresponding embeddings $\boldsymbol{z} = \{z_{cot}, \boldsymbol{z}_s\}$. The inner transformer layer learns to fuse these two modalities to generate actions using the attention mechanism, *i.e.*, $\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$, where $\boldsymbol{Q} = \boldsymbol{z}\boldsymbol{W}^Q$,

$\boldsymbol{K} = \boldsymbol{z}\boldsymbol{W}^K$, and $\boldsymbol{V} = \boldsymbol{z}\boldsymbol{W}^V$. The output prediction action $\tilde{a}_t$, based on the fused representation, is trained to match the labeled action a_t in the dataset, *i.e.*,

$$\mathcal{L}_a(\phi) = \|\tilde{a}_t - a_t\|_2 . \tag{5}$$

Upon convergence, the LBM-Act can comprehend language guidance, follow its instructions, and achieve Return-to-go in continuous action space. During the inference stage, the return-to-go is typically initialized to its maximum value at the beginning timestep, aiming to achieve an optimal bidding strategy that aligns with the objective of LBM-Think.

3.3 Offline Reinforcement Fine-Tuning for LBM-Think via GQPO

Since existing LLMs have not been extensively pretrained on auto-bidding datasets in language format, they may exhibit hallucinations and demonstrate a suboptimal bidding strategy. Previous related works on decision-making using LLMs have widely employed reinforcement fine-tuning techniques for enhanced performance [2, 11, 16, 54]. A representative work is the group relative policy optimization (GRPO) [16], expressed as

$$\mathcal{J}_{\mathrm{GRPO}}(\theta) = \mathbb{E}_{q\sim P(Q),\{o_i\}_{i=1}^{G}\sim\pi_{\theta_{\mathrm{old}}}(O|q)} \tag{6}$$

$$\frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\left\{\min\left[\rho_{i,t}\hat{A}_{i,t}, \mathrm{clip}\left(\rho_{i,t}, 1\pm\varepsilon\right)\hat{A}_{i,t}\right] - \beta\mathcal{D}_{\mathrm{KL}}\left[\pi_\theta\|\pi_{\mathrm{ref}}\right]\right\},$$

where $\rho_{i,t} = \frac{\pi_\theta(o_{i,t}|q,o_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(o_{i,t}|q,o_{i,<t})}$ is the importance sampling weight, π_θ and $\pi_{\theta_{\mathrm{old}}}$ represent the current and previous policy models, respectively. The variables q and o denote questions and outputs sampled from the question dataset and the old policy $\pi_{\theta_{\mathrm{old}}}$. The parameter ϵ is a clipping-related hyperparameter to stabilize training. The advantage $\hat{A}_t$ is the advantage calculated based on the relative rewards of the outputs inside each group. It needs a verifiable outcome-based reward function to score the outputs $\{o_i\}_{i=1}^G$, yielding G rewards $r = \{r_1, r_2, ..., r_G\}$ correspondingly. GRPO could be directly extended to multi-turn decision-making tasks, leading to methods like GiGPO [11]. However, these methods rely on rollouts in a simulator or the real world to obtain correct rewards, which is not feasible and could be risky in the context of auto-bidding. Consequently, it is necessary to explore a reinforcement fine-tuning approach that operates entirely offline.

The primary challenge lies in evaluating the impact of CoT reasoning on decision-making performance without real-world rollouts. Recall previous offline RL methods, the training objective is typically based on an advantage-weighted regression (AWR) technique for policy extraction [31, 32, 34, 46], *i.e.*,

$$\mathcal{J}_{\mathrm{AWR}}(\theta) = \mathbb{E}_{s,a\sim\mathcal{D}}[\exp(\beta(Q_\varphi(s,a) - V_\psi(s))\log\pi_\theta(a|s))], \tag{7}$$

where Q represents the state-action pair value, V represents the state value that serves as a baseline for variance reduction, and $\beta \in [0, +\infty)$ is a hyperparameter. In this work, we can train the value functions using a stable offline RL method named Implicit Q-Learning (IQL) [20], formulated as:

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}[L_2^\tau(Q_{\bar{\varphi}}(s,a) - V_\psi(s))], \tag{8}$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u<0)|u^2$ represents an expectile regression loss. The value network $V_\psi(s)$ is utilized in the learning of Q-values:

$$\mathcal{L}_Q(\varphi) = \mathbb{E}_{(s_t,a_t,s_{t+1})\sim\mathcal{D}}[r(s_t,a_t) + \gamma V_\psi(s_{t+1}) - Q_\varphi(s_t,a_t))^2]. \tag{9}$$

In the context of the relationship between these offline RL methods and our LBM, the policy $\pi(a|s)$ can be viewed as an expectation of the CoT c_t, expressed as

$$\pi(a|s) = \mathbb{E}_{c_t\sim\pi(c_t|s)}\pi(a|s,c_t). \tag{10}$$

We denote the value of a state-action pair conditioned on the CoT c_t as $Q(s_t, \tilde{a}_t)$. Our objective is to obtain an improved policy through an enhanced CoT by

$$\pi(a|s, c_t^j), c_t^j = \arg\max_{c_t}[Q(s_t,\tilde{a}_t) - Q(s_t,a_t)]. \tag{11}$$

For a practical implementation, we can assess the impact of CoT on decision-making by examining the difference in predicted Q-values for actions generated with and without CoT, termed *relative-Q*. Specifically, we let LBM-Think generate a CoT c_t using Eq. (3) and send it to the trained LBM-Act to obtain an action $\tilde{a}_t$ via Eq. (4). We then compare this with the dataset action a_t, which represents the ideal generation of a DT method without CoT. The **relative-Q** is calculated as:

$$\Delta Q = Q_\varphi(s_t,\tilde{a}_t) - Q_\varphi(s_t,a_t). \tag{12}$$

If $\Delta Q > 0$, it indicates that CoT has a positive impact on decision-making; otherwise, it suggests a negative impact.

Therefore, to achieve an expected objective in Eq. (11), we propose a stable and flexible method, termed Group relative-Q Policy Optimization (GQPO). Grasping the idea behind GRPO in finetuning an LLM, it is to drive the policy from π_{old} to the near region of optimal policy π^*. We can generate a group of CoTs $\{c_t^i\}_{i=1}^N$ and then assess their values via relative-Q, and then we find the best one c_t^j that satisfies

$$j = argmax_i \Delta Q_i, \quad s.t. \quad \Delta Q_i > 0. \tag{13}$$

Inspired by the AWR method in Eq. (7), we can interpret the CoT c_t as the action and ΔQ as the advantage. Given that we can sample multiple CoTs simultaneously, the regression would be effective and stable when performed directly on the CoT c_t^j with the highest relative-Q. This leads us to the objective of LBM-Think:

$$\begin{aligned}\mathcal{J}_{\mathrm{GQPO}}(\theta) &= \mathbb{E}_{s,a\sim\mathcal{D},c_t^i\sim h_\theta}[\exp(\beta\Delta Q_i)\log h_\theta(c_t|s)] \\ &\propto \mathbb{E}_{s,a\sim\mathcal{D},c_t^j\sim\{c_t^i\}_{i=1}^N}\log h_\theta(c_t^j|s).\end{aligned} \tag{14}$$

After training the LBM-Think, we now give a detailed inference procedure of the LBM for auto-bidding in Fig. 4.

4 Experiment

4.1 Setup

Datasets. To evaluate the performance of bidding strategy decision-making in large-scale ad auctions, we employ AuctionNet and its sparse variant, a publicly available benchmark from Alibaba designed for ad auctions, based on a real-world online advertising platform [42]. Each dataset comprises 21 advertising delivery periods, with each period containing approximately 5,000,000 impression opportunities, divided into 48 intervals. Details are in Appendix A.1.

Evaluation Metrics. To assess performance, we evaluate both the baselines and our method using four metrics:

Table 1: Comparison between our LBM methods and previous non-LLM-based methods in AuctionNet.

AuctionNet		USCB	CQL	IQL	BCQ	DT	DT-Q	DiffBid	DiffBid-Q	LBM(P)	LBM(GQPO)
Dense	Conversions (↑)	267	336	322	321	364	371	316	319	376 ±3.0	**382**±3.3
	Score (↑)	157	171	281	270	329	334	214	260	320 ±2.2	**348**±2.3
Sparse	Conversions (↑)	28.5	30.2	36.0	31.1	31.3	33.8	31.3	31.5	37.4 ±0.3	**38.5** ±0.3
	Score (↑)	17.5	22.2	30.0	30.5	29.6	31.1	23.1	25.1	32.9 ±0.2	**33.4** ±0.2

Table 2: Comprehensive evaluation of LLM-based auto-bidding methods in AuctionNet.

	Method	**Budget Utilization** (↑)		**CPA Ratio** (↓)		**Conversions** (↑)		**Score** (↑)	
		Dense	Sparse	Dense	Sparse	Dense	Sparse	Dense	Sparse
LLM	Prompting	0.743	0.722	**0.878**	0.903	289	24.2	286	23.4
	SFT	0.765	0.741	0.882	**0.804**	297	27.0	286	26.9
	GRPO	0.860	0.773	0.945	0.816	327	29.9	292	29.7
	LLM-DT	0.902	0.861	0.936	0.847	366	36.0	328	32.5
	Prompt-LLM-DT	0.914	0.970	0.931	0.965	355	35.8	322	30.3
	-ours								
	LBM(P)	**0.981**	0.956	1.02	0.901	376	37.4	320	32.9
	LBM(GQPO)	0.938	**0.974**	0.96	0.901	**382**	**38.5**	**348**	**33.4**

- **Conversions**: in the maximize conversions bidding (MCB) task[2], advertisers care about the total received impression value, *i.e.*, number of conventions $\sum_i o_i v_i$ in this task;
- **Budget Utilization**: ratio of the spent budget by the end of the period;
- **CPA Ratio**: advertisers are concerned with whether the strategy's realized CPA, C_{real}, is lower than the CPA constraint, C. Therefore, we use the CPA ratio, defined as ratio = C_{real}/C, to assess this.
- **Score**: by introducing a $penalty = min\{(\frac{1}{ratio})^2, 1\}$, we can get an additional metric as $score = (\sum_i o_i v_i) \times penalty$.

We use ↑ and ↓ to denote the direction of improved performance under these metrics and **bold** the best performance in results.

Baselines. We perform an extensive comparison of our approach against a diverse set of baselines, encompassing both non-LLM-based and LLM-based methods. For non-LLM-based methods, our comparison includes RL techniques such as **USCB** [18], **BCQ** [12], and **CQL** [21]. We also evaluate decision transformer (DT)-based methods, including **DT** and its variant **DT-Q**, which utilizes GAS [25] for fine-tuning a DT with Q-values. Additionally, we consider diffuser-based methods like **Diffbid** [17] and **Diffbid-Q**, which leverage CBD [24] to align a diffuser with a trajectory-level reward model. For LLM-based methods, we compare against a **Prompting** method, a **SFT** method, and a **GRPO** method within language modality. Furthermore, we evaluate **LLM-DT** and Prompt-LLM-DT, which handle numerical sequence inputs to produce numerical actions, with **Prompt-LLM-DT** incorporating an additional language task description.

Implementation Details. The hyperparameters of baselines are set based on the default values referenced in the original papers, with further tuning performed to optimize performance. The LBM-Think model can be based on any pretrained LLMs. In this work, we use Qwen2.5-3B-Instruct for the main results. To train LBM-Think, we utilize 2000 samples obtained through the GQPO method with $N = 3$. We then fine-tune it with a batch size of 64 and a learning rate of 1e-6 for 5 epochs, using the Llama-factory framework under full parameters [55]. The LBM-Act model is built on a smaller LLM, specifically Qwen2.5-0.5B-Instruct [3]. We employ a three-layer MLP with dimensions [896, 896, 896] to process items in the numerical sequence into embeddings that match the size of the token embeddings. The output from the final transformer layer is projected into actions using another three-layer MLP with dimensions [896, 896, 1]. The training of LBM-Act undergoes a total of 400,000 training steps using the AdamW [27] optimizer with a learning rate of 1e-5 and a batch size of 64. The training process is conducted using the PyTorch framework on 8 GPUs. The results presented are averaged over five random runs.

Details of the baselines and implementations can be seen in Appendix A.3.

4.2 Comparison to Auto-Bidding Baselines

We start by comparing our LBM method with established auto-bidding approaches, including offline RL methods and various generative backbones, using Conversions and Scores as metrics. As shown in Table 1, the generative method based on DT exhibits better performance than previous offline RL methods, demonstrating the efficiency of generative models in learning a policy from a large-scale dataset. Without incorporating Q-value, our LBM method variant, referred to as LBM-pretrained (denoted as **LBM(P)** in the results), which employs a pretrained LLM as the LBM-Think model alongside a trained LBM-Act model, could generally outperform previous methods, particularly DT. Even with various budget settings in the test, our LBM method still demonstrates generalizability and consistently outperforms DT, as shown in Table 5. When incorporating Q-value to implement the GQPO method, a fine-tuned LBM-Think integrated into the LBM framework (denoted as **LBM(GQPO)**) further enhances the performance of LBM(P), which demonstrates the advantage of finetuning for LLMs. In summary, the performance comparison across various methods highlights our LBM as a superior generative backbone for auto-bidding.

[2]https://support.google.com/google-ads/answer/7381968?hl=en

One advantage of our LBM method is its ability to leverage the prior knowledge embedded in LLMs, which can be more efficient than relying solely on reward design for training like DT. We utilize the Score metric as an evaluation tool for this purpose. While the current reward design focuses only on conversions, the Score metric offers a more comprehensive evaluation by considering both conversions and the CPA ratio through a penalty term. However, this penalty term is calculated over the entire bidding period, making it challenging to assign accurately to a single-step signal, thus complicating the return-to-go setting in DT methods. Empirically, we propose a more sophisticated return-to-go design for DT to achieve a better Score, expressed as $rtg_w = \sum_{i=t}^{T} r_i + w \times penalty_{t:T}$ where $penalty_{t:T}$ is calculated for future timesteps in the training stage and set to 1 during the inference stage. The parameter w serves as a reweighting factor to balance conversions and the CPA ratio. Intuitively, setting $w = 0$ encourages the DT to prioritize conversions, while $w > 0$ shifts focus towards the CPA ratio. By adjusting this reweighting factor, we can balance conversions and the CPA ratio to achieve a better Score. As demonstrated in Table 3, DT with $w = 0.2$ achieves the highest score. However, the LLM-based method can bypass this inefficient training based on reward design due to its inherent language knowledge and generalization capabilities, allowing it to outperform them.

Table 3: Performance with various reward setting.

	DT-score					LBM
Reweight w	0	0.1	0.2	0.5	1.0	
Budget Utlization (↑)	0.884	0.839	0.818	0.780	**0.632**	0.938
CPA Ratio (↓)	0.895	0.846	0.826	0.831	0.725	**0.960**
Conversions (↑)	364	354	352	329	278	**382**
Score (↑)	329	340	343	323	276	**348**

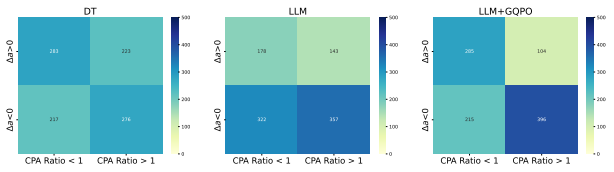
The advantage of prior knowledge could also be illustrated in Fig. 2. A fundamental principle in auto-bidding is to adhere to economic constraints while maximizing conversions. For instance, if the CPA ratio exceeds 1, it is more reasonable and likely to decrease the bidding parameters to meet the constraint. Conversely, when the CPA ratio is below 1, the model can safely increase the bidding parameters to gain more impression opportunities. As shown in Fig. 2, where we randomly sampled 1000 samples, the behavior of the LLM finetuned with GQPO clearly aligns with this principle, whereas DT models show weaker adherence, underscoring the crucial role of reasoning with prior knowledge in auto-bidding.

Table 4: Language instruction's impact on performance.

Instruction	Budget Utilization(↑)	CPA Ratio(↓)	Conversions(↑)	Score(↑)
Increase	**0.969**	1.028	368	315
Decrease	0.894	**0.924**	373	325
base	0.938	0.960	**382**	**348**

4.3 Performance of LLM-based Methods

To apply the LLM for auto-bidding, we initially explored multiple methods within the pure language modality, where the long numerical sequence information is directly translated into language tokens, and the generated action is also represented by language tokens. First, through **Prompting**, we use prompt engineering to apply an LLM to generate the CoT and an action in language format



(a) DT **(b) LLM** **(c) LLM+GQPO**

Figure 2: Relationship between the CPA ratio and model behavior by 1000 random samples. When the CPA ratio exceeds 1, it indicates that the CPA constraint is not satisfied, suggesting a high likelihood of decreasing the bidding parameter, *i.e.*, $\Delta a < 0$. When the CPA ratio is smaller than 1, it will be more likely to increase the bidding parameter $\Delta a > 0$. However, this relationship is not clearly reflected in DT and pretrained LLM, but is more evident in LLM finetuned with GQPO.

Table 5: Generalization to various budget settings. Performance under conversions.

	Budget Ratio	0.5	0.75	1.0	1.25	1.5
Dense	DT	186	234	364	393	445
	LBM(P)	**198**	**285**	**376**	**462**	**520**
Sparse	DT	18.2	27.5	31.3	43.5	50.0
	LBM(P)	**21.3**	**28.5**	**37.4**	**43.6**	**50.6**

without fine-tuning, which can be seen as the RTB-agent method [5]. Second, with **SFT**, we perform supervised fine-tuning to enable the LLM to generate an action directly in language format. Lastly, using **GRPO**, we fine-tune the LLM to generate CoT and action by using the GRPO method to stimulate its reasoning ability, where the reward is the L1 distance between the generated action and the dataset action. However, as shown in Table 2, these three methods perform poorly in auto-bidding that cannot fully utilize the budget, potentially due to the inefficient representation of long sequential state information. Additionally, we observed that GRPO's training is unstable in this task, often resulting in shorter or absent CoT, as shown in Fig. 6. This observation also motivated us to propose a hierarchical structure for an LLM-based method by decoupling reasoning and acting into separate models.

Therefore, we tend to the LLM-DT method, which employs the LLM for decision-making tasks in a continuous state and action space. As it does not contain information from the language, we add a variant to LLM-DT by incorporating the task description in the input embedding, termed Prompt-LLM-DT, making the LLM understand the basic task objective. These LLM-DT methods show a better performance than the above three LLM-based methods in single language modality, demonstrating the efficiency in representing the numerical information via additional embedding layers. However, when compared to our LBM methods, their performance is limited, underscoring the advantages of incorporating reasoning by LBM-Think. This can also be attributed to the efficiency of LBM-Act in integrating the two modalities and utilizing the language reasoning component for decision-making. As demonstrated in the visualizations in Fig. 5 and Fig. 3, LBM-Act exhibits faster

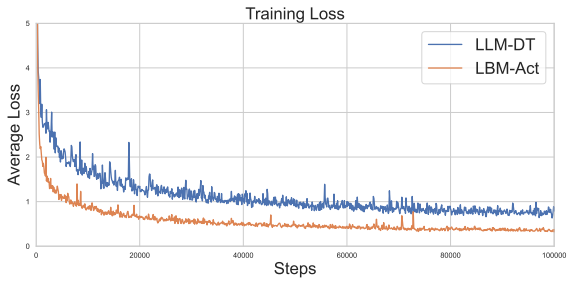


Figure 3: Training loss curve of LLM-DT and LBM-Act, supporting the efficiency of language-guided learning.

Table 6: Impact of LBM-Think's model size.

	Dense		Sparse	
Model Size	Conversions	Score	Conversions	Score
3B	376	320	37.4	32.9
7B	375	320	37.3	34.6
32B	374	318	37.1	33.5

convergence of the training loss compared to LLM-DT and provides informative attention scores. We also included an instruction-following experiment in Table 4, where the reasoning component is instructed to either increase or decrease the bidding parameter. The results demonstrate strong instruction-following capabilities, as a decreased bidding parameter leads to lower budget utilization and a lower CPA ratio. This highlights the flexibility of using language instructions to modify auto-bidding strategies.

Ablation Study. Besides the LLM-DT and Prompt-LLM-DT supporting the importance of the reasoning ability of LBM-Think for auto-bidding, we further evaluate the performance of different LLM choices as the backbone for LBM-Think, including Qwen2.5-3B-Instruct, Qwen2.5-7B-Instruct, and Qwen2.5-32B-Instruct. As the results shown in Table 6, the performance is very close, showing the robustness of the LBM method, and a 3B LLM could be sufficient in handling the auto-bidding task.

5 Related Works

Auto-Bidding Methods. Initially, traditional control methods like PID [8] were employed to optimize budget spending using predefined curves, while OnlineLP [52] adjusted bids based on traffic forecasts. As online bidding environments grew more complex, RL algorithms such as USCB [18], SORL [29], and MAAB [48] became essential for decision-making. Offline RL methods, which learn policies from existing datasets without online interaction, have been particularly successful. Key methods include BCQ [12], which restricts the action space; CQL [21], which regularizes Q-values; and IQL [20], which enables multi-step updates without querying out-of-sample Q-values. However, these methods are limited by the MDP assumption, whereas generative models offer greater potential for auto-bidding. Recently, DiffBid [17], an innovative generative auto-bidding method, has shown the remarkable potential of generative models over RL by directly using a decision diffuser for planning and control, although it has not yet been tested in large-scale auctions. GAS [25] and GAVE [13] propose data augmentation techniques to enhance the quality of offline datasets, but they still rely on the single-step reward. RTB-agent [5] proposes a prompting method based on LLMs for auto-bidding, which we found is limited in large-scale auctions.

LLM for Decision-Making. LLMs have recently been explored as autonomous decision-making agents across domains such as navigate web pages [56], android device control [37] and robotics [33]. Early studies have proposed structured workflows that integrate reasoning, action, and reflection steps such as ReAct [51] and Reflexion [41], as well as interaction through memory [50], retrieval systems [14], and external tools [43]. Recent research has shifted beyond static prompting toward adaptive learning via supervised fine-tuning (SFT) with limited human demonstrations [35]. To further allow LLMs to operate in dynamic environments, RL has become essential for enhancing decision-making capabilities of LLMs. PPO [39] and its variant GRPO [16] are the most widely adopted. Building on this line of work, GiGPO [11] introduces fine-grained credit assignment by leveraging repeated environment states. However, on-policy policy gradient methods often face limitations in real-world agent tasks, where interacting with the actual environment is expensive and inefficient [28]. To overcome this, offline RL approaches have emerged to leverage static experience data. Digi-Q [2] trains VLM-based Q-functions to filter offline action data. LLM-DT [40, 53] finetunes an LLM in a DT manner. However, these methods do not focus on enhancing reasoning ability.

6 Limitations and Conclusion

Due to safety concerns, we focus on offline fine-tuning techniques rather than continual training of LLMs via real-world rollouts. As a result, the extent of performance improvement remains limited. Therefore, a promising future direction is to fine-tune LLMs in real advertising environments with safety control. Additionally, the inference latency could be explored by incorporating more advanced acceleration methods, while we only use vLLM [22] for acceleration in this work. Therefore, our method may face limitations in auto-bidding services that require extremely high-frequency adjustments. However, we emphasize that a common setting involves much longer intervals, which can be as long as 30 minutes, thus providing ample time for LLM-based methods. A discussion and experiment on inference latency can be seen in Appendix A.2

The increasing complexity of ad auctions necessitates advanced auto-bidding strategies, while current methods using offline reinforcement learning or generative approaches face challenges due to their black-box nature and direct applying LLMs is hindered by the need for precise actions and specialized bidding knowledge. In this work, we introduce a hierarchical Large auto-Bidding Model (LBM), comprising LBM-Think for reasoning and LBM-Act for action generation. Our approach utilizes a dual embedding mechanism for efficient modality fusion and introduces GQPO, an offline reinforcement fine-tuning technique, to enhance decision-making without relying on rollouts in the real world. Experiments demonstrate the LBM's excellence in training efficiency and generalization, setting the stage for future auto-bidding approaches based on LLMs.

Acknowledgments

This research is supported by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 (RG18/24).

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. 2023. Is Conditional Generative Modeling all you need for Decision Making?. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023.*

[2] Hao Bai, Yifei Zhou, Li Erran Li, Sergey Levine, and Aviral Kumar. 2025. Digi-Q: Learning VLM Q-Value Functions for Training Device-Control Agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025.* OpenReview.net.

[3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen Technical Report. *arXiv preprint arXiv:2309.16609* (2023).

[4] Santiago R. Balseiro and Yonatan Gur. 2017. Learning in Repeated Auctions with Budgets: Regret Minimization and Equilibrium. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA, USA, June 26-30, 2017.*

[5] Leng Cai, Junxuan He, Yikai Li, Junjie Liang, Yuanping Lin, Ziming Quan, Yawen Zeng, and Jin Xu. 2025. RTBAgent: A LLM-based Agent System for Real-Time Bidding. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025.* ACM, 104–113.

[6] Matteo Castiglioni, Andrea Celli, and Christian Kroer. 2024. Online Learning under Budget and ROI Constraints via Weak Adaptivity. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024.*

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. 2021. Decision Transformer: Reinforcement Learning via Sequence Modeling. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual.* 15084–15097.

[8] Ye Chen, Pavel Berkhin, Bo Anderson, and Nikhil R. Devanur. 2011. Real-time bidding algorithms for performance-based display ad allocation. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011.* ACM, 1307–1315.

[9] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. 2024. A Survey on In-context Learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024.* Association for Computational Linguistics, 1107–1128.

[10] David S Evans. 2009. The online advertising industry: Economics, evolution, and privacy. *Journal of economic perspectives* 23, 3 (2009), 37–60.

[11] Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. 2025. Group-in-Group Policy Optimization for LLM Agent Training. *CoRR* abs/2505.10978 (2025).

[12] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA (Proceedings of Machine Learning Research, Vol. 97).* PMLR, 2052–2062.

[13] Jingtong Gao, Yewen Li, Shuai Mao, Peng Jiang, Nan Jiang, Yejing Wang, Qingpeng Cai, Fei Pan, Kun Gai, Bo An, et al. 2025. Generative Auto-Bidding with Value-Guided Explorations. *arXiv preprint arXiv:2504.14587* (2025).

[14] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* 2, 1 (2023).

[15] Negin Golrezaei, Patrick Jaillet, Jason Cheuk Nam Liang, and Vahab Mirrokni. 2023. Pricing against a Budget and ROI Constrained Buyer. In *International Conference on Artificial Intelligence and Statistics, 25-27 April 2023, Palau de Congressos, Valencia, Spain.*

[16] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[17] Jiayan Guo, Yusen Huo, Zhilin Zhang, Tianyu Wang, Chuan Yu, Jian Xu, Bo Zheng, and Yan Zhang. 2024. Generative Auto-bidding via Conditional Diffusion Modeling. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024.* ACM, 5038–5049.

[18] Yue He, Xiujun Chen, Di Wu, Junwei Pan, Qing Tan, Chuan Yu, Jian Xu, and Xiaoqiang Zhu. 2021. A unified solution to constrained bidding in online display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining.* 2993–3001.

[19] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. 2022. Planning with Diffusion for Flexible Behavior Synthesis. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA (Proceedings of Machine Learning Research, Vol. 162).* PMLR, 9902–9915.

[20] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2022. Offline Reinforcement Learning with Implicit Q-Learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022.*

[21] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.*

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles.*

[23] Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2025. Long-context LLMs Struggle with Long In-context Learning. *Trans. Mach. Learn. Res.* 2025 (2025).

[24] Yewen Li, Jingtong Gao, Nan Jiang, Shuai Mao, Ruyi An, Fei Pan, Xiangyu Zhao, Bo An, Qingpeng Cai, and Peng Jiang. 2025. Generative Auto-Bidding in Large-Scale Competitive Auctions via Diffusion Completer-Aligner. *arXiv preprint arXiv:2509.03348* (2025).

[25] Yewen Li, Shuai Mao, Jingtong Gao, Nan Jiang, Yunjian Xu, Qingpeng Cai, Fei Pan, Peng Jiang, and Bo An. 2024. GAS: Generative Auto-bidding with Post-training Search. *CoRR* abs/2412.17018 (2024).

[26] Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. 2023. Constrained Decision Transformer for Offline Safe Reinforcement Learning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA (Proceedings of Machine Learning Research, Vol. 202).* PMLR, 21611–21630.

[27] Ilya Loshchilov and Frank Hutter. [n. d.]. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations.*

[28] Ziyang Luo, Zhiqi Shen, Wenzhuo Yang, Zirui Zhao, Prathyusha Jwalapuram, Amrita Saha, Doyen Sahoo, Silvio Savarese, Caiming Xiong, and Junnan Li. 2025. Mcp-universe: Benchmarking large language models with real-world model context protocol servers. *arXiv preprint arXiv:2508.14704* (2025).

[29] Zhiyu Mou, Yusen Huo, Rongquan Bai, Mingzhou Xie, Chuan Yu, Jian Xu, and Bo Zheng. 2022. Sustainable Online Reinforcement Learning for Auto-bidding. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.*

[30] S. Muthukrishnan. 2009. Ad Exchanges: Research Issues. In *Internet and Network Economics, 5th International Workshop, WINE 2009, Rome, Italy, December 14-18, 2009. Proceedings (Lecture Notes in Computer Science, Vol. 5929).* Springer, 1–12.

[31] Ashvin Nair, Murtaza Dalal, Abhishek Gupta, and Sergey Levine. 2020. Accelerating Online Reinforcement Learning with Offline Datasets. *CoRR* abs/2006.09359 (2020).

[32] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. *CoRR* abs/1910.00177 (2019).

[33] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. 2025. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747* (2025).

[34] Jan Peters and Stefan Schaal. 2007. Reinforcement learning by reward-weighted regression for operational space control. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007 (ACM International Conference Proceeding Series, Vol. 227).* ACM, 745–750.

[35] Akshara Prabhakar, Zuxin Liu, Ming Zhu, Jianguo Zhang, Tulika Awalgaonkar, Shiyu Wang, Zhiwei Liu, Haolin Chen, Thai Hoang, Juan Carlos Niebles, et al. 2025. Apigen-mt: Agentic pipeline for multi-turn data generation via simulated agent-human interplay. *arXiv preprint arXiv:2504.03601* (2025).

[36] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[37] Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2023. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems* 36 (2023), 59708–59728.

[38] Kan Ren, Weinan Zhang, Ke Chang, Yifei Rong, Yong Yu, and Jun Wang. 2017. Bidding machine: Learning to bid for directly optimizing profits in display advertising. *IEEE Transactions on Knowledge and Data Engineering* 30, 4 (2017), 645–659.

[39] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *CoRR* abs/1707.06347 (2017).

[40] Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon Shaolei Du, and Huazhe Xu. 2024. Unleashing the Power of Pre-trained Language Models for Offline Reinforcement Learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024.* OpenReview.net.

[41] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning.

Advances in Neural Information Processing Systems 36 (2023), 8634–8652.
[42] Kefan Su, Yusen Huo, Zhilin Zhang, Shuai Dou, Chuan Yu, Jian Xu, Zongqing Lu, and Bo Zheng. 2024. AuctionNet: A Novel Benchmark for Decision-Making in Large-Scale Games. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
[43] Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. 2024. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems* 37 (2024), 2686–2710.
[44] Jun Wang and Shuai Yuan. 2015. Real-Time Bidding: A New Frontier of Computational Advertising Research. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining, WSDM 2015, Shanghai, China, February 2-6, 2015*. ACM, 415–416.
[45] Jun Wang, Weinan Zhang, and Shuai Yuan. 2017. Display Advertising with Real-Time Bidding (RTB) and Behavioural Targeting. *Found. Trends Inf. Retr.* 11, 4-5 (2017), 297–435.
[46] Qing Wang, Jiechao Xiong, Lei Han, Peng Sun, Han Liu, and Tong Zhang. 2018. Exponentially Weighted Imitation Learning for Batched Historical Data. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*. 6291–6300.
[47] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.
[48] Chao Wen, Miao Xu, Zhilin Zhang, Zhenzhe Zheng, Yuhui Wang, Xiangyu Liu, Yu Rong, Dong Xie, Xiaoyang Tan, Chuan Yu, Jian Xu, Fan Wu, Guihai Chen, Xiaoqiang Zhu, and Bo Zheng. 2022. A Cooperative-Competitive Multi-Agent Framework for Auto-bidding in Online Advertising. In *WSDM '22: The Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event / Tempe, AZ, USA, February 21 - 25, 2022*. ACM, 1129–1139.
[49] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
[50] Wujiang Xu, Kai Mei, Hang Gao, Juntao Tan, Zujie Liang, and Yongfeng Zhang. 2025. A-mem: Agentic memory for llm agents. *arXiv preprint arXiv:2502.12110* (2025).
[51] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.
[52] Hao Yu, Michael J. Neely, and Xiaohan Wei. 2017. Online Convex Optimization with Stochastic Constraints. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. 1428–1438.
[53] Wenhao Zhao, Qiushui Xu, Linjie Xu, Lei Song, Jinyu Wang, Chunlai Zhou, and Jiang Bian. 2025. Unveiling markov heads in pretrained language models for offline reinforcement learning. In *Forty-second International Conference on Machine Learning*.
[54] Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group Sequence Policy Optimization. *CoRR* abs/2507.18071 (2025).
[55] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*. Association for Computational Linguistics, Bangkok, Thailand.
[56] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854* (2023).

A Appendix

A.1 Dataset Details

The AuctionNet benchmark comprises two datasets: AuctionNet and AuctionNet-sparse. AuctionNet-sparse is a sparser version of AuctionNet, featuring fewer conversions. Each dataset includes 21 advertising delivery periods, with each period containing approximately 5,000,000 impression opportunities, divided into 48 intervals. The models are trained on these two datasets, while 5,000 randomly selected trajectories from each dataset are used exclusively for evaluating the visualization of the generation. Detailed parameters are provided in Table 7.

A.2 Computational Efficiency

We measured the inference latency using vLLM for acceleration on an H800 GPU for all models. As shown in Table 8, the LBM-Think based on the Qwen2.5-3B-Instruct model achieves the fastest speed, while the LBM-Act requires significantly less time. As illustrated in Fig. 4, the LBM-Think has a maximum allowable inference latency, denoted as Δt, which is the interval between two consecutive decision-making timesteps. Given that current industrial auto-bidding methods operate at a low frequency, sometimes as infrequently as every half hour, the LBM-Think of the current inference latency is able to be deployed. However, we emphasize the importance of fully harnessing the potential of smaller LLMs, such as a 3B model, to achieve better computational efficiency.

Table 7: The parameters of AuctionNet and AuctionNet-sparse.

Params	AuctionNet	AuctionNet-Sparse
Trajectories	479,376	479,376
Delivery Periods	9,987	9,987
Time steps in a trajectory	48	48
State dimension	16	16
Action dimension	1	1
Action range	[0, 493]	[0, 8178]
Impression's value range	[0, 1]	[0, 1]
CPA range	[6, 12]	[60, 130]
Total conversion range	[0, 1512]	[0, 57]

Table 8: Inference Latency of different modules.

	LBM-Think			**LBM-Act**
#params	3B	7B	32B	0.5B
Inference Latency	2.5s	3.6s	8.9s	63ms

A.3 Details of Baselines and Implementation

Prompting and **SFT** are two straightforward techniques for processing prompts that include task descriptions and observed sequential information, such as states, previous actions, and return-to-go, to generate actions in the language modality as responses. The **Prompting** method relies solely on prompt engineering, crafting prompts that encapsulate task and sequence details and instruct the model to perform reasoning to elicit desired outputs. In contrast, **SFT** employs a supervised fine-tuning approach, training the model on labeled action data in the language modality to enhance its ability to interpret prompts and produce accurate actions, akin to a Decision Transformer (DT) training manner but in the language modality. As the LLM with SFT has no CoT, we further employ the **GRPO** to fine-tune the LLM for generating CoT and actions, aiming to enhance its reasoning capabilities while keeping action generation ability in a single LLM. Nonetheless, as illustrated in Table 2, this GRPO approach underperforms in auto-bidding, failing to fully

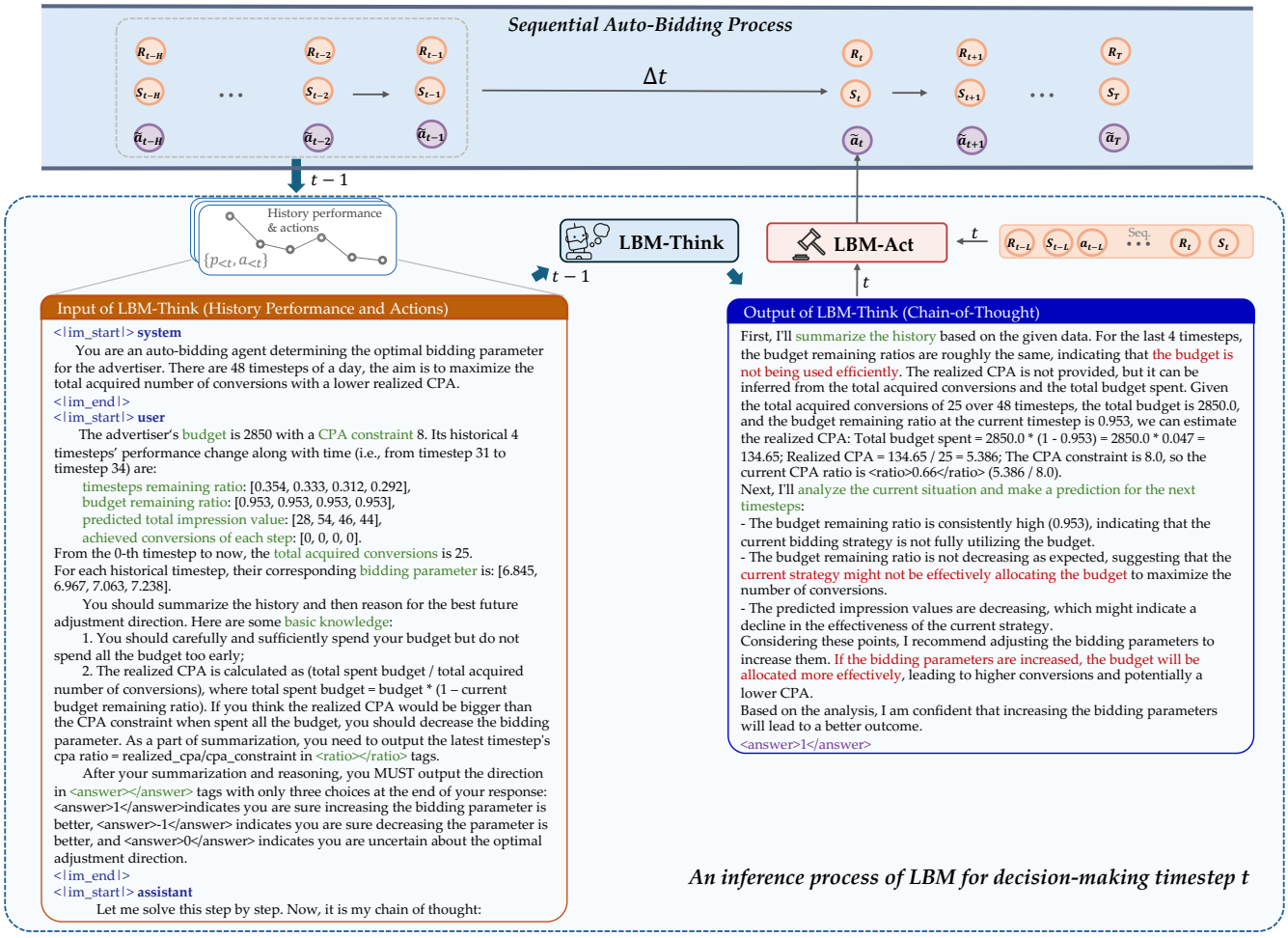
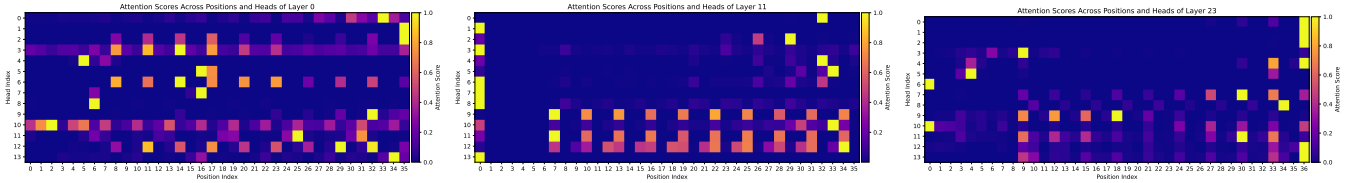


Figure 4: Illustration of the inference procedure of LBM with a detailed case of the prompt and CoT. For a timestep t, the LBM-Think first generates a CoT at timestep $t-1$ within the interval Δt. Then, when timestep t arrives, the LBM-Act receives the pre-generated CoT along with the detailed numerical sequence information to generate an action, which involves adjusting the bidding parameter.



(a) Attention scores of the beginning layer. **(b) Attention scores of the middle layer.** **(c) Attention scores of the final layer.**

Figure 5: Visualization of attention scores varying across different positions and heads. For simplicity, we let positions from #0 to #6 as the language part, while the remaining positions are the numerical part. The two modalities are effectively fused using different attention heads. For instance, heads such as #4, #5, #6, and #10 in the final layer predominantly focus on the language part, whereas other heads are more attuned to the numerical parts. Interestingly, we observed that the attention in the middle layers distinctly separates the language parts (positions 0-6) from the numerical parts, with position 7 marking the beginning of the numerical part.

utilize the budget. Furthermore, we noticed instability in GRPO's training for this task, frequently leading to shorter or missing CoT, as depicted in Fig. 6. To achieve computational efficiency, all three methods are built on Qwen2.5-3B-Instruct.

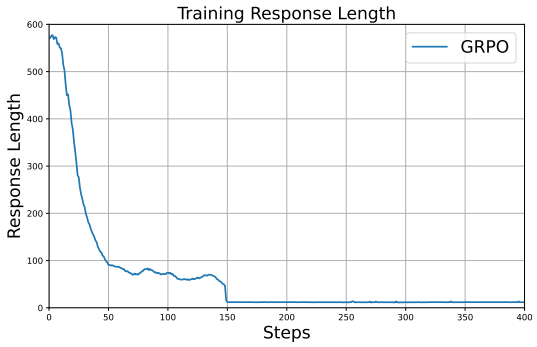


Figure 6: Finetune the LLM using GRPO so that it first performs step-by-step reasoning and then outputs an action. However, after around 150 training steps, the model only outputs the action without performing additional reasoning.

LLM-DT is specifically designed to deploy LLMs for decision-making in continuous action spaces. Its inputs and outputs are the same as those of a Decision Transformer (DT); specifically, the input is a sequence containing return-to-go, states, and actions in numerical modality, and the output is an action in numerical modality. The key distinction between LLM-DT and DT is that LLM-DT utilizes transformer layers directly from a pretrained LLM. Since these transformer layers are trained to process embeddings of language tokens, LLM-DT introduces additional embedding layers to handle numerical sequences. This is achieved using a three-layer multilayer perceptron (MLP), which is the same as our decision embedding layer. In this way, a sequence of n items can be projected into n embeddings of the same size as the token embeddings, providing a highly informative representation, unlike representing a number with multiple token embeddings. LLM-DT's training process is identical to that of a Decision Transformer (DT). **Prompt-LLM-DT** extends LLM-DT by incorporating a task description in natural language to determine if it can enhance decision-making. This language component is encoded using the pretrained token embedding layers of the LLM. The primary difference between these LLM-DT methods and our LBM methods is that LLM-DTs do not incorporate the reasoning capabilities of LLMs.

For a detailed implementation of our LBM method's prompt design, we present a case in Fig. 4. At each decision timestep t , the prompt is generated at timestep $t-1$. This prompt includes key history 4 timesteps' performance metrics such as achieved conversions, remaining budget, and predicted impression value (calculated as the average value of single impression opportunities multiplied by the number of impression opportunities). The LLM is then tasked with determining the CPA ratio using this information, which enhances the LBM-Think's understanding of the prompt in a self-supervised manner and can be used to assess any hallucinations. Finally, the LLM must reason about the adjustment direction for future bidding parameters. Finally, at timestep t, the LBM-Act model receives this CoT and the previous 10 timesteps' numerical sequence information as inputs to generate the final action for execution, as illustrated in Fig. 4.

For detailed information on training the Q-value in the GQPO method, we utilize a transformer that receives a sequence of states and actions as inputs to build the Q-value. The hyperparameter details necessary for reproduction are listed in Table 9. The training method follows the IQL approach [20].

Table 9: The detailed hyperparameters of Q-value networks

Hyperparameters	Value
Batch size	128
Number of steps	400000
Sequence length	10
Learning rate	1e-4
Number of attention layers	6
Number of heads	8
Optimizer	AdamW
Optimizer eps	1e-8
Weight decay	1e-2
Scale	2000
Episode length	48
Hidden size	512
Activation function	ReLU
Gamma	0.99
Tau	0.01
Expectile	0.7

A.4 Discussion on Online Learning and RL

Online Learning (OL) is also an important learning method for addressing auto-bidding problems [4, 6, 15]. OL primarily focuses on minimizing regret by typically selecting actions from a discrete feasible set while adhering to budget and ROI constraints for every timestep [6]. In contrast, RL aims to maximize cumulative rewards by satisfying constraints at the end of the period and can operate in a continuous action space, typically implemented via neural networks, which offers potentially superior solutions. Additionally, OL estimates the optimal dual variables or bids under assumptions like stable and predictable cost and conversion distribution, while RL is free from these optimality assumptions by adjusting the bidding parameters, making it more adaptable to dynamic and complex environments. Moreover, OL typically requires policy refinement through exploration in the environment, which can cause risk concerns, similar to the challenges faced in online RL. In contrast, offline RL directly learns policies from datasets, with state-of-the-art techniques including generative methods like Decision Transformer, Diffusers, and our LBM. For the relationship between LBM and offline RL, intuitively, the core technique of LBM, GQPO, serves as an effective adaptation of AWR, transitioning from common numerical action spaces in offline RL tasks to language spaces, where GQPO treats CoT as the "action".