DARA: Few-shot Budget Allocation in Online Advertising via In-Context Decision Making with RL-Finetuned LLMs

Mingxuan Song
Peking University
School of Computer Science
Beijing, China
songmingxuan@stu.pku.edu.cn

Yusen Huo
Alibaba Group
Beijing, China
huoyusen.huoyusen@alibaba-inc.com

Bohan Zhou
Peking University
School of Computer Science
Beijing, China
zhoubh@stu.pku.edu.cn

Shenglin Yin
Peking University
School of Computer Science
Beijing, China
yinsl@stu.pku.edu.cn

Zhen Xiao*
Peking University
School of Computer Science
Beijing, China
xiaozhen@pku.edu.cn

Jieyi Long
Theta Labs, Inc.
San Jose, USA
jieyi@thetalabs.org

Zhilin Zhang*
Alibaba Group
Beijing, China
zhangzhilin.pt@alibaba-inc.com

Chuan Yu
Alibaba Group
Beijing, China
yuchuan.yc@alibaba-inc.com

Abstract

Optimizing the advertiser's cumulative value of winning impressions under budget constraints poses a complex challenge in online advertising, under the paradigm of AI-Generated Bidding (AIGB). Advertisers often have personalized objectives but limited historical interaction data, resulting in few-shot scenarios where traditional reinforcement learning (RL) methods struggle to perform effectively. Large Language Models (LLMs) offer a promising alternative for AIGB by leveraging their in-context learning capabilities to generalize from limited data. However, they lack the numerical precision required for fine-grained optimization. To address this limitation, we introduce **GRPO-Adaptive**, an efficient LLM post-training strategy that enhances both reasoning and numerical precision by dynamically updating the reference policy during training. Built upon this foundation, we further propose **DARA**, a novel dual-phase framework that decomposes the decision-making process into two stages: a few-shot reasoner that generates initial plans via in-context prompting, and a fine-grained optimizer that refines these plans using feedback-driven reasoning. This separation allows DARA to combine LLMs' in-context learning strengths with precise adaptability required by AIGB tasks. Extensive experiments on both real-world and synthetic data environments demonstrate that our approach consistently outperforms existing baselines in terms of cumulative advertiser value under budget constraints.

CCS Concepts

• **Theory of computation → Algorithmic mechanism design**;
• **Computing methodologies → Reinforcement learning**.

Keywords

Online Advertising, Few-shot Decision, Large Language Models, Reinforcement learning

ACM Reference Format:
Mingxuan Song, Yusen Huo, Bohan Zhou, Shenglin Yin, Zhen Xiao, Jieyi Long, Zhilin Zhang, and Chuan Yu. 2026. DARA: Few-shot Budget Allocation in Online Advertising via In-Context Decision Making with RL-Finetuned LLMs. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Maximizing the cumulative value of winning impressions is a crucial sequential decision-making challenge in online advertising, where the system must automatically generate bidding decisions on behalf of advertisers to optimize long-term performance [26, 35]. Under the emerging paradigm of AI-Generated Bidding (AIGB), A growing line of work has advocated for decomposing bidding strategies into budget allocation and bid decision, treating the former as a global planning task across time periods [7, 9, 11]. This separation allows the system to first produce high-level budget plans and then optimize bid decisions within the budget constraints, well suited for this task [1]. In budget allocation task, advertisers must determine how to allocate limited budgets across multiple time periods to maximize return on investment (ROI), while dynamically adapting to temporal cost-reward patterns from shifting user behavior and dynamics of online environments [22].

This problem exhibits several key characteristics—sequential dependency, delayed rewards, and non-stationary environment dynamics—that make Reinforcement Learning (RL) particularly well

*Corresponding author.

WWW '26, April 13–17, 2026, Dubai, United Arab Emirates

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

suited [19]. RL methods, including Q-learning [16, 19] and hierarchical RL [7, 36], have been introduced to address these challenges by learning optimal allocation strategies from historical interactions. In advertising applications, advertisers often have personalized goals and dynamic advertiser preferences, which naturally give rise to few-shot or cold-start scenarios where only limited data is available for each task [9, 32]. However, RL-based methods typically require extensive environment interactions and large volumes of task-specific data to train. Therefore, the resulting policies often fail to generalize well and cannot adapt quickly when faced with new or dynamically changing environments [6, 25, 27].

Recent advances in large language models (LLMs) have opened new possibilities for decision making in low-data regimes [21, 37]. The in-context learning of LLMs is to perform new tasks by conditioning on a few annotated examples presented in the input prompt. This ability allows LLMs to rapidly adapt to new tasks with limited data, making them especially suitable for personalized budget planning in data-scarce AIGB environments [31]. However, despite their flexibility in few-shot prompting, LLMs often exhibit insufficient numerical sensitivity, leading to a lack of the fine-grained precision necessary for structured optimization tasks such as budget allocation [10, 15]. Furthermore, our ablation studies in Section 5.3 reveal that the in-context learning capability of LLMs is limited when applied to complex decision-making tasks: a single-stage prompting setup struggles to both capture underlying few-shot data regularities and execute fine-grained budget optimization. These limitations underscore the need for a new framework that combines the generalization strength of LLMs with the numerical precision and adaptability of reinforcement learning (RL), thereby enabling more robust and effective budget allocation under realistic constraints.

To bridge this gap, we revisit the structure of the budget allocation task and observe that it naturally decomposes into two sub-problems with distinct characteristics. The first phase involves generalizing from limited historical patterns to derive high-level allocation intentions, while the second phase requires fine-grained adjustments based on dynamic environment feedback. Existing approaches either treat the task as a single phase optimization problem or rely on monolithic RL training, failing to capture this structural separation [7, 33, 34]. We posit that an effective solution should explicitly separate the generalization and optimization processes, aligning model capabilities with the distinct demands of each phase. In addition, RL can be employed to fine-tune the LLM, further enhancing its reasoning capacity and optimization precision across phases [14, 20]. To overcome the limited amount of few-shot data, we draw inspiration from the idea of dynamics randomization in sim-to-real learning [28], and design a simulation environment tailored for budget allocation. This environment is modeled after real-world data distributions and is capable of continuously generating diverse allocation scenarios.

Based on the above analysis, we propose DARA, a novel **D**ual-phase **A**daptive **R**easoning and **A**llocation framework for few-shot budget allocation in online advertising under the AIGB paradigm. We decompose the task into two phases and accordingly design a dual-agent architecture: a **Few-shot Reasoner** that generates initial budget plans based on few-shot in-context data, and a **Fine-grained Optimizer** that incrementally refines the plan through feedback-driven reasoning. This design follows the principle that early-phase decisions benefit from few-shot generalization, while later-phase refinements require numerically sensitive, feedback-aware adjustments.

Moreover, purely prompt-based reasoning in LLMs often lacks numerical sensitivity and fine-grained control [23]. To overcome this limitation, we introduce a reinforcement learning fine-tuning strategy to enhance the decision quality of both LLMs in our framework. We propose GRPO-Adaptive, which periodically updates the reference model during training. By comparing the current policy against a more recent baseline rather than a fixed one, GRPO-Adaptive enables more flexible and effective policy improvements.

We also integrate a dual-environment setup into our method: (1) a real-world environment derived from enterprise-scale advertising data, and (2) a simulation environment designed based on controllable polynomial functions. The simulation environment enables the continuous generation of diverse allocation scenarios, enhancing policy robustness through exposure to varied conditions. Experiments show that DARA consistently outperforms baselines.

In summary, our contributions are as follows.

- We introduce DARA, a dual-phase LLM-based architecture for budget allocation, enabling robust and interpretable decision making under few-shot conditions.
- We propose GRPO-Adaptive, a novel RL algorithm that dynamically updates the reference model to enhance the reasoning ability of LLM.
- We design a simulation environment that mimics real-world allocation dynamics and continuously generates diverse budget scenarios. This simulation enables the model to learn robust and generalizable policies from limited data.

2 Background and Related Work

2.1 Preliminaries

In Real Time Bidding (RTB), advertisers aim to maximize the cumulative value of winning impressions within a fixed budget B. Let b_t denote the budget allocated to period t, and $v_t(b_t)$ denote the expected return when allocating a budget of b_t in time period t. Then, the ROI is:

$$\text{ROI} = \frac{\sum_{t=1}^{T} v_t(b_t)}{\sum_{t=1}^{T} b_t} = \frac{\sum_{t=1}^{T} v_t(b_t)}{B} \tag{1}$$

Since B is fixed, maximizing ROI is equivalent to maximizing the total return $\sum_{t=1}^{T} v_t(b_t)$ under the constraint $\sum_{t=1}^{T} b_t = B$.

Assumption. Each function $v_t(b_t)$ is assumed to be differentiable, strictly increasing, and concave in b_t. This implies that the marginal ROI, given by $v_t'(b_t)$, is positive and strictly decreasing with respect to budget—capturing the realistic effect of diminishing returns.

Optimality Condition. Under the above assumptions, the optimal allocation satisfies that the marginal ROI is equal across all time periods:

$$v_1(b_1^*) = v_2(b_2^*) = \cdots = v_T(b_T^*) \tag{2}$$

where b_t^* denotes the optimal budget for period t.

Proof Sketch. Suppose, for contradiction, that there exist two periods i and j such that $v_i'(b_i) > v_j'(b_j)$. By shifting a small amount

of budget $\delta > 0$ from period j to period i, the increase in total return is:

$$v_i(b_i + \delta) - v_i(b_i) > v_j(b_j) - v_j(b_j - \delta) \tag{3}$$

due to $v_i'(b_i) > v_j'(b_j)$ and concavity. This strictly increases the total return under the same budget, contradicting optimality. Therefore, all marginal ROIs must be equal at optimality. The complete proof is provided in Appendix A. Accordingly, maximizing expected return under a fixed budget can be approximated by minimizing the variance of marginal ROI across time periods.

2.2 Related Work

Budget allocation has long been a core problem in auction-based advertising systems, where advertisers aim to distribute limited budgets to maximize their overall return on investment (ROI) [3, 5]. Early approaches in this domain typically rely on hand-crafted heuristics or rule-based strategies, which often fail to generalize across dynamic environments and complex auction conditions [13, 24]. To address these limitations, reinforcement learning (RL) has been widely adopted in recent years to learn optimal allocation strategies through interactions with the environment [7, 16, 19, 36]. For example, in Q-MCKP [19], the budget space is discretized into multiple bins, and Q-value functions are learned independently for each stage. A multi-choice knapsack problem is then solved to determine the final allocation. This stage-wise independence overlooks potential interactions between time periods, which may be critical for achieving long-term budget efficiency. HiBid [36] adopts a hierarchical RL framework where a high-level policy allocates budget across time periods, and a low-level bidding agent learns to bid effectively within each period under budget constraints. However, it lacks mechanisms for continuous online adaptation or rapid fine-tuning in dynamic environments. Similarly, ABPlanner [7] employs an LSTM-based RL planner to allocate budget across time periods. As an RL approach, it struggles to perform well under limited data and shifting advertiser conditions.

Recent advances in large language models (LLMs) have shown promising capabilities in low-data regimes through in-context learning [4, 17, 18]. Conditioning on a few examples, LLMs can perform structured decision-making tasks without explicit gradient updates [30]. These properties have inspired the application of LLMs to planning, control, and economic optimization tasks, where generalization and reasoning are critical. However, vanilla prompting of LLMs often lacks the numerical sensitivity required for precise budget allocation. To address this, DPO [2, 29] has been proposed. DPO aligns LLM behavior with human preferences by learning from pairwise comparisons between preferred and rejected responses, offering better generalization in under-constrained tasks. More recently, Group Relative Proximal Optimization (GRPO) [8] extends this idea to structured decision-making settings by introducing group-wise KL regularization, enabling more stable and sample-efficient LLM fine-tuning under noisy or feedback-driven supervision.

3 LLM-based Dual-phase Decision Making

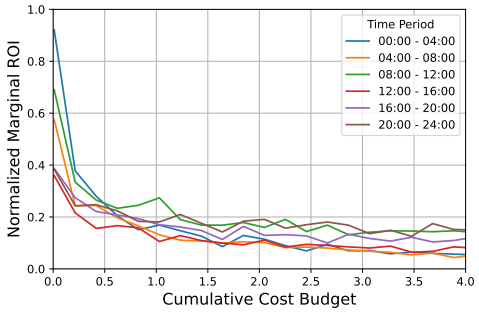
3.1 Environment Modeling

To address the challenges of personalized budget planning, we incorporate a modeling strategy to support robust and generalizable policy learning under limited data, as illustrated in Figure 1. The rationale lies in the empirical statistics derived from real-world data: due to the cost-value curve of budget, the ROI slope typically decreases with increased budget. In addition, different time periods exhibit varying ROI patterns, making dynamic allocation essential. This prototype builds upon empirical patterns observed in online advertising data and provides two distinct environment modeling approaches:

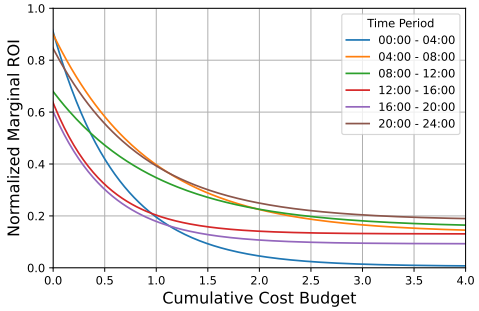
- **Real-World Data Environment**: This environment is constructed directly from real-world online advertiser data, ensuring that environmental features, cost consumption behavior, and ROI dynamics align closely with actual scenarios. An example environment of the resulting marginal ROI curves is illustrated in Figure 1a.
- **Synthetic Data Environment**: Given the limitations of real-world data, such as the inability to flexibly extend to budget strategies across varying time periods, we construct a simulation environment based on functional modeling to replicate the relationship between budget allocation and marginal ROI. This approach enables controlled experimentation and improves the generalizability of learned policies. As illustrated in Figure 1b, the marginal ROI function *MROI* is designed to decrease monotonically for budget $b > 0$ until it reaches zero at a predefined budget:

$$MROI_i(b) = \max\{F_i(b),\ 0\},\ 0 \le b \le B,\ i = 1\ to\ T, \tag{4}$$

where $F_i(b)$ can be either a polynomial or an exponential function, B is the total budget, and T is the number of periods. This function is continuous and differentiable, making it well suited for optimization and analytical analysis.



(a) Real-World Data Environment.



(b) Synthetic Data Environment.

Figure 1: Data Environment.

These environmental modeling strategies provide two key advantages: (1) High fidelity to real-world settings: the cost and ROI patterns align closely with actual market behaviors, ensuring practical relevance. (2) Performance benchmarking: budget ceilings are manually defined to establish clear evaluation baselines, facilitating comparisons across models and policies. Additionally, the environment supports flexible adjustment of sequence lengths and environment parameters, enabling in-depth analysis of model behaviors under varying levels of complexity.

3.2 Problem Formulation

We formulate the few-shot budget allocation task as follows. Given a few-shot dataset $\mathcal{H}$ of n previous episodes:

$$\mathcal{H} = \left\{ \left(\mathbf{b}^{(i)}, \mathbf{m}^{(i)} \right) \right\}_{i=1}^{n}, \tag{5}$$

$$\mathbf{b}^{(i)} = \left(b_1^{(i)}, \ldots, b_T^{(i)} \right), \tag{6}$$

$$\mathbf{m}^{(i)} = \left(MROI_1^{(i)}, \ldots, MROI_T^{(i)} \right) \tag{7}$$

where each $\mathbf{b}^{(i)}$ is a budget allocation vector across T time periods and $\mathbf{m}^{(i)}$ is the corresponding marginal ROI vector, the goal is to generate a new allocation $\mathbf{b} = (b_1, \ldots, b_T)$ that satisfies the total budget constraint:

$$\sum_{t=1}^{T} b_t = B, \quad b_t \geq 0, \tag{8}$$

and minimizes the variance of the resulting T $MROI$. This objective must be achieved with a minimal number of exploration trials, reflecting the few-shot constraint inherent in practical online advertising scenarios.

3.3 Few-shot Prompting

In advertising budget planning tasks, data is limited and deployment scenario changes dynamically. LLMs possess strong language understanding and generalization capabilities. They can complete complex tasks with only a few contextual examples, offering a promising alternative for handling small-sample budget allocation challenges. We propose a few-shot prompting framework that encodes historical episode data into prompts, allowing the model to generate effective strategies for current allocation tasks.

To support task comprehension and improve response consistency, we design a structured prompt format that explicitly includes the following components: the task objective, few-shot data, trial records, and a clear output format specification. The objective section defines the goal of minimizing reward variance across periods while strictly adhering to a total budget constraint. The historical data block $\mathcal{H}$, which serves as few-shot data collected from advertisers, provides examples of previous budget allocation records and their associated ROI outcomes, helping the model learn from past patterns. The records block contains previously attempted strategies to guide policy refinement. The output specification section instructs the model to return a concrete allocation vector along with reasoning for the proposed strategy, ensuring interpretability and executability. Table 1 illustrates a concrete implementation example of our prompting approach. Two complete examples of such prompts are provided in the Appendix B.

```
You are assigned a budget allocation task consisting
of {NUM} time periods. Each time period corresponds
to a specific reward value. It is known that
the relationship between budget and reward is
approximately monotonically decreasing... <few-shot>
<trial_records> <objective> <output_format>
```

Table 1: Template for Few-shot Prompting. {NUM} will be replaced with the specific number of periods.

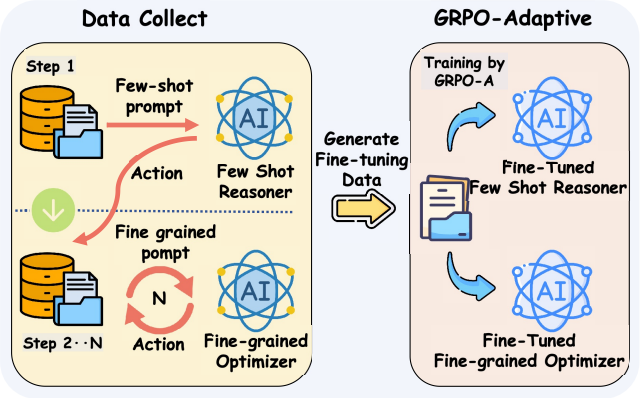


Figure 2: The training workflow of DARA.

3.4 Multi-Agent Paradigm

In budget allocation tasks, we initially attempted to use a single LLM to perform direct decision-making for few-shot budget planning. However, we found that a single LLM struggles to simultaneously achieve rapid few-shot learning and precise policy modeling. This limitation was clearly validated in our ablation studies in Section 5.3, which show that relying solely on a single LLM leads to performance ceilings in complex decision-making scenarios.

To overcome the limitations of using a single LLM for end-to-end budget planning, we propose a Dual-Phase Cooperative Agent Architecture that decomposes the task along the temporal axis into two distinct phases: the first few-shot reasoning phase and subsequent decision-making phases, as illustrated in Figure 2. This design is motivated by the intuition that different stages of budget allocation require different reasoning capabilities—initial planning relies heavily on few-shot generalization, while later stages require fine-grained optimization based on early outcomes. Accordingly, we introduce two independent yet complementary LLM agents:

Few Shot Reasoner is responsible for the first episode budget allocation based on a few-shot prompt. It utilizes a small set of historical budget allocation episodes to form compact and informative demonstration sequences. Through few-shot prompting, it produces an initial allocation vector that indicates how the budget should be proportionally distributed across different time periods. This layer emphasizes capturing high-level strategy and distribution trends.

Fine-grained Optimizer focuses on refining and executing the first episode plan in a fine-grained manner. It takes as input the proposed allocation from Few Shot Reasoner and marginal ROI indicators for each time period, and aims to locally optimize the budget to maximize marginal ROI. Unlike Few Shot Reasoner, the executor is trained to be more responsive to adjustments in budget allocation. We also introduce a **sliding window mechanism** that allows Fine-grained Optimizer to dynamically adjust its strategy based on the most recent episodes' feedback records.

Algorithm 1: Dual-Phase Cooperative Budget Allocation

Input: Few-shot historical episodes $\mathcal{H}$, budget constraint B, periods T, sliding window size w
Output: Allocation trajectory $\mathcal{B} = \{b^{(1)}, b^{(2)}, \ldots, b^{(i)}, \ldots\}$, where $b^{(i)} \in \mathbb{R}^T$

Initialize: $\mathcal{B} \leftarrow [\,]$, episode index $s \leftarrow 1$
// Step 1: Few-shot Initialization via Few Shot Reasoner
Generate prompt $\mathcal{P}_1 \leftarrow \text{Encode}(\mathcal{H})$
Initial allocation $b^{(1)} \leftarrow$ Few Shot Reasoner$(\mathcal{P}_1)$
Append $b^{(1)}$ to $\mathcal{B}$
Observe *MROI* feedback $r^{(1)}$
$s \leftarrow s + 1$
// Initialize sliding window $\mathcal{W}_2$
Extract the latest $(w-1)$ records from $\mathcal{H}$ as $\mathcal{H}_{\text{tail}}$
$\mathcal{W}_2 \leftarrow \mathcal{H}_{\text{tail}} \cup \{(b^{(1)}, r^{(1)})\}$
// Step 2: Fine-Grained Refinement via Fine-grained Optimizer
while ***True*** **do**
  Generate prompt $\mathcal{P}_t \leftarrow \text{Encode}(\mathcal{W}_s)$
  Refined allocation $b^{(s)} \leftarrow$ Fine-grained Optimizer$(\mathcal{P}_t)$
  Append b^s to $\mathcal{B}$
  Observe marginal ROI feedback $r^{(s)}$
  Update sliding window history $\mathcal{W}_{s+1}$ with $(b^{(s)}, r^{(s)})$
  $s \leftarrow s + 1$
  if *termination condition met* **then**
    break
return $\mathcal{B}$

The overall workflow is summarized in Algorithm 1. The core advantage of this two-tier architecture lies in the clear separation of concerns: the Few Shot Reasoner handles global strategic reasoning and trend estimation, while the Fine-grained Optimizer continuously refines the plan based on feedback. To further enhance LLMs' reasoning capabilities, we fine-tune these two LLMs using our proposed reinforcement learning method, enabling them to generate strong initial solutions even with limited training samples.

4 RL Fine-tuning Strategies

4.1 GRPO-Adaptive

4.1.1 Optimization Objective. Given an input prompt x, the language model defines a policy $\pi_\theta(b \mid \mathcal{P})$ that generates an output sequence $b = (b^1, b^2, \ldots, b^T)$ of the period length T, where the quality of the output is evaluated by a reward function $R(b)$. We define the objective as the expected discounted cumulative return under a given budget cost constraint:

$$\max_\theta J(\theta) = \mathbb{E}_{\{b^{(s)}\}\sim\pi_\theta(\cdot|\mathcal{P})}\left[\sum_{s=1}^{\infty}\gamma^{s-1}R(b^{(s)})\right] \text{ s.t. } \forall s, \sum_{i=1}^{T} b_i^{(s)} = B. \tag{9}$$

where B is the allocation budget. The discount factor $\gamma \in (0, 1]$ controls the weight of future rewards.

While KL regularization is crucial for constraining policy drift in GRPO [8], our experiments in Section 5.4.2 reveal a key limitation: when the reference policy π_{θ_0} is kept static throughout training, the model's sensitivity to structured reasoning and symbolic precision gradually deteriorates. This becomes a performance bottleneck, especially in tasks requiring rigorous multi-step reasoning.

To address this, we propose **GRPO-Adaptive (GRPO-A)**, an enhanced variant of GRPO where the reference model is periodically updated to reflect the latest policy improvements. Specifically, GRPO-A operates as follows:

- After every K GRPO updates, we take a snapshot of the current policy π_θ,
- Replace the static reference π_{θ_0} with this updated version,
- Reset the KL regularization to reflect divergence from the new baseline.

The resulting optimization objective becomes:

$$\max_\theta J_{\text{GRPO-A}}(\theta) = J(\theta) - \beta\,\mathbb{E}_{\mathcal{P}}\left[\mathbb{D}_{\text{KL}}\left(\pi_\theta(\cdot \mid \mathcal{P}) \,\|\, \underbrace{\pi_{\theta_{\text{ref}}}(\cdot \mid \mathcal{P})}_{\text{periodically updated}}\right)\right], \tag{10}$$

where β controls the trade-off between reward maximization and policy stability, and $\mathbb{D}_{\text{KL}}[\cdot\|\cdot]$ is the KL divergence.

This dynamic adjustment allows the model to continue learning from its improved reasoning abilities while still benefiting from KL stabilization. It prevents over-regularization by outdated references and promotes a more adaptive, stable optimization process in later-stage fine-tuning.

4.1.2 Optimization loss. To practically optimize the policy, we adopt a clipped advantage-weighted objective as used in GRPO. For a batch of G trajectories, the loss is:

$$\mathcal{L}_{\text{GRPO-A}}(\theta) = -\,\mathcal{L}_{\text{adv}}(\theta) + \beta\,\mathbb{D}_{\text{KL}}\left[\pi_\theta \,\|\, \pi_{\text{ref}}\right] \tag{11}$$

$$\mathcal{L}_{\text{adv}}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|o_i|}\sum_{t=1}^{|o_i|}\min\left(\frac{\pi_\theta(o_{i,t} \mid \mathcal{P})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid \mathcal{P})}\hat{A}_{i,t},\ \text{clip}\left(\frac{\pi_\theta(o_{i,t} \mid \mathcal{P})}{\pi_{\theta_{\text{old}}}(o_{i,t} \mid \mathcal{P})}, 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\right) \tag{12}$$

$$\hat{A}_i = \frac{r_i - \mu_r}{\sigma_r}, \quad \text{with} \quad \mu_r = \frac{1}{G}\sum_{j=1}^{G} r_j, \ \ \sigma_r = \sqrt{\frac{1}{G}\sum_{j=1}^{G}(r_j - \mu_r)^2}. \tag{13}$$

Here, $\pi_{\theta_{\text{old}}}$ is the previous policy snapshot, π_{ref} is the reference policy updated periodically, and ϵ is the clipping parameter.

4.2 Fine-tuning Procedure with GRPO-Adaptive

To simulate budget allocation in a realistic yet diverse training setting, we design a controlled multi-environment sampling procedure. Within each environment instance, we simulate a single episode budget allocation process. After a fixed number of sampling steps, the model switches to a new environment to emulate a different budget planning scenario. This avoids contamination between training and evaluation phases and encourages the model to generalize across settings rather than memorize a fixed dataset.

Every fixed S steps, we randomly resample or procedurally generate a new environment, which includes T novel marginal ROI curves. The objective is to ensure that the model learns a generalizable ability to identify better solutions, rather than overfitting to any single distribution. Once the model exhibits stable performance in simulation, it can be deployed directly in real environments.

In Equation 12, for each prompt, we sample G candidate outputs in parallel within each environment. The "preferred" output is selected from feasible completions with higher reward. Using normalized group-wise advantage estimation and a clipped PPO-style objective, we update the policy to increase the relative probability of high-quality completions while suppressing low-reward outputs. After each round, the best-performing candidate is added to the trajectory history and the sampling-update cycle is repeated until S total steps are completed. The overall training process is formally described in Algorithm 2 in Appendix C.

4.3 Policy Interface and Reward Design

In our framework, two LLMs are coordinated: the first LLM provides an initial allocation strategy on the first day, while the second LLM refines and outputs updated allocation vectors from the second day onward. Despite differing prompt formats, both models share a unified action interface. This section introduces the formal design of input state representations, output actions, and reward functions tailored to budget allocation.

4.3.1 State and action format. The state is defined as the input prompt $\mathcal{P}$, which includes the task description as well as the sequence of optimal allocations selected in previous steps. The action is defined as a budget allocation vector of length N:

$$b = [b_1, b_2, \ldots, b_T], \tag{14}$$

where b_i denotes the budget assigned to the i-th period.

To extract the action from the LLM output, we first identify the content enclosed in `<answer>` … `</answer>`, and then use a regular expression to extract the vector pattern $[b_1, b_2, \ldots, b_T]$.

4.3.2 Reward Function. A major challenge in LLM-based budget allocation lies in the numerical sensitivity and structural complexity of the task. Small deviations in token-level outputs can lead to substantial losses in overall performance, and the underlying reward landscape is often non-convex and highly structured. To mitigate these issues, we design a composite reward function that incorporates both structural decomposition and auxiliary shaping mechanisms to enhance learning stability and policy robustness.

For each action $b \in \mathbb{R}^T$, the core reward signal $R(b)$ is composed of three terms. The first term, the environment reward R_{env}, encourages consistent marginal returns across time segments. Specifically, let the marginal reward at time segment i be $r_i = MROI_i(p_i)$, and let the average marginal reward be $\bar{r} = \frac{1}{N}\sum_{i=1}^{N} r_i$. We define the environment reward as:

$$R_{\text{env}}(b) = -\alpha \sum_{i=1}^{N} |r_i - \bar{r}|, \tag{15}$$

where α is a scaling coefficient controlling the model's sensitivity to reward variance.

To ensure the validity of generated actions, we define a constraint penalty $R_{\text{constraint}}$ that enforces two conditions: the budget vector must have the correct dimensionality and the total allocation should be close to the total budget B. This penalty is defined as:

$$R_{\text{constraint}}(b) = \begin{cases} -M, & \text{if } \dim(b) \neq N, \\ -\left|\sum_{i=1}^{N} p_i - B\right|, & \text{otherwise.} \end{cases} \tag{16}$$

In addition, we introduce a bonus reward term R_{bonus} to encourage the model to refine its allocation strategy based on last historical performance adaptively. Let $c \in \mathbb{R}^N$ be the last allocations. We define two subsets of time segments: $\mathcal{H}_+$ for those with high historical rewards, and $\mathcal{H}_-$ for those with low historical rewards. Given a minimum adjustment threshold $\delta > 0$ and a clipping bound $\tau > 0$, the bonus reward for each segment is computed as follows. For each $b^i \in \mathcal{H}_+$, if the current allocation $b^i > c^i + \delta$, we assign a positive adjustment reward:

$$R^+_{\text{bonus},i} = \min\left(|b^i - c^i|, \tau\right).$$

For each $i \in \mathcal{H}_-$, if $b^i < c^i - \delta$ and $b^i > 0$, we also assign a complementary bonus:

$$R^-_{\text{bonus},i} = \min\left(|b^i - c^i|, \tau\right).$$

The total bonus reward is given by:

$$R_{\text{bonus}}(b) = \sum_{b^i \in \mathcal{H}_+} R^+_{\text{bonus},b^i} + \sum_{b^i \in \mathcal{H}_-} R^-_{\text{bonus},b^i}.$$

Finally, the full scalar reward used to guide policy updates is defined as the sum of all components:

$$R(b) = R_{\text{env}}(b) + R_{\text{constraint}}(b) + R_{\text{bonus}}(b). \tag{17}$$

5 Evaluation

5.1 Experimental Setup

This study evaluates the model in two types of environments.[1] The first is the **real data environment**, where we use online advertising data from a leading global e-commerce platform to construct a highly realistic simulation of the advertisers. This environment reflects the dynamic variations in the ad allocation process, ensuring that the experimental results have strong practical relevance. The second type is the **synthetic data environment**. We design a series of *MROI* functions with diminishing marginal returns to test the model's generalization ability in a controlled setting. These two environments are designed to assess the model's performance and robustness from different perspectives. Each episode corresponds

[1] Our implementation is available at https://github.com/mx-song/DARA

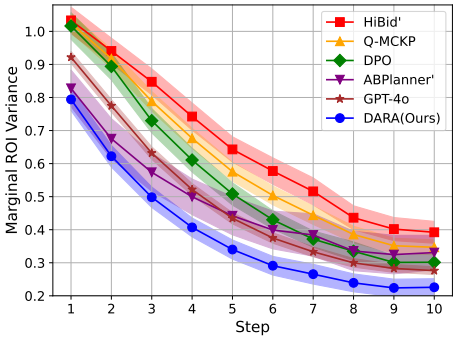


Figure 3: Performance of different algorithms in reducing marginal ROI variance.

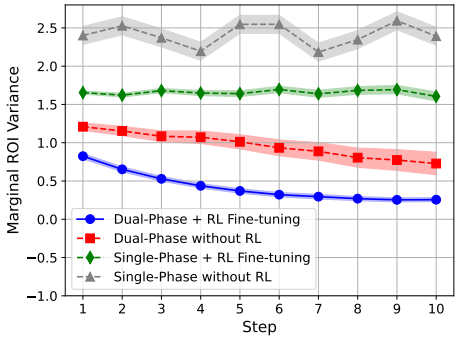


Figure 4: Impact of dual-phase architecture and RL fine-tuning on marginal ROI variance.

to a single day's budget allocation across multiple time periods in real-world ad planning.

All training tasks were conducted on enterprise-level private servers with hardware configuration of 8 NVIDIA H20 GPUs (each with 96GB of memory). The baselines and main hyperparameters used in the experiments are listed in the Appendix D. To ensure the reliability of the results, each experiment was repeated five times, and the mean performance along with the 95% confidence interval is reported. The shaded areas in the plots represent these confidence intervals. The **variance of MROI** measures the balance of allocation across time periods and reflects the stability and uniformity of the budget distribution.

5.2 Result Analysis

After training on the simulation environment, we evaluate the performance of our method in the real-world data environment. As shown in Figure 3, our method significantly outperforms all baseline algorithms across all steps in terms of reducing the marginal ROI variance. This demonstrates the strong stability and adaptability of our approach in dynamic budget allocation environments.

Compared to DPO, which rely on either supervised or preference-aligned fine-tuning, our method exhibits consistently lower variance, indicating a more balanced and effective distribution across time periods. Notably, while DPO improves generalization via token-level preference alignment, this approach is fundamentally limited by its static modeling structure and lack of dynamic feedback responsiveness. Moreover, DARA demonstrates particularly significant improvements in reducing marginal ROI variance during the later stages of allocation. This advantage arises from its fine-grained optimization phase, where recent allocation outcomes provide more reliable guidance than initial few-shot exemplars. By explicitly separating early generalization from late-stage adaptation, our dual-phase design enables the model to focus on precise, context-aware refinements—an ability that single-stage baselines inherently lack.

Compared with ABPlanner', which also uses RL for budget strategy design, our method achieves better variance reduction, especially in later steps. This improvement is attributed to our method's enhanced numerical sensitivity and the structured separation, which allows each LLM to specialize and avoid underfitting or overgeneralization.

5.3 Ablation Performance

To investigate the effectiveness of each component in our framework, we conduct a comprehensive ablation study with four different configurations. The corresponding results are shown in Figure 4.

We observe that directly applying a single LLM to perform end-to-end budget planning yields the poorest performance, exhibiting the highest marginal ROI variance throughout the entire allocation time horizon. This clearly reflects the inherent limitations of large language models in handling numerically sensitive and temporally dynamic tasks. Although LLMs demonstrate strong few-shot reasoning capabilities, they tend to generalize poorly in complex allocation scenarios where subtle reward shifts need to be captured and responded to accurately.

Introducing RL fine-tuning into the single-phase setup offers a slight improvement, confirming that reinforcement learning can enhance the numerical reasoning ability of the model to some extent. However, the gain remains limited. This is because the single LLM is still responsible for both global few-shot reasoning and local decision optimization, leading to interference and capacity bottlenecks when faced with long-horizon, structured decisions.

In contrast, when we adopt the dual-phase architecture without RL, the performance improves substantially and consistently. By explicitly separating the reasoning process into two specialized LLM agents—one focusing on early-stage few-shot generalization and the other on late-stage fine-grained adaptation—the model is better able to handle the heterogeneous requirements of the task. This confirms our hypothesis that task decomposition is critical for complex budget allocation.

Finally, our full model, which incorporates RL fine-tuning into both LLMs, achieves the best overall performance. The Few Shot Reasoner benefits from RL by learning to generate more strategic and generalizable initial allocation plans, while the Fine-grained Optimizer becomes more numerically sensitive and responsive to feedback, thanks to the reinforcement signal. The collaboration between these two specialized agents allows the model to learn both abstract patterns and local refinements, resulting in a significantly lower variance and more stable performance. The performance boost brought by RL is significantly amplified when applied within the dual-phase architecture. While RL alone offers modest improvements in the single-phase setup, its effect becomes substantially more pronounced when each LLM is structurally disentangled and

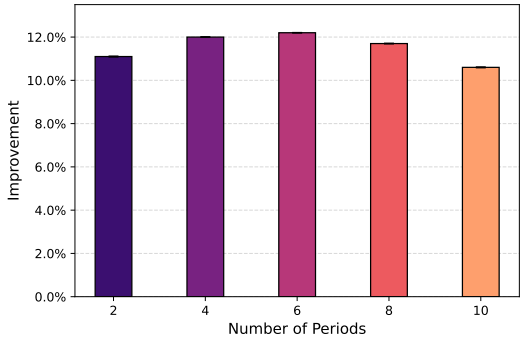


Figure 5: Experimental results with different numbers of periods.

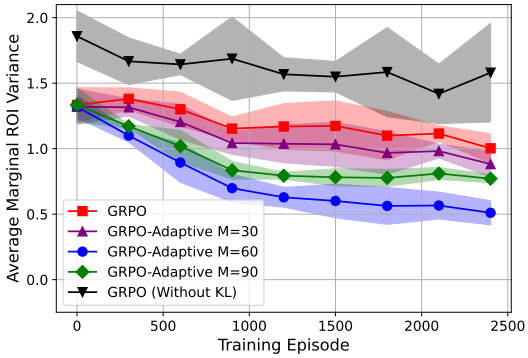


Figure 6: Experimental results with different frequencies of updating the reference model.

specialized. This observation highlights that its presence does not solely determine the effectiveness of RL in budget planning tasks, but also by how it is integrated—underscoring the importance of tailored architectural design for fully leveraging the benefits of policy learning.

5.4 Sensitivity Analysis

5.4.1 Time periods. To assess the robustness of our approach to the granularity of temporal partitioning, we conduct a sensitivity analysis by varying the number of time periods into which the total budget horizon is divided. Specifically, we evaluate the performance of our model under 5 different configurations: 2, 4, 6, 8, and 10 time periods. The results, presented in Figure 5, report the percentage improvement of our method over the strongest baseline (ABPlanner') in each setting.

We observe that our model consistently achieves performance improvements across all configurations, ranging from approximately 10.6% to 12.2%. The performance curve exhibits only mild fluctuations as the number of periods increases, indicating that our method is not sensitive to the specific temporal granularity chosen. In particular, the improvement is most pronounced when the number of periods is 6, aligning with typical real-world ad scheduling practices where campaigns are often divided into early, middle, and late phases. This also demonstrates the flexibility of our framework in capturing temporal dependencies at multiple resolutions.

5.4.2 Effect of KL Reference Refresh Strategy. To analyze the effect of KL regularization in GRPO-based fine-tuning, we compare standard GRPO with several GRPO-Adaptive variants under different KL reference update frequencies, including updates every $M = 30$, 60, and 90 iterations, as well as a variant that entirely removes the KL term. As shown in Figure 6, our full model (GRPO-Adaptive, $M = 60$) achieves the lowest Marginal ROI variance across training episodes, indicating that periodically refreshing the reference model substantially improves training stability and alignment. This design allows the target policy to continuously benefit from improved generations while maintaining a sufficiently stable regularization anchor, striking an effective balance between adaptability and constraint.

When the reference is updated too frequently (e.g., $M = 30$), the gain over vanilla GRPO becomes marginal, which we attribute to instability during early training: frequent updates tend to anchor the KL term to partially optimized or noisy generations, including incoherent reasoning and erratic numerical decisions. In contrast, updating the reference too slowly (e.g., $M = 90$) delays alignment with the improved policy and slows convergence, although it still outperforms static GRPO. Removing the KL term altogether leads to significantly higher variance and unstable training, demonstrating that KL regularization is essential for constraining the policy within a meaningful distributional bound rather than serving as a mere stabilizer. Overall, these results confirm that an intermediate update frequency provides the most effective trade-off, and that GRPO-Adaptive offers a simple yet robust improvement over fixed-reference regularization schemes.

6 Conclusion

In this work, we explore the challenging task of few-shot budget allocation in auction advertising, where data scarcity presents significant obstacles to effective decision making. Our solution features a dual-phase LLM architecture that separates few-shot reasoning from numerical optimization. To further enhance the decision quality of both agents, we introduce GRPO-Adaptive, an improved RL fine-tuning algorithm that dynamically updates the KL regularization anchor to stabilize and guide policy learning. Through comprehensive experiments across real-world and simulated environments, we demonstrate that our method significantly outperforms strong baselines. Overall, our work offers a promising direction for combining few-shot LLM reasoning with RL fine-tuning in budget allocation problems.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments. This work was supported by the National Key R&D Program of China under Grant 2023YFB2703800, and supported by Alibaba Group through Alibaba Innovative Research Program. The contact author is Zhen Xiao and Zhilin Zhang.

References

[1] Vashist Avadhanula, Riccardo Colini Baldeschi, Stefano Leonardi, Karthik Abinav Sankararaman, and Okke Schrijvers. 2021. Stochastic bandits for multi-platform budget optimization in online advertising. In *Proceedings of the Web Conference 2021*. 2805–2817.

[2] Dongbin Bai, Jiannong Cao, Yinfeng Cao, Long Wen, and Milos Stojmenovic. 2024. Ormer: A manipulation-resistant and gas-efficient blockchain pricing oracle for defi. *arXiv preprint arXiv:2410.07893* (2024).

[3] Christian Borgs, Jennifer Chayes, Nicole Immorlica, Kamal Jain, Omid Etesami, and Mohammad Mahdian. 2007. Dynamics of bid optimization in online advertisement auctions. In *Proceedings of the 16th international conference on World Wide Web*. 531–540.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Yinfeng Cao, Jiannong Cao, Dongbin Bai, Long Wen, Yang Liu, and Ruidong Li. 2025. Map the blockchain world: A trustless and scalable blockchain interoperability protocol for cross-chain applications. In *Proceedings of the ACM on Web Conference 2025*. 717–726.

[6] Shi Dong, Benjamin Van Roy, and Zhengyuan Zhou. 2022. Simple agent, complex environment: Efficient reinforcement learning with agent states. *Journal of Machine Learning Research* 23, 255 (2022), 1–54.

[7] Zhijian Duan, Yusen Huo, Tianyu Wang, Zhilin Zhang, Yeshu Li, Chuan Yu, Jian Xu, Bo Zheng, and Xiaotie Deng. 2025. An Adaptable Budget Planner for Enhancing Budget-Constrained Auto-Bidding in Online Advertising. *arXiv preprint arXiv:2502.05187* (2025).

[8] 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).

[9] 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*. 5038–5049.

[10] Muhammad Usman Hadi, Qasem Al-Tashi, Rizwan Qureshi, Abbas Shah, Amgad Muneer, Muhammad Irfan, Anas Zafar, Muhammad Bilal Shaikh, Naveed Akhtar, Mohammed Ali Al-Garadi, et al. [n. d.]. LLMs: A Comprehensive Survey of Applications, Challenges, Datasets, Models, Limitations, and Future Prospects. ([n. d.]).

[11] MohammadTaghi Hajiaghayi and Max Springer. 2022. Analysis of a Learning Based Algorithm for Budget Pacing. *arXiv preprint arXiv:2205.13330* (2022).

[12] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[13] Joni L Jones and Gary J Koehler. 2002. Combinatorial auctions using rule-based bids. *Decision Support Systems* 34, 1 (2002), 59–74.

[14] Timo Kaufmann, Paul Weng, Viktor Bengs, and Eyke Hüllermeier. 2024. A survey of reinforcement learning from human feedback. (2024).

[15] Vojtěch Klouda. 2025. Meta-prompts for LLM Prompt Optimization. (2025).

[16] Haoming Li, Yusen Huo, Shuai Dou, Zhenzhe Zheng, Zhilin Zhang, Chuan Yu, Jian Xu, and Fan Wu. 2024. Trajectory-wise iterative reinforcement learning framework for auto-bidding. In *Proceedings of the ACM Web Conference 2024*. 4193–4203.

[17] Linyu Li, Zhi Jin, Yuanpeng He, Dongming Jin, Haoran Duan, Zhengwei Tao, Xuan Zhang, and Jiandong Li. 2025. Rethinking regularization methods for knowledge graph completion. *arXiv preprint arXiv:2505.23442* (2025).

[18] Linyu Li, Zhi Jin, Xuan Zhang, Haoran Duan, Jishu Wang, Zhengwei Tao, Haiyan Zhao, and Xiaofeng Zhu. 2025. Multi-view riemannian manifolds fusion enhancement for knowledge graph completion. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[19] Pengcheng Li, Ammar Hawbani, et al. 2018. An efficient budget allocation algorithm for multi-channel advertising. In *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 886–891.

[20] Pengze Li, Mingxuan Song, Mingzhe Xing, Zhen Xiao, Qiuyu Ding, Shengjie Guan, and Jieyi Long. 2024. Spring: Improving the throughput of sharding blockchain via deep reinforcement learning based state placement. In *Proceedings of the ACM Web Conference 2024*. 2836–2846.

[21] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang Yang, Jian-Guang Lou, and Dongmei Zhang. 2023. Layoutprompter: awaken the design ability of large language models. *Advances in Neural Information Processing Systems* 36 (2023), 43852–43879.

[22] Mengjuan Liu, Wei Yue, Lizhou Qiu, and Jiaxing Li. 2020. An effective budget management framework for real-time bidding in online advertising. *IEEE Access* 8 (2020), 131107–131118.

[23] Manikanta Loya, Divya Anand Sinha, and Richard Futrell. 2023. Exploring the Sensitivity of LLMs' Decision-Making Capabilities: Insights from Prompt Variation and Hyperparameters. *arXiv preprint arXiv:2312.17476* (2023).

[24] Anik Mukherjee, Rangaraja P Sundarraj, and Kaushik Dutta. 2017. Apriori rule–based in-app ad selection online algorithm for improving supply-side platform revenues. *ACM Transactions on Management Information Systems (TMIS)* 8, 2-3 (2017), 1–28.

[25] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics* 50, 9 (2020), 3826–3839.

[26] Alessandro Nuara, Francesco Trovò, Nicola Gatti, and Marcello Restelli. 2022. Online joint bid/daily budget optimization of internet advertising campaigns. *Artificial Intelligence* 305 (2022), 103663.

[27] Sindhu Padakandla. 2021. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys (CSUR)* 54, 6 (2021), 1–25.

[28] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 3803–3810.

[29] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2023), 53728–53741.

[30] Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended abstracts of the 2021 CHI conference on human factors in computing systems*. 1–7.

[31] Nabeel Seedat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. 2023. Curated llm: Synergy of llms and data curation for tabular augmentation in ultra low-data regimes. (2023).

[32] Anastasiia Soboleva, Andrey Pudovikov, Roman Snetkov, Alina Babenko, Egor Samosvat, and Yuriy Dorn. 2025. Optimizing Online Advertising with Multi-Armed Bandits: Mitigating the Cold Start Problem under Auction Dynamics. *arXiv preprint arXiv:2502.01867* (2025).

[33] Mingxuan Song, Chengyu Hu, Wenyin Gong, and Xuesong Yan. 2022. Domain knowledge-based evolutionary reinforcement learning for sensor placement. *Sensors* 22, 10 (2022), 3799.

[34] Mingxuan Song, Pengze Li, Bohan Zhou, Shenglin Yin, Zhen Xiao, and Jieyi Long. 2025. AERO: Enhancing sharding blockchain via deep reinforcement learning for account migration. In *Proceedings of the ACM on Web Conference 2025*. 706–716.

[35] 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. *Advances in Neural Information Processing Systems* 37 (2024), 94428–94452.

[36] Hao Wang, Bo Tang, Chi Harold Liu, Shangqin Mao, Jiahong Zhou, Zipeng Dai, Yaqi Sun, Qianlong Xie, Xingxing Wang, and Dong Wang. 2023. HiBid: A cross-channel constrained bidding system with budget allocation by hierarchical offline deep reinforcement learning. *IEEE Trans. Comput.* 73, 3 (2023), 815–828.

[37] Chuhan Zhang, Antoine Miech, Jiajun Shen, Jean-Baptiste Alayrac, and Pauline Luc. 2023. Making the most of what you have: adapting pre-trained visual language models in the low-data regime. *arXiv preprint arXiv:2305.02297* (2023).

A Proof of Marginal ROI Optimality Condition

In Section 2, we state that under concave return functions, the optimal allocation under a fixed total budget satisfies that all marginal ROIs (i.e., first derivatives of return functions) are equal. We provide a formal proof of this result below.

Setting. Let B denote the total available budget to be allocated across T time periods. For each period $t \in \{1, \dots, T\}$, let b_t denote the allocated budget, and let $v_t(b_t)$ denote the return function in that period. The optimization objective is:

$$\max_{\{b_t\}} \sum_{t=1}^{T} v_t(b_t) \quad \text{subject to} \quad \sum_{t=1}^{T} b_t = B, \quad b_t \geq 0 \tag{18}$$

Assumptions. For each t, the return function $v_t(b)$ is:

- Differentiable,
- Strictly increasing: $v_t'(b) > 0$,
- Concave: $v_t''(b) < 0$.

Claim. At the optimum $\{b_t^*\}$, the marginal ROI must be equal across all periods:

$$v_1'(b_1^*) = v_2'(b_2^*) = \cdots = v_T'(b_T^*) \tag{19}$$

Proof. Assume, for contradiction, that there exists an optimal allocation $\{b_t^*\}$ and two indices $i \neq j$ such that:

$$v_i'(b_i^*) > v_j'(b_j^*) \tag{20}$$

We construct a new allocation $\{\tilde{b}_t\}$ as:

$$\tilde{b}_i = b_i^* + \delta, \quad \tilde{b}_j = b_j^* - \delta, \quad \tilde{b}_t = b_t^* \quad \text{for } t \notin \{i, j\} \tag{21}$$

where $\delta > 0$ is sufficiently small so that $\tilde{b}_i, \tilde{b}_j \geq 0$.

Note that $\sum_{t=1}^{T} \tilde{b}_t = \sum_{t=1}^{T} b_t^* = B$, so the new allocation remains feasible.

The change in total return is:

$$\Delta R = \left[v_i(b_i^* + \delta) - v_i(b_i^*)\right] + \left[v_j(b_j^* - \delta) - v_j(b_j^*)\right] \tag{22}$$

Using Taylor expansion (and concavity) around b_i^* and b_j^*:

$$v_i(b_i^* + \delta) - v_i(b_i^*) = v_i'(b_i^*)\delta - \frac{1}{2}|v_i''(\xi_i)|\delta^2 \tag{23}$$

$$v_j(b_j^* - \delta) - v_j(b_j^*) = -v_j'(b_j^*)\delta - \frac{1}{2}|v_j''(\xi_j)|\delta^2 \tag{24}$$

for some $\xi_i \in (b_i^*, b_i^* + \delta)$ and $\xi_j \in (b_j^* - \delta, b_j^*)$.

Thus,

$$\Delta R = \left[v_i'(b_i^*) - v_j'(b_j^*)\right]\delta - \frac{1}{2}\left(|v_i''(\xi_i)| + |v_j''(\xi_j)|\right)\delta^2 \tag{25}$$

Since $v_i'(b_i^*) > v_j'(b_j^*)$, the first-order term dominates the second-order term for small enough δ, so $\Delta R > 0$. Hence, $\{\tilde{b}_t\}$ yields strictly higher total return, contradicting optimality.

Therefore, the optimal solution must satisfy:

$$v_1'(b_1^*) = v_2'(b_2^*) = \cdots = v_T'(b_T^*) \tag{26}$$

B Prompt Examples

We present two representative prompt templates used for guiding the language model's decision-making process.

Prompt A: Few Shot Reasoner

```
You are given a total budget of {TOTAL} to allocate across {
     NUM} time periods. Based on your last attempt, identify
      the time period with the **lowest reward**, and
     reallocate some of its budget to the time period with
     the **highest reward**. Keep the allocations for other
     periods unchanged.

Last attempt: {few-shot data}

Please output the new allocation:
<reason>
...
</reason>
<answer>
[y1, y2, ..., y{NUM}]
</answer>

Your Response:
<reason>
```

Prompt B: Fine-grained Optimizer

```
You are given a budget allocation task across {NUM} time
     periods. The total budget is {TOTAL}, and it must be
     fully used - no more, no less. In every time period,
     allocating more budget results in smaller reward. Your
     goal is to equalize the reward across all periods.

You are provided with the last some rounds of allocations
     and observed rewards. Use this historical data to
     identify:
- Which periods likely has lower reward, reduce allocation
     slightly.
- Which periods likely has higher reward, increase
     allocation slightly.

Do not resort to a naive and robust uniform allocation.
     Instead, carefully analyze the underlying patterns and
     allocate the budget based on the relationship between
     historical data and observed rewards.

Last attempt: {sliding-window data}

Please output the new allocation:
<reason>
...
</reason>
<answer>
[y1, y2, ..., y{NUM}]
</answer>

Your Response:
<reason>
```

C Training Procedure of GRPO-Adaptive

Algorithm 2: GRPO-Adaptive in DARA

Input: Pretrained policy π_{θ_0}, prompt distribution $\mathcal{P}(Q)$
Output: Updated policy π_θ

Initialize policy parameters $\theta \leftarrow \theta_0$
Initialize KL reference baseline $\pi_{\text{ref}} \leftarrow \pi_{\theta_0}$
Set hyperparameters: group size G, clip range ϵ, KL weight β, baseline update period M
for *iteration* = 1 *to* N **do**
  `// Step 1: Sample batch and generate outputs`
  Sample prompts $\{q_1, \ldots, q_B\} \sim \mathcal{P}(Q)$
  For each q, generate outputs $\{o_i^q\}_{i=1}^G \sim \pi_\theta(\cdot \mid q)$
  `// Step 2: Compute reward for each output`
  $r_i \leftarrow R(q, o_i^q)$
  `// Step 3: Compute normalized advantage`
  $\hat{A}_{i,t} \leftarrow \frac{r_{i,t}-\mu}{\sigma}$ where $\mu = \text{mean}(\{r_{i,t}\})$, $\sigma = \text{std}(\{r_{i,t}\})$
  `// Step 4: Compute loss`
  $\mathcal{L}_{\text{GRPO-A}}(\theta) = -L_{Adv}(\theta) + \beta \cdot D_{\text{KL}}(\pi_\theta \| \pi_{\text{ref}})$
  `// Step 5: Update policy parameters`
  $\theta \leftarrow \theta - \alpha \cdot \nabla_\theta \left(\frac{1}{B} \sum_q \mathcal{L}_{\text{GRPO-A}}(q) \right)$
  `// Step 6: Refresh KL baseline`
  if *iteration mod* M == *0* **then**
    $\pi_{\text{ref}} \leftarrow$ `deepcopy`(π_θ)

D Hyperparameter and Baseline Configuration

The hyperparameter settings used in DARA are summarized in Table 2. To comprehensively assess the performance of our proposed method, we have selected five algorithms for comparison:

- **ABPlanner'** [7]: ABPlanner' is a simplified version of ABPlanner, designed to control for variables by focusing solely on the reinforcement learning component and excluding the controller design. ABPlanner' effectively demonstrates the strategy optimization capability of reinforcement learning algorithms in budget allocation tasks.
- **HiBid'** [36]: HiBid' is a simplified version of HiBid (Wang et al., 2023), which uses a hierarchical reinforcement learning algorithm to learn both high-level budget planning and low-level bidding strategies. Unlike the original HiBid, HiBid' focuses only on the high-level budget planning component, making it suitable for improving fixed underlying auto-bidder systems.
- **Q-MCKP** [19]: Q-MCKP is an algorithm that discretizes the budget space into multiple bins and independently learns the Q-value function for each stage. It solves the optimal budget allocation based on the learned Q-values.
- **DPO Fine-tuning** [29]: Direct Preference Optimization (DPO) fine-tuning is a preference-alignment-based fine-tuning method that enhances the model's generalization ability in few-shot tasks by extracting key tokens from preference and rejection pairs during training.
- **GPT-4o** [12]: GPT-4o is one of the most advanced LLMs developed by OpenAI. It demonstrates strong in-context learning capabilities across a wide range of tasks. In this work, GPT-4o performs reasoning and decision-making solely based on prompt engineering, without any parameter updates.

Table 2: Hyperparameters and their values

Hyperparameter	Value
Number of training iterations, N	500
Number of baseline update period, M	60
Number of time periods, T	6
Batch size, *batch_size*	8
Number of generations, *num_generations*	3
Max completion length, *max_completion_length*	500
KL regularization coefficient, β	0.04
Learning rate, lr	5×10^{-6}
Scaling factor for preference loss, μ	1
Epsilon for PPO clipping, ϵ	0.1
Minimum Adjustment threshold, δ	0.2
Clipping bound, τ	0.2
Scaling coefficient, τ	0.5
Total budget, B	6
Number of tries per environment, *N_TRY*	10

E Response Sample

Below is a representative example of the model's generated reasoning and corresponding action in a decision-making round:

```
<reason>
Looking at the historical data, I notice that periods 3 and
     5 have consistently low rewards (e.g., 0.0000 in
     multiple rounds), indicating that allocating more
     budget to these periods would result in smaller rewards
     . Therefore, I should reduce the allocation for these
     periods.
Conversely, periods 4 and 6 have higher rewards (e.g.,
     0.2469, 0.2693, 0.4895 in previous rounds), suggesting
     that increasing the budget for these periods may yield
     higher rewards. Thus, I should increase the allocation
     for these periods.
Additionally, period 2 has shown some reward (0.0733,
     0.0726), so I should slightly increase its allocation
     to take advantage of the modest reward.
However, I must ensure the total allocation sums to 6. After
      adjusting allocations based on the above observations,
      I need to balance the budget carefully to maintain the
      total allocation of 6.
</reason>
<answer>
[0.93, 1.18, 0.77, 1.19, 0.75, 1.18]
</answer>
```