# FLOOD: Fine-tuning LLM with Offline Optimal Oracle for Online Decision Making

Tao Qiu[1†], Kaiming Xiao[2†], Hang Zhang[1], Zhihao Chen[1], Haoyu Yang[1], Xuan Li[1], Mao Wang[1*]
[1]Laboratory for Big Data and Decision, National University of Defense Technology, Changsha, China
[2]Wuxi Research Institute of Applied Technologies, Tsinghua University, Wuxi, China
Email: {qiutao20, kmxiao, zhanghang13, chenzhihao22, yanghaoyu20, xuanli, wangmao}@nudt.edu.cn

***Abstract*—Online decision refers to making sequential choices without future information, which is widely applied in real world scenarios such as dynamic resource allocation, scheduling and online advertising. Motivated by the remarkable reasoning and planning capabilities of Large Language Model (LLM), researchers have applied these models to such online problems. However, existing LLM-based approaches primarily rely on parameter-frozen LLM, which often exhibit weak performance in complex, non-stationary and long-horizon environment. Although fine-tuning LLM for online decision making is a promising direction, this remains challenging due to the lack of supervision labels and effective training mechanisms for online scenarios. To address this, we propose the FLOOD framework (Fine-tuning LLM for Online decision via Oracle Distillation), which distills the capabilities of an offline optimal oracle into an online decision LLM. Specifically, our framework leverages an offline optimal oracle (with full information access) to generate optimal decision trajectories. These trajectories are then distilled into an online LLM via fine-tuning. Crucially, to adapt to the characteristics of online decision making, we design Decision Period Aligned Gradient Accumulation (DPAGA) strategy where gradients are accumulated to match the granularity of decision periods. We train FLOOD solely on synthetically generated oracle trajectories, and evaluate its performance on both synthetic data and a real world sequential emergency dataset. These experiments demonstrate that FLOOD generalizes effectively, achieving competitive performance compared to existing baselines. Further experiment confirms that the DPAGA strategy is essential for the distillation process.**

***Index Terms*—Online Decision Making, Large Language Model, Fine-tuning via LoRA**

## I. Introduction

Online decision making is an important problem in real-world applications like dynamic resource allocation, scheduling and online advertising, focusing on sequential choices under uncertainty [1], [2]. In each round, the decision-maker selects an irrevocable decision based on historical and current information, and the reward revealed after the decision is committed. The ultimate goal is to minimize regret, the performance gap between the online policy and an optimal offline oracle that has full access to future information.

Despite its broad applicability, online decision faces two key challenges. First, decisions are made without access to future information, making the design of effective online algorithms inherently difficult. Second, traditional approaches rely on hand-crafted heuristics, which often lead to poor performance in complex settings where environments are highly dynamic or non-stationary [3]. These challenges motivate more effective approaches to online decision making.

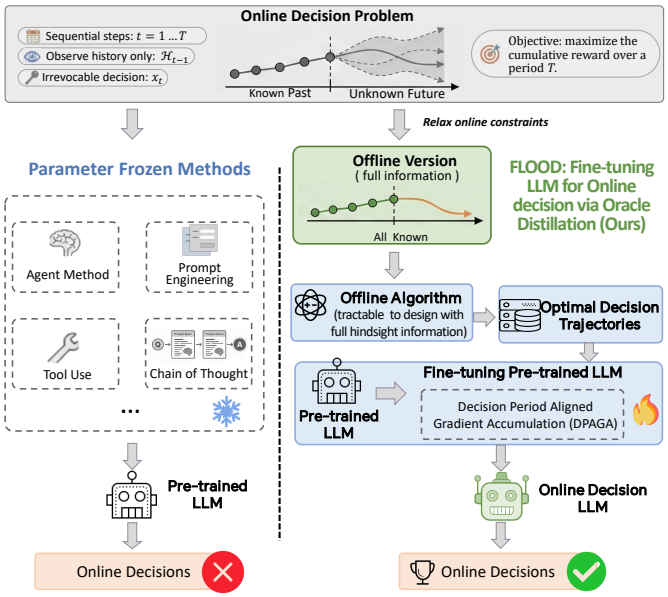


Fig. 1. Compared with parameter-frozen methods (left) and overview of FLOOD (right). FLOOD relaxes the online problem to an offline version to compute optimal decision trajectories, then distills them into LLM by fine-tuning with DPAGA strategy.

Large Language Model (LLM) have recently revealed remarkable capabilities in reasoning and planning [4]–[6], motivating their application to online decision making. Existing studies address online problems with parameter-frozen LLM [7]–[10], relying mainly on general capabilities and prompting. However, without parameter updates, these approaches often struggle in complex, non-stationary, and long-horizon environments [7], [8]. Moreover, training LLM to enhance online decision ability of LLM remains challenging due to the lack of effective mechanisms and supervision label.

To address this challenge, we propose Fine-tuning LLM for Online decision via Oracle Distillation (FLOOD), a method

†These authors contributed equally to this work.
*Corresponding author.

that distills an offline optimal oracle into an online decision LLM (as shown in Fig.1). Specifically, we leverage an offline optimal oracle by relaxing online constraints to allow full information access to generate optimal decision trajectories, which serve as supervision to fine-tune the LLM. Crucially, to adapt to the characteristics of online decision, we design a Decision Period Aligned Gradient Accumulation (DPAGA) strategy, where gradients are accumulated to match the granularity of decision period. This transforms intractable online algorithm design into a tractable offline decision problem and general oracle distillation pipeline. This addresses the lack of supervision labels and provides an effective training mechanism. Experiments on both synthetic environments and a real-world sequential emergency dataset demonstrate that FLOOD achieves superior performance. Further experiment confirms that the proposed DPAGA strategy is essential for the distillation process.

Our main contributions are as follows:

- We propose a method to fine-tune LLM for online decision making, transforming the task of designing intractable online algorithms into the task of solving a relaxed offline version of the problem.
- We design a decision period aligned gradient accumulation strategy that accumulates gradients at the granularity of decision periods. This makes the parameter updates during fine-tuning aligned with the objectives of online decision making.
- We demonstrate that FLOOD trained by synthetic data can be deployed online and perform well under a real-world dataset, validating its effectiveness.

## II. RELATED WORK

### A. Online Decision Making

Online decision making studies sequential decision under uncertainty, where decisions are irrevocable and rewards are revealed after commitment. The main goal of these problems is minimize *regret*, defined as the performance gap against an optimal offline oracle [11]. Alternatively, the *competitive ratio* (CR) is also used in the online algorithms literature [12]. Andrew et al. [13] established a fundamental theoretical limit, proving that sublinear regret and constant CR cannot be achieved simultaneously. These problems are widely applicable, including dynamic resource allocation [14], online scheduling [15] and recommendation systems [16].

Regarding algorithmic development, extensive research has been conducted to address these problems under various settings, ranging from stochastic input models to adversarial worst-case scenarios [12], [17], [18]. However, many traditional methods rely on rigid mathematical formulations and structured numerical inputs, and often require substantial redesign when constraints or problem variants change. This limitation motivates the exploration of LLM-based agents capable of understanding complex environments and generalizing across tasks [6]–[8], [19].

### B. LLM for Online Decision Making

Existing research on applying LLM to online decision making typically follows two paradigms. (i) The first relies on *in-context learning* with frozen models. For instance, Yao et al. proposed the ReAct framework, integrating reasoning and action generation to interact with environments via prompting [20], and Park et al. demonstrated that LLM agents can exhibit no-regret behaviors on simple tasks through chain-of-thought prompting [7]. However, this paradigm faces limitations. As demonstrated by Krishnamurthy et al., parameter-frozen LLM struggles in complex bandit instances, indicating that prompt engineering alone is insufficient for complex online settings [8]. (ii) The second paradigm employs hybrid architectures, where the LLM serves as an assistant within a traditional algorithmic framework. In this paradigm, LLM is relegated to a support role: acting as a reward predictor [19], a preference interpreter [9], or a generator of prior knowledge to start contextual bandits [21]. While these methods leverage LLM capabilities, the final decision is still governed by a traditional online algorithm. As a result, their performance and applicability remain constrained by traditional online algorithms.

These limitations motivate a direct approach: enabling the LLM itself to function as an online decision maker through fine-tuning. However, the primary obstacle to this path is the lack of supervision mechanisms in partial-feedback environments, few works have successfully trained LLM for this purpose. FLOOD addresses this gap by using an offline optimization oracle to generate supervision and distilling it into an online LLM.

### C. Imitation Learning and Distillation

Our approach Oracle Distillation is inspired by Imitation Learning (IL), specifically the paradigm of learning from an oracle. Standard IL typically treats imitation as supervised learning on state-action pairs [22], [23]. This setting involves an oracle who possesses access to extra information that is unavailable to the "student" during inference. This paradigm has been widely applied in reinforcement learning, such as Hindsight Experience Replay (HER) [24], to accelerate policy learning by leveraging teachers during training.

In the context of LLM, knowledge distillation is widely employed, typically to transfer reasoning capabilities from larger models (e.g., GPT-4) to smaller ones for model compression or to enforce chain-of-thought consistency [25]–[27]. We utilize this teacher–student paradigm along a different dimension: distilling the online decision logic in oracle-generated trajectories.

## III. METHOD

In this section, we present FLOOD, an oracle distillation framework designed to transform LLM into online decision makers. The core philosophy of FLOOD is to convert offline optimal solutions into sequential training signals, allowing the LLM to learn implicit decision policies that can be executed in online setting. As illustrated in Fig. 2, the framework comprises two primary stages: (i) Offline Oracle Construction,

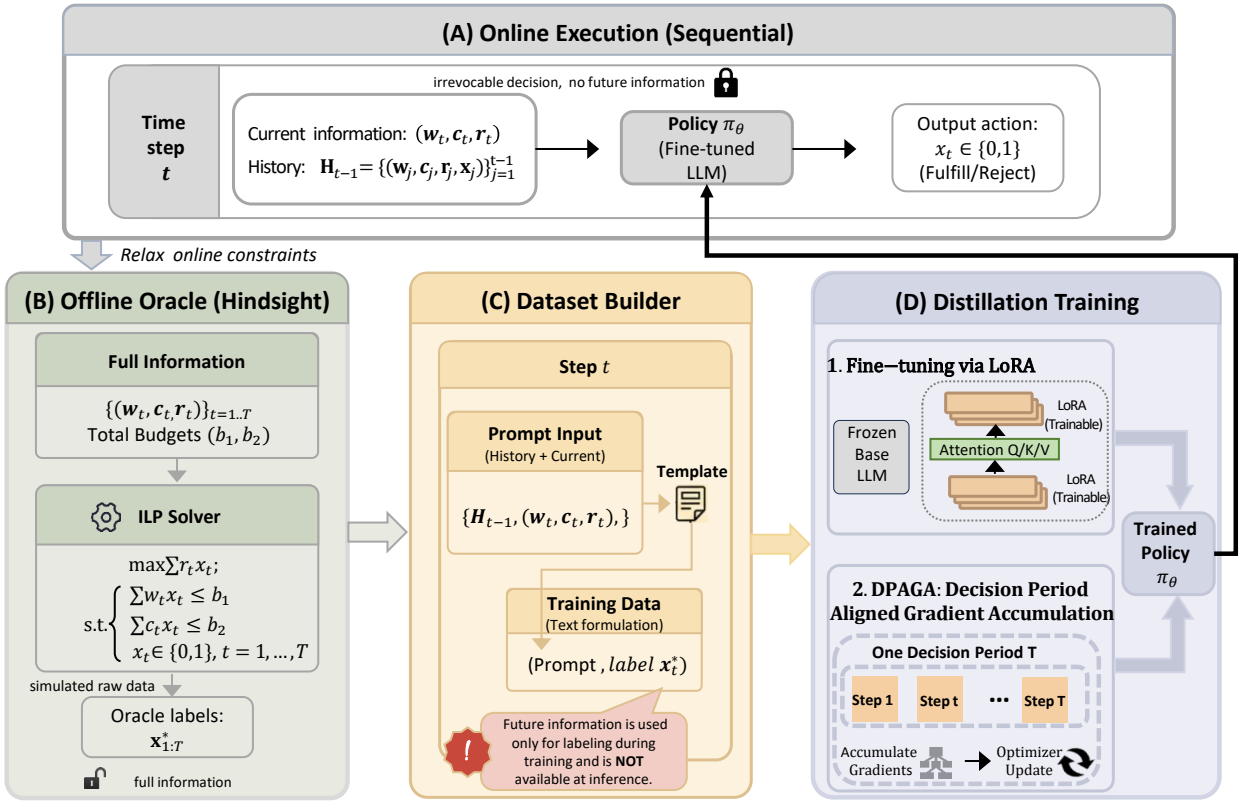


Fig. 2. FLOOD framework for online decision via oracle distillation: (A) a deployed LLM makes sequential decisions; (B) an offline oracle with full information to produce optimal labels; (C) a dataset builder constructs prompts paired with oracle decisions; (D) the LLM is fine-tuned with LoRA and decision-period-aligned gradient accumulation.

and (ii) Online Policy Distillation through decision period aligned fine-tuning.

### A. Problem Formulation

We instantiate FLOOD on an online resource allocation problem with $T$ steps. At each step $t \in \{1, \ldots, T\}$, a request tuple $(w_t, c_t, r_t)$ arrives, denoting the resource demand, transportation cost and potential reward. Besides current information, decision maker has access to the historical trajectory $H_{t-1}$, defined as:

$$H_{t-1} = (r_j, w_j, c_j, x_j)_{j=1}^{t-1}. \tag{1}$$

Based on $H_{t-1}$ and the current input $(w_t, c_t, r_t)$, the decision maker make an irrevocable decision $x_t$ without knowledge of future arrivals:

$$x_t = \pi(w_t, c_t, r_t, H_{t-1}), \tag{2}$$

where $x_t = 1$ denotes fulfill request and $x_t = 0$ denotes rejection. If the request is fulfilled, the decision maker suffers consume $w_t$ of the limited rescue resource and transportation cost $c_t$, while yielding reward $r_t$. The goal is to maximize the total cumulative reward with a budget constraints for resources $b_1$ and transportation costs $b_2$. The challenge is to dynamically satisfy these global constraints under uncertainty, balancing the immediate reward of the current request against the potential value of saving budget for unknown future demands. In essence, solving the online problem is to find a policy $\pi$.

### B. Offline Oracle Construction

The first step of FLOOD is constructing the oracle and it relies on the fact that online problems can be relaxed into tractable offline versions with full information. As shown in Fig. 2(B), we produce the optimal trajectory $x^*_{1:T}$ with hindsight information.

Given the sequence of all requests and total budgets, the Offline Oracle is formulated as a binary Integer Linear Program (ILP):

$$\max \sum_{t=1}^{T} r_t x_t$$
$$\text{s.t.} \begin{cases} \sum_{t=1}^{T} w_t x_t \le b_1 \\ \sum_{t=1}^{T} c_t x_t \le b_2 \\ x_t \in \{0,1\}, \quad t = 1, \ldots, T \end{cases} \tag{3}$$

Following [28], it can be proven that the oracle decision vector $\mathbf{x}^* = [x_1^*, \ldots, x_T^*]$ is the optimal trajectory. In the FLOOD pipeline, the oracle serves two essential roles: (i) generating training label for fine-tuning, and (ii) providing

a baseline for evaluating online performance via competitive ratio and regret.

### C. Training via Oracle Distillation

The goal of oracle distillation is to learn $\pi$ in (2) so that the LLM can execute decisions under online setting.

**Utilizing oracle to get fine-tuning data.** To distill the oracle via fine-tuning, we construct the training data in a prompt-based format. Moreover, to distill the oracle into an online policy, we generate the training instances following the online decision setting. While the oracle produce optimal decisions $\mathbf{x}^*$ using the full information, the online policy infers $x_t$ solely from the observable history, as illustrated in Fig. 3.

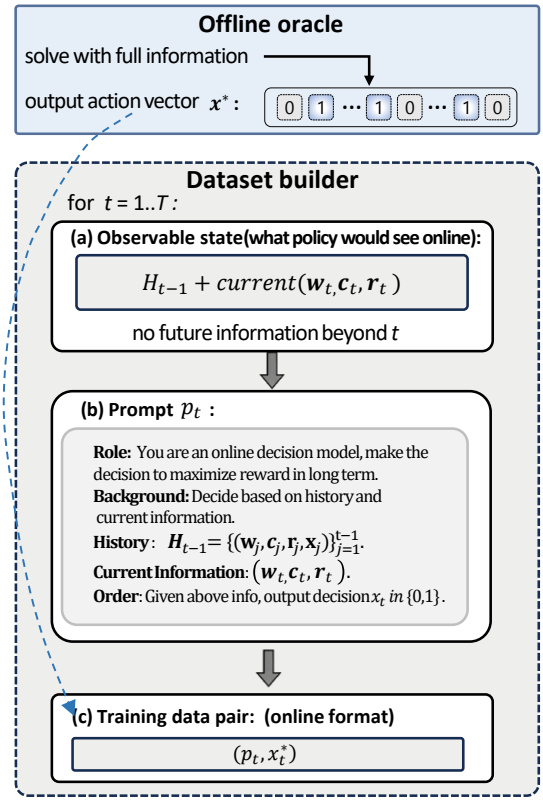


Fig. 3. **Utilizing oracle to build fine-tuning data** The offline ILP oracle computes the optimal decision sequence $\mathbf{x}^*$. The dataset builder then iterates over $t = 1..T$ and constructs each prompt under online setting (history $H_{t-1}$ and current request $(w_t, c_t, r_t)$) while using the oracle decision $x_t^*$ as the supervised label.

For each time step $t$, we construct a training pair $(p_t, x_t^*)$ as follows:

- Prompt Construction $(p_t)$: We serialize the observable state into a textual instruction. This state includes the historical context $H_{t-1}$, the current information $(w_t, c_t, r_t)$. Crucially, all future information are masked.
- Oracle Annotation $(x_t^*)$: The supervised target is directly derived from the offline oracle's decision at step $t$.

Through this process, the offline optimal trajectory is converted into a online format dataset $\mathcal{D} = \{(p_t, x_t^*)\}_{t=1}^T$. By distilling on this dataset, the complex neural network of the LLM learns to characterize the implicit decision patterns embedded in the data.

**Decision Period Aligned Gradient Accumulation.** Standard fine-tuning typically treats training samples as independent and identically distributed (i.i.d.), minimizing loss on a per-sample or random-batch basis. However, online decision making is inherently sequential, aiming to maximize cumulative rewards over a period $T$. To bridge this gap, we align the gradient optimization step with the problem's structure. Specifically, we adopt a decision period aligned gradient accumulation strategy where gradients are accumulated over the entire decision period before updating parameter. Mathematically, while this resembles using a batch size of $T$, the key distinction lies in the data sampling strategy. Standard training shuffles samples randomly, breaking temporal dependencies. DPAGA treats the entire trajectory as a single atomic sample, ensuring that parameter updates reflect the global objective of the entire period. The procedure is formally detailed in Algorithm 1.

**Algorithm 1** Decision-Period-Aligned Gradient Accumulation

**Input:** Training dataset $\mathcal{D}$; Period length $T$; Initial parameters $\theta$; Learning rate $\eta$
**Output:** Optimized parameters $\theta^*$

Initialize $g_{\text{acc}} \leftarrow 0$ ▷ Initialize gradient accumulator
Set step counter $k \leftarrow 0$
**for** each trajectory $\tau = \{(p_t, x_t^*)\}_{t=1}^T$ **in** $\mathcal{D}$ **do**
  **for** $t = 1$ **to** $T$ **do**
    $k \leftarrow k + 1$
    **Forward:** Compute $\hat{x}_t \leftarrow \pi_\theta(p_t)$
    **Loss:** $\ell_t \leftarrow \mathcal{L}(\hat{x}_t, x_t^*)$
    **Backward:** $g_t \leftarrow \nabla_\theta \ell_t$
    **Accumulate:** $g_{\text{acc}} \leftarrow g_{\text{acc}} + \frac{1}{T} g_t$
    **if** $k \mod T = 0$ **then** ▷ End of a decision period
      **Update:** $\theta \leftarrow \theta - \eta \cdot g_{\text{acc}}$
      **Reset:** $g_{\text{acc}} \leftarrow 0$
    **end if**
  **end for**
**end for**
**return** $\theta$

**Parameter-efficient fine-tuning with LoRA.** We apply LoRA [29] to adapt the pretrained LLM efficiently. LoRA rewrites the pretrained weight matrix $W_0 \in \mathbb{R}^{d\times k}$ as $W_0 + \Delta W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}, A \in \mathbb{R}^{r\times k}$, and $r \ll \min(d, k)$. This constrains updates to a low-dimensional subspace and reduces training cost. In implementation, LoRA inject these low-rank adapters into the transformer attention weights (e.g., $Q, K$, and $V$) while freezing the remaining parameters.

Overall, FLOOD follows a two-stage pipeline: (1) construct an offline oracle to obtain optimal decision trajectories from an all-knowing perspective, and (2) distill these trajectories into an online LLM policy using DPAGA and LoRA-based fine-tuning. This design turns online algorithm design into a tractable framework.

## IV. EXPERIMENTS

In this section, we evaluate the proposed FLOOD framework in synthetic and real world datasets. We first detail the experimental setup and baseline comparisons, followed by a comprehensive analysis of the results. The code is available at https://github.com/qiutao20/FLOOD.

### A. Experiment Setup

**Implementation Details.** All experiments were conducted on a server with a NVIDIA A100 GPU and 220GB RAM. We employed ChatGLM3-6B [30] as the backbone LLM. The model was fine-tuned via LoRA for 20 epochs with a learning rate of $6 \times 10^{-6}$.

**Training data construction.** We synthesize raw training data of 200 decision periods, each consisting of $T = 60$ time steps, resulting in 12,000 step-level samples. For each period, we build request tuples $(w_t, c_t, r_t)$ as follows: the resource requests $w_t$ is simulated to model different demand situations, while the transportation cost and reward are generated randomly and kept fixed. Specifically, we simulated resource requests using a normal distribution with a mean of $\mu = 10$ and a standard deviation of $\sigma = 3$. And budgets are calculated based on scarcity ratios $w_1 = 0.5$ and $w_2 = 0.4$ (as analyzed in IV-G). For each decision period, the offline oracle (Section III-B) was executed to generate the optimal decision $x_t^*$ as the optimal label. Finally, we hold out 5% of the synthetic samples as the synthetic test set.

**Real-World Benchmark.** To evaluate the model's robustness to distribution shifts and real-world complexity, we utilized the 2020 Ebola Outbreak Dataset [31]. Derived from actual logistics logs in the Congo, this dataset features non-stationary demand patterns and authentic constrains. Crucially, to evaluate the generalization capability of FLOOD, we do not use this real world dataset for training.

### B. Baselines

To validate the effectiveness of our oracle distillation framework, we compare FLOOD against three baselines:

- **MonERP [28]:** A specialized online algorithm designed for the online resource allocation problem. It is the state-of-the-art (SOTA) traditional algorithm on this problem.
- **Base LLM [30] :** The base model operating via prompting without fine-tuning. This baseline assesses the inherent reasoning capability of the base model and highlights the necessity of oracle distillation. The Base LLM uses the same prompt and input fields as FLOOD.
- **RanDM [32]:** A naive policy that makes random decisions when lack information, mimicking human intuition.

It is important to clarify the distinction between the online resouce allocation formulation and Reinforcement Learning (RL). In this setting, the arrival of future requests $(w_i, c_i, r_i)_t$ is independent of the decision maker's historical actions. This lacks the state transition dynamics ($S_{t+1} \sim P(S_t, A_t)$) characteristic of Markov Decision Processes (MDPs). Therefore, we compare against OLP-based algorithm rather than RL algorithms.

### C. Evaluation Metrics

To provide a holistic view of performance, we employ four metrics covering accuracy, optimality, and feasibility.

**Accuracy (Acc).** The offline oracle decisions are optimal labels. Accuracy measures the proportion of time steps where the online policy's decision $x_t$ matches the oracle's decision $x_t^*$:

$$\text{Accuracy} = \frac{1}{T}\sum_{t=1}^{T} \mathbb{I}(x_t = x_t^*) \tag{4}$$

*Note:* Accuracy measures local alignment with the oracle's specific trajectory. However, as the ultimate objective is maximizing cumulative reward, alternative paths may exist that yield comparable performance.

**Competitive Ratio (CR).** CR is a standard metric in online algorithms, evaluating the ratio of the cumulative reward obtained by the online algorithm ($\text{R}_{\text{ALG}}$) to that of the offline global optimal oracle ($\text{R}_{\text{OPT}}$):

$$\text{CR} = \frac{\text{R}_{\text{ALG}}}{\text{R}_{\text{OPT}}}, \tag{5}$$

where CR $\in [0, 1]$, with higher values indicating performance closer to the optimal upper bound.

**Regret.** Regret quantifies the difference in cumulative reward up to time $T$:

$$Regret(T) = \sum_{t=1}^{T} r_t^* - \sum_{t=1}^{T} r_t, \tag{6}$$

where $r^*$ is the reward of the offline optimal solution. Since the oracle is globally optimal over the entire horizon $T$, intermediate regret at step $t < T$ can be positive or negative.

**Over Ratio (OR).** To assess the feasibility of LLM's decisions, we define the Over Ratio(OR), which measures the extent to which the planned resource usage exceeds the global budget constraint $B$:

$$\text{OR} = \max\left(0, \frac{R_{\text{used}} - B}{B}\right), \tag{7}$$

where $R_{\text{used}}$ is the total resource consumed by the policy. Crucially, when testing other metrics, any decision violating the budget is strictly executed as invalid, ensuring performance scores are not inflated by infeasible decisions.

### D. Main Results

Trained on synthetic data, Table I summarizes the performance of FLOOD against the baselines on both the synthetic and real world test sets.

**Superiority over LLM.** The comparison between FLOOD and the base LLM highlights a critical insight: foundation models alone are insufficient for complex online decision. The base baseline struggles to outperform the random policy *RanDM* in terms of CR and OR. This failure stems from the inability of base model to comprehend the trade-offs between immediate gratification and long-term budget conservation without specific training. In contrast, FLOOD achieves a substantial improvement (+12.8% Accuracy and +10.7% CR

TABLE I
RESULTS ON BOTH SYNTHETIC AND REAL WORLD DATA

| Dataset | Method | Acc ↑ | CR ↑ | OR ↓ |
|---|---|---|---|---|
| **Synthetic** | RanDM | 0.4970 (±0.0697) | 0.7746 (±0.0856) | 0.4009 (0.1037) |
| | baseLLM | 0.4965 (±0.0542) | 0.7614 (±0.1419) | 0.1851 (0.2112) |
| | MonERP | 0.5271 ( – ) | 0.8013 ( – ) | 0 ( – ) |
| | FLOOD | **0.6027 (±0.0471)** | **0.8429 (±0.0417)** | **0.1293 (0.2743)** |
| **Real World** | RanDM | 0.5048 (±0.0634) | 0.7253 (±0.1201) | 0.4172 (0.1542) |
| | baseLLM | 0.4817 (±0.0624) | 0.6776 (±0.0519) | 0.2417 (0.2208) |
| | MonERP | 0.5667 ( – ) | **0.8029 ( – )** | 0 ( – ) |
| | FLOOD | **0.6075 (±0.0522)** | 0.7413 (±0.1051) | **0.1819 (0.2917)** |

* *MonERP* produces identical results, hence standard deviation is not reported.

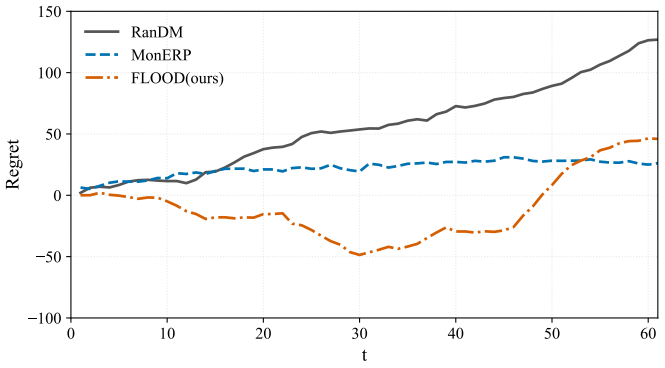


Fig. 4. Cumulative Regret over time. FLOOD demonstrates low regret compared to the static linear growth of RanDM.

on simulation data), demonstrating that our oracle distillation framework successfully transfers decision logic to the LLM.

**Comparison with Specialized algorithm.** When compared to *MonERP*, a SOTA algorithm specifically hand-crafted for this problem, FLOOD demonstrates competitive performance. Notably, FLOOD achieves higher Accuracy (Oracle Alignment) on both datasets, indicating a stronger ability to learn the oracle's decision patterns. In terms of CR, FLOOD outperforms MonERP on the synthetic test set (0.8429 vs. 0.8013). On the real-world dataset, FLOOD exhibits a slight performance gap compared to MonERP. We attribute this to the inherent trade-off between generality and specialization: MonERP relies on conservative, hard-coded rules that guarantee safety, whereas FLOOD learns decision intelligence from oracle. While slightly less robust to extreme distribution shifts, FLOOD achieves this performance without manual engineering, highlighting its potential as a more general framework.

These results are particularly significant given that FLOOD has never seen the real-world data nor fine-tuned on specific distribution. It successfully extrapolates the learned policy to a completely real-world environment.

### *E. Analysis of FLOOD Behavior*

To deeply understand *how* FLOOD makes decisions, we analyze how the FLOOD decides through regret curve and metric divergence.

**Regret Analysis.** To analyze the stability of the learned policy, we visualize the cumulative regret trajectories in Fig. 4. The *RanDM* baseline exhibits a steep linear growth, indicating a consistent failure to secure high-value requests. In contrast, *FLOOD* successfully suppresses the regret growth, maintaining a flattened trajectory. Flood achieves a final cumulative regret of approximately 40, which is comparable to the specialized algorithm MonERP. Notice that the regret temporarily drop below zero at intermediate steps. This is because the Oracle could preserves budget for future high-reward requests, while online policy prefer capturing immediate rewards. However, at the final step $T$, the Oracle always achieves superior or equal performance, consistent with its global optimality. The curves indicates distinct behavioral signatures: MonERP follows a smooth, conservative path derived from fixed rules, whereas FLOOD displays slight fluctuations. These fluctuations indicate that the LLM is actively trading off between immediate greedy decisions and long-term planning, proving that it has learned policy of the optimization problem rather than merely memorizing static labels.

**The Accuracy-Optimality Gap.** A key observation is that FLOOD achieves higher accuracy than MonERP but occasionally lower CR. This corroborates our statement in Section IV-C: high local accuracy does not strictly guarantee global optimality. MonERP utilizes conservative thresholds to avoid "fatal" errors (e.g., early resource depletion), ensuring a decent baseline CR. In contrast, FLOOD, distilled from the oracle, may learn aggressive strategies. A single wrong decision in a critical high-value resource scenario can disproportionately impact the cumulative reward, even when the model maintains 99% accuracy in other cases. This suggests that future work could benefit from incorporating reward-weighted loss functions to prioritize critical decision steps.

**Feasibility and Resource Management.** The *OR* reflects the model's ability to respect global constraints. As expected, MonERP achieves a 0% violation rate due to its hard constraints. Unlike such deterministic heuristics, probabilistic errors are inherent to neural networks. However, FLOOD significantly mitigates this issue, drastically reducing OR compared to baselines. This indicates that FLOOD has implicitly learned to model resource scarcity solely from the oracle's labels, effectively internalizing the concept of a budge without explicit constraint programming.

TABLE II
ABLATION ANALYSIS: EFFECTIVENESS OF THE DPAGA STRATEGY

| Model | Decision Period Aligned Update | Acc | CR |
|---|---|---|---|
| RanDM | - | 0.4970 | 0.7746 |
| Base LLM | - | 0.4965 | 0.7614 |
| FLOOD w/o DPAGA | × | 0.4666 | 0.7219 |
| **FLOOD (Ours)** | ✓ | **0.6027** | **0.8429** |

### F. Ablation Studies

**Effectiveness of DPAGA.** A pivotal contribution of FLOOD is the DPAGA strategy. Table II presents an ablation analysis comparing our method against a standard fine-tuned variant and baselines.

The results demonstrate the critical role of this strategy. As shown in the table, removing the DPAGA strategy (FLOOD w/o DPAGA) results in a sharp performance degradation, with the Competitive Ratio (CR) dropping from **0.8429** to **0.7219**. Notably, this lower performance is comparable to the non-fine-tuned *base LLM*, suggesting that standard fine-tuning fails to capture the long-term objectives inherent in online decision making. Consequently, these results confirm that the DPAGA strategy is an essential mechanism for effective online decision fine-tuning.

### G. Parameter analysis

In online resource allocation problem, the decision difficulty is linked to the scarcity of total resources $b_1$ and transportation budgets $b_2$. Tighter constraints lead lower tolerance for errors, demanding more precise decision-making. To quantify this, we define the *resource scarcity coefficient* $w_1$ and the *transportation budget scarcity coefficient* $w_2$ as:

$$w_1 = \frac{b_1}{\sum \text{Demand}}, \quad w_2 = \frac{b_2}{\sum \text{Total\_distance}} \tag{8}$$

where $w_1, w_2 \in (0, 1]$. Lower values indicate higher scarcity.

**Sensitivity Analysis of Scarcity Parameters.** We first analyze how these constraints impact the upper bound of performance (offline optimal) to identify critical decision regions. We varied $w_1$ and $w_2$ at intervals of 0.1 to generate 100 parameter combinations. Figure 5 illustrates the impact of these coefficients on total satisfied demand.

The heatmap indicates three circumstances:

- Resource Bottleneck Segment ($w_1 < 0.3$): As shown in the bottom red region, when total resources are critically scarce, the satisfied demand remains consistently low regardless of the transportation budget $w_2$. This indicates that resource availability is the dominant hard constraint in this region.
- Budget-Constrained Segment ($w_1 > 0.6, w_2 < 0.4$): In the top-left region, although resources are abundant, the low transportation budget limits the distribution, preventing high demand satisfaction.
- Sensitive Transition Segment : The region around $w_1 = 0.5$ and $w_2 = 0.4$ (indicated by the yellow-green transition) represents both constraints are binding. In this region, increasing either resource or budget yields significant improvements, but neither is abundant.

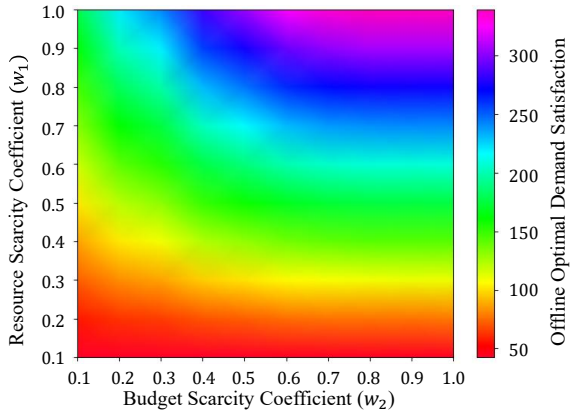


Fig. 5. Sensitivity analysis of resource and budget scarcity coefficients on demand satisfaction (Offline Optimal).

TABLE III
IMPACT OF TRAINING DATA DISTRIBUTION ON GENERALIZATION (FIXED VS. VARIABLE SCARCITY)

| Dataset | Model | Acc ↑ | CR ↑ | OR ↓ |
|---|---|---|---|---|
| **Synthetic** | FLOOD-Fixed | **0.6027** | **0.8429** | 0.1293 |
| | FLOOD-Var | 0.5876 | 0.8029 | **0.1095** |
| **Real-world** | FLOOD-Fixed | 0.5973 | 0.7413 | 0.2119 |
| | FLOOD-Var | **0.6075** | **0.7632** | **0.1928** |

Based on this analysis, we selected $w_1 = 0.5$ and $w_2 = 0.4$ for our primary dataset construction. This parameter combination ensures the decision environment is neither trivially easy (purple region) nor impossibly constrained (red region), providing a challenging benchmark to rigorously evaluate the model's allocation capabilities.

**Impact of Scarcity in Training.** Finally, we investigated whether training on diverse scarcity levels improves robustness. We compared two training strategies:

1) **FLOOD-Fixed:** Trained on data with fixed scarcity coefficients ($w_1 = 0.5, w_2 = 0.4$).
2) **FLOOD-Var:** Trained on data with dynamic scarcity coefficients sampled randomly from a moderate range ($w_1 \in [0.4, 0.65], w_2 \in [0.3, 0.7]$).

Table III reports the performance on both the Synthetic Test Set (same distribution as Fixed) and the Real-world Data (out-of-distribution). The results suggest an interesting trade-off between specialization and generalization. *FLOOD-Fixed* outperforms *FLOOD-Var* on the Synthetic Data, because the test set parameters align perfectly with its training distribution. However, on Real-world Data, the trend reverses: *FLOOD-Var* achieves higher Accuracy and CR with lower OR. This suggests that exposing the model to varying degrees of resource scarcity during training prevents overfitting to specific constraint ratios. The dynamic training environment forces the model to learn a more robust policy, thereby enhancing its capability to extrapolate to unseen, real-world scenarios.

## V. CONCLUSIONS

In this work, we introduced FLOOD, a framework that transforms the challenge of online algorithm design into a tractable offline decision problem. By leveraging offline oracle to generate optimal trajectories, FLOOD provides a general oracle distillation paradigm. The main findings are as follows:

- **Feasibility of FLOOD:** Through designed fine-tuning strategy, a general LLM acquires specialized online decision capabilities, significantly outperforming base LLM and achieving parity with domain-specific algorithms.
- **Importance of Temporal Alignment:** The Decision Period Aligned Gradient Update strategy is a structural necessity when fine-tuning online decision LLM. It is crucial for capturing the long-horizon temporal dependencies inherent in sequential decision-making, which standard fine-tuning fails to address.
- **The Accuracy-Optimality Gap:** Our analysis indicates a critical insight: high local alignment with oracle actions does not strictly guarantee global optimality.

Despite the promising results, our study also revealed some possible directions. Building on the observed inconsistency between accuracy and other performance, it is worth studying objectives that directly optimize regret or competitive ratio, as well as feasibility mechanisms to better satisfy resource constraints in safety-critical deployments.

## ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (No. 72301291).

## REFERENCES

[1] X. Li and Y. Ye, "Online linear programming: Dual convergence, new algorithms, and regret bounds," *Operations Research*, vol. 70, no. 5, pp. 2948–2966, 2022.

[2] A. Kalai and S. Vempala, "Efficient algorithms for online decision problems," *Journal of Computer and System Sciences*, vol. 71, no. 3, pp. 291–307, 2005.

[3] A. Bietti, A. Agarwal, and J. Langford, "A contextual bandit bake-off," *Journal of Machine Learning Research*, vol. 22, no. 133, pp. 1–49, 2021.

[4] Y. Wu, N. Ge, Y. Li, L. Qian, M. Zhu, H. Yang, H. Chen, and J. Wu, "Can large language models tackle graph partitioning?" in *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025, pp. 15 710–15 730.

[5] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 31 967–31 987, 2023.

[6] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large language models as optimizers," in *International Conference on Learning Representations*, 2024.

[7] C. Park, X. Liu, A. E. Ozdaglar, and K. Zhang, "Do LLM agents have regret? a case study in online learning and games," in *International Conference on Learning Representations*, 2025.

[8] A. Krishnamurthy, K. Harris, D. J. Foster, C. Zhang, and A. Slivkins, "Can large language models explore in-context?" *Advances in Neural Information Processing Systems*, vol. 37, pp. 120 124–120 158, 2024.

[9] F. Xia, H. Liu, Y. Yue, and T. Li, "Beyond numeric rewards: In-context dueling bandits with LLM agents," in *Findings of the Association for Computational Linguistics*, 2025, pp. 9959–9988.

[10] D. Chen, Q. Zhang, and Y. Zhu, "Efficient sequential decision making with large language models," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 9157–9170.

[11] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[12] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 1998.

[13] L. Andrew, S. Barman, K. Ligett, M. Lin, A. Meyerson, A. Roytman, and A. Wierman, "A tale of two metrics: Simultaneous bounds on competitiveness and regret," in *Conference on Learning Theory*. PMLR, 2013, pp. 741–763.

[14] A. Mehta, "Online matching and ad allocation," *Foundations and Trends® in Theoretical Computer Science*, vol. 8, no. 4, pp. 265–368, 2010.

[15] R. Fleischer and M. Wahl, "On-line scheduling revisited," *Journal of Scheduling*, vol. 3, no. 6, pp. 343–353, 2000.

[16] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proceedings of the 19th International Conference on World Wide Web*, 2010.

[17] Y. Li, L. Liu, X. Wang, Z. Mao, J. Wu, and W. Bao, "A causal target for learning to defer under hidden confounding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, no. 28, 2026, pp. 23 248–23 255.

[18] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[19] J. Sun, Z. Wang, R. Yang, C. Xiao, J. Lui, and Z. Dai, "Large language model-enhanced multi-armed bandits," *arXiv preprint arXiv:2502.01118*, 2025.

[20] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *International Conference on Learning Representations*, 2023.

[21] P. Alamdari, Y. Cao, and K. Wilson, "Jump starting bandits with llm-generated prior knowledge," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 19 821–19 833.

[22] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, vol. 1, 1988.

[23] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 627–635.

[24] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[25] C.-Y. Hsieh, C.-L. Wu, C.-K. Yu, Y.-S. Chen, H.-y. Lee, M. Sun *et al.*, "Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023, pp. 8003–8017.

[26] L. C. Magister, J. Mallinson, J. Adamek, G. Poesia, and J. Andreas, "Teaching small language models to reason," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023, pp. 1773–1781.

[27] N. Ho, L. Schmid, and S.-Y. Yun, "Large language models are reasoning teachers," in *Proceedings of the 61st annual meeting of the association for computational linguistics*, 2023, pp. 14 852–14 882.

[28] H. Yang, K. Xiao, L. Liu, H. Huang, and W. Zhang, "An online learning approach towards far-sighted emergency relief planning under intentional attacks in conflict areas." in *International Joint Conference on Artificial Intelligence*, 2022, pp. 4679–4685.

[29] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," in *International Conference on Learning Representations*, 2021.

[30] T. GLM, A. Zeng, B. Xu, B. Wang, C. Zhang, D. Yin, D. Zhang, D. Rojas, G. Feng, H. Zhao *et al.*, "Chatglm: A family of large language models from glm-130b to glm-4 all tools," *arXiv preprint arXiv:2406.12793*, 2024.

[31] World Health Organization, "Ebola virus disease, democratic republic of the congo, 2020," [Online]. Available: https://www.who.int/emergencies/disease-outbreak-news/item/2020-DON281.

[32] G. Erdélyi, O. J. Erdélyi, and V. Estivill-Castro, "Randomized classifiers vs human decision-makers: Trustworthy ai may have to act randomly and society seems to accept this," *arXiv preprint arXiv:2111.07545*, 2021.