RL-MPCA: A Reinforcement Learning Based Multi-Phase Computation Allocation Approach for Recommender Systems

Jiahong Zhou* Meituan, Beijing, China zhoujiahong02@meituan.com

Bo Tang Meituan, Beijing, China tangbo17@meituan.com Shunhui Mao Meituan, Beijing, China maoshunhui@meituan.com

Qianlong Xie Meituan, Beijing, China xieqianlong@meituan.com Guoliang Yang Meituan, Beijing, China yangguoliang@meituan.com

Lebin Lin Meituan, Beijing, China linlebin@meituan.com

Xingxing Wang Meituan, Beijing, China wangxingxing04@meituan.com

ABSTRACT

Recommender systems aim to recommend the most suitable items to users from a large number of candidates. Their computation cost grows as the number of user requests and the complexity of services (or models) increases. Under the limitation of computation resources (CRs), how to make a trade-off between computation cost and business revenue becomes an essential question. The existing studies focus on dynamically allocating CRs in queue truncation scenarios (i.e., allocating the size of candidates), and formulate the CR allocation problem as an optimization problem with constraints. Some of them focus on single-phase CR allocation, and others focus on multi-phase CR allocation but introduce some assumptions about queue truncation scenarios. However, these assumptions do not hold in other scenarios, such as retrieval channel selection and prediction model selection. Moreover, existing studies ignore the state transition process of requests between different phases, limiting the effectiveness of their approaches.

This paper proposes a Reinforcement Learning (RL) based Multi-Phase Computation Allocation approach (RL-MPCA), which aims to maximize the total business revenue under the limitation of CRs. RL-MPCA formulates the CR allocation problem as a Weakly Coupled MDP problem and solves it with an RL-based approach. Specifically, RL-MPCA designs a novel deep Q-network to adapt to various CR allocation scenarios, and calibrates the Q-value by introducing multiple adaptive Lagrange multipliers (adaptive- λ) to avoid violating the global CR constraints. Finally, experiments on the of-fline simulation environment and online real-world recommender system validate the effectiveness of our approach.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WWW '23, April 30-May 4, 2023, Austin, TX, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 978-1-4503-9416-1/23/04...\$15.00 https://doi.org/10.1145/3543507.3583313

Dong Wang Meituan, Beijing, China wangdong07@meituan.com

CCS CONCEPTS

Information systems → Recommender systems; Online advertising; Computational advertising.

KEYWORDS

Computation Resource Allocation, Deep Reinforcement Learning, Recommender System, Weakly Coupled MDP

ACM Reference Format:

Jiahong Zhou, Shunhui Mao, Guoliang Yang, Bo Tang, Qianlong Xie, Lebin Lin, Xingxing Wang, and Dong Wang. 2023. RL-MPCA: A Reinforcement Learning Based Multi-Phase Computation Allocation Approach for Recommender Systems. In *Proceedings of the ACM Web Conference 2023 (WWW '23), April 30-May 4, 2023, Austin, TX, USA*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3543507.3583313

1 INTRODUCTION

Recommender systems aim to recommend the most suitable items to users from a large number of candidates and expect to gain revenue from users' views, clicks, and purchases. They are playing an increasingly important role in e-commerce platforms [20].

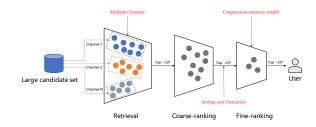


Figure 1: The typical structure of recommender systems.

Industrial recommender systems are often designed as cascading architectures [11, 24]. As shown in Figure 1, a typical recommender system consists of several stages, including retrieval, coarseranking, fine-ranking, etc. In these stages, online advertising systems (a kind of recommender system applied to online advertising) generally contain several computation-intensive services or models, including bid models [19, 37], prediction services [14, 43], etc. These

^{*}Corresponding author

services require a lot of computation resources (CRs). Take the display advertising system of Meituan Waimai platform (hereinafter referred to as Meituan advertising system), for example. It consumes a lot of CRs in both the retrieval stage and the fine-ranking stage. As the number of user requests increases dramatically, the system's CR consumption rises accordingly. Due to the limitation of CRs, recommender systems need to make a trade-off between CR cost and business revenue when the traffic exceeds the system load. From the perspective of CR utilization efficiency, the goal of recommender systems is to maximize the total business revenue under the CR constraint.

To address the challenges of huge traffic and a large number of candidate items, the real-world recommender systems usually use two types of strategies: static strategies and dynamic strategies [21, 38]. Static strategies select suitable fixed rules through stress testing and practical experience to allocate CRs. They also provide fixed downgrades to cope with unexpected traffic. Static strategies require constant manual intervention to adapt to quick changes in traffic, and fixed downgrades provided by static strategies are generally detrimental to business revenue and user experience. Dynamic strategies [21, 38] dynamically allocate CRs for requests based on the value of requests. They prioritize allocating CRs to more valuable requests to achieve better revenue. Compared to static strategies, dynamic strategies are more efficient in utilizing CRs and require fewer manual intervention.

Recommender systems with multiple stages have various CR allocation scenarios. Based on the application scenario, we summarize the dynamic CR allocation methods into three types: **Elastic Channel**, **Elastic Queue**, and **Elastic Model**:

- Elastic Channel: dynamically adjust the retrieval strategy. A typical recommender system contains multiple retrieval channels. When CRs are insufficient, static strategies usually use fixed rules to drop some retrieval channels with high computation consumption. Different to static strategies, *Elastic Channel* dynamically adjusts the retrieval strategy for each request according to the online environment and the features of the request.
- Elastic Queue: dynamically adjust the length of queue. Under the limitation of CRs, recommender systems cannot provide the prediction service and ranking service for all candidate items. In static strategies, before entering the prediction service and ranking service, the queue of items needs to be truncated to a global fixed length. In contrast, *Elastic Queue* dynamically adjusts truncation for each request length according to the online environment and the features of the request.
- Elastic Model: dynamically select prediction models. Recommender systems often provide multiple prediction models with different computation consumption for one prediction service. A complex model achieves better revenue while taking more computation consumption. When CRs are insufficient, static strategies usually use fixed rules to downgrade high computation consumption models to low consumption models. In contrast, Elastic Model dynamically adjusts the prediction model for each request according to the online environment and the features of the request.

Recently, some dynamic strategies [21, 38] have been proposed to achieve "personalized" CR allocation. DCAF [21] focuses on a single CR allocation phase. CRAS [38] focuses on multi-phase queue truncation problems, but it introduces some assumptions about Elastic Queue scenario. For example, it uses the queue length to represent the computation cost when modeling the CR allocation problem, and assumes that the revenue varies logarithmically with the queue length. However, these assumptions do not hold in Elastic Channel and Elastic Model scenarios. Moreover, existing studies ignore the state transition process of requests between different phases, which limits the effectiveness of their approaches.

To address the limitations of existing studies, we propose RL-MPCA, which formulates the CR allocation problem as a Weakly Coupled Markov Decision Process (Weakly Coupled MDP) [26] problem and solves it with an RL-based approach. Compared to Constrained Markov Decision Process (CMDP) [4], Weakly Coupled MDP allows global weakly coupled constraints across sub-MDPs. Thus, it can model the problem of CR allocation across requests better than CMDP [2, 7, 10].

Our main contributions are summarized as follows:

- (1) We propose an innovative CR allocation solution for recommender systems. To the best of our knowledge, this is the first work that formulates the CR allocation problem as a Weakly Coupled MDP problem and solves it with an RL-based approach.
- (2) We design a novel multi-scenario compatible Q-network adapting to the various CR allocation scenarios, then calibrate Q-value by introducing multiple adaptive Lagrange multipliers (adaptive-λ) to avoid violating the global CR constraints in training and serving.
- (3) We validate the effectiveness of our proposed RL-MPCA³ approach through offline experiments and online A/B tests. Offline experiment results show that RL-MPCA can achieve better revenue than baseline approaches while satisfying the CR constraints. Online A/B tests demonstrate the effectiveness of RL-MPCA in real-world industrial applications.

2 RELATED WORK

2.1 CR Allocation and RL for Recommender Systems

Recommender systems have been a popular topic in industry and academia in recent years. Most studies focus on improving the business revenue under the assumption of sufficient CRs [41, 42]. Some of these studies focus on applying RL to recommender systems, including recommendations [12, 18, 44], real-time bidding[30, 35], ad slots allocation [23, 36, 41], etc. Some studies concern CR consumption and try to reduce it through model compression [13, 28]. All the above studies rarely focus on CR allocation. As an exception, DCAF [21] and CRAS [38] propose two "personalized" CR allocation approaches. They formulate the Elastic Queue CR allocation problem as an optimization problem, and then solve it with linear programming algorithms. Different from the above studies, our proposed RL-MPCA uses an RL-based dynamic CR allocation approach to improve the effectiveness.

 $^{^1\}mathrm{In}$ general, computation resources include CPU/GPU computing capacity, memory capacity and resonnse time, etc.

²https://waimai.meituan.com/, one of the largest e-commerce platforms in China.

 $^{^3}$ The publicly accessible code at https://anonymous.4open.science/r/RL-MPCA-130D.

RL and Weakly Coupled MDPs 2.2

A Weakly Coupled MDP [26] comprises multiple sub-MDPs, which are independent except that global resource constraints weakly couple them [10]. Due to the linking constraints, the scale of the problem grows exponentially in the number of sub-problems [10]. Some studies try to relax Weakly Coupled MDP to CMDP [4] and then solve it [2, 10]. The solutions to the CMDP problem include CPO [1], RCPO [31], IPO [25], etc. They focus on the internal constraints of MDP. Recently, some studies focus on directly solving Weakly Coupled MDP problems. $BCORLE(\lambda)$ [40] solves it with λ -generalization. BCRLSP [9] first trains the unconstrained reinforcement model and then imposes a global constraint on the model with linear programming methods in near real-time. Both BCORLE and BCRLSP guarantee that budget allocations strictly satisfy a single global constraint. CrossDQN [23] attempts to make the model avoid violating a single global constraint by introducing auxiliary batch-level loss. It uses a soft version of argmax to solve the problem of non-derivability of the native argmax function, which makes the model unable to strictly satisfy the global constraints during both offline training and online serving.

Offline RL methods aim to learn effective policies from a fixed dataset without further interaction with the environment [17]. Offpolicy methods (e.g., DQN [27], DDQN [32]) can be directly applied to Offline RL while ignoring the out-of-distribution (OOD) problem. To solve the OOD problem, some offline RL methods are also proposed, including BCQ [16], CQL [22], COMBO [39], etc. BCQ addresses the problem of extrapolation error via restricting the action space to force the agent towards behaving close to on-policy with respect to a subset of the given data. In addition, REM [3] enforces optimal Bellman consistency on random convex combinations of multiple Q-value estimates to enhance the generalization capability in the offline setting. In the experiments of this paper, we choose three popular methods (DDQN, BCQ, and REM) as base models. Essentially, our proposed RL-MPCA only modifies the Q-network, so it can also apply to other Q-learning methods.

In addition, we can also consider the Weakly Coupled MDP problem as a black-box optimization problem, then solve it with evolutionary algorithms, such as Cross-Entropy Method (CEM) [29] and Natural Evolution Strategies (NES) [34].

3 PROBLEM FORMULATION

Original Problem Description

The recent work [21] formulated the single-phase CR allocation problem as a knapsack problem. Similarly, we formulate the multiphase CR allocation problem as a knapsack problem.

$$\max_{j_1,\dots,j_T} \sum_{i=1}^{M} \sum_{j_1} \dots \sum_{j_T} \left(\prod_{t=1}^{T} x_{i,j_t} \right) Value_{i,j_t,\dots,j_T}$$
 (1)

$$s.t. \sum_{i=1}^{M} \sum_{j_1} \cdots \sum_{j_T} \left(\prod_{t=1}^{T} x_{i,j_t} \right) Cost_{i,j_t,\dots,j_T} \le C$$

$$\sum_{j_t} x_{i,j_t} \le 1, \quad \forall i,t$$

$$(3)$$

$$\sum_{i_t} x_{i,j_t} \le 1, \quad \forall i, t \tag{3}$$

$$x_{i,j_t} \in \{0,1\}, \ \forall i, j_t$$
 (4)

We suppose there are M online requests $\{i = 1, ..., M\}$ in a given time slice, and the maximum computation budget of the system in this time slice is C. For each request i, T phases need to make computation decisions, and N_t actions can be taken for the specified phase t. We define j_1, \ldots, j_T as a complete decision process of a request, and the decision action of phase t is j_t ($j_t \in \{1, ..., N_t\}$). Meanwhile, for request i, if the decision process is j_1, \ldots, j_T , we use $Value_{i,j_i,...,j_T}$ and $Cost_{i,j_i,...,j_T}$ to represent the expected revenue and computation cost, respectively. x_{i,j_t} is the indicator that request *i* is assigned action *j* in phase *t*. In phase *t*, for request *i*, there is one and only one action j_t can be taken.

InEq. (2) above assumes that all phases share an overall computation budget. However, in a real-world online recommender system, the CRs of each phase are often relatively independent. For example, recommender systems often deploy prediction and retrieval services on different clusters for ease of maintenance, and their CRs cannot be shared. Considering that each phase has a separate CRs budget, we replace the global constraint (InEq. (2)) with multiple constraints InEq. (5), where $Cost_{i,j_t}$ represents the computation cost when the decision of phase t is j_t for request i, and C_t is the computation budget of phase t. This paper focuses on the scenario of single-constraint CR allocation at each phase. If there is more than one constraint per phase, we can relax multiple constraints in the same phase and combine them into one.

$$s.t. \sum_{i=1}^{M} \sum_{j_t=1}^{N_t} x_{i,j_t} Cost_{i,j_t} \le C_t, \ \forall t = 1, ..., T$$
 (5)

Weakly Coupled MDP Problem Formulation

The decision results before phase t affect the input state of phase t. To better describe our approach, we take a three-phase CR allocation situation as an example in this paper. It contains one Elastic Channel phase, one Elastic Queue phase, and one Elastic Model phase, which is a typical case of recommender system CR allocation. As shown in Figure 2, for request i, the decision result of Elastic Channel phase determines the real retrieval queue, and it directly affects the input state of Elastic Queue phase. Similarly, the decision result of Elastic Queue phase affects the input state of Elastic Model phase. Therefore, to better adapt to the state transition process, in the multi-phase joint CR allocation, we introduce the "state" of the request.

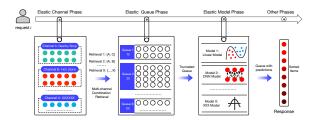


Figure 2: Request query procedure of recommender systems in a three-phase computation resource allocation situation.

In this paper, we formulate the CR allocation problem as a Weakly Coupled MDP [26] problem. Formally, the Weakly Coupled MDP consists of a tuple of six elements $(S, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma, C)$, which are defined as follows:

- **State Space** S. For phase t of request i, $s_t^i \in S$ consists of user information u, time slice information ts, context information c, ad items information $\{ad_1, \ldots, ad_{N_{ad}}\}$, and the CR allocation decision results of phase t-1.
- Action Space A. Our CR allocation situation has three phases with different action spaces. The actions of the Elastic Channel phase, the Elastic Queue phase, and the Elastic Model phase are the retrieval strategy number, the truncation length, and the prediction model number, respectively.
- **Reward** \mathcal{R} . For request i, after the agent takes action for the final phase, the system returns the final sorted items to the user. The user browses the items and gives feedback, including order price $price_o$ and advertising fee fee_{ad} of request i. The reward $r(s_t, a_t)$ is the weighted sum of them:

$$r(s_t, a_t) = k_1 * fee_{ad} + k_2 * price_o$$
 (6)

- Transition Probability \mathcal{P} . $P(s_{t+1}|s_t, a_t)$ is the state transition probability from phase t to phase t+1 after taking action a_t . For each request i, trajectory (τ_i) is its whole state transition process in the recommender system.
- **Discount Factor** γ . $\gamma \in [0, 1]$ is the discount factor for future rewards.
- **Global Constraint** *C*. Each phase has its global constraint that couples sub-MDPs. InEq. (5) defines these constraints.

4 METHODOLOGY

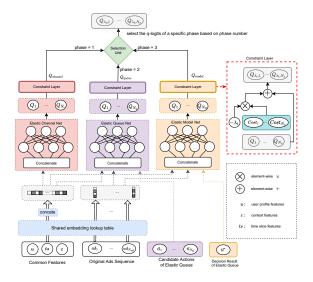


Figure 3: Q-Network of RL-MPCA. It first models each phase using a separate network and calibrates the Q-value with the constraint layer. Then the selection unit selects the q-logits of a specific phase based on the phase number t.

Deep Q-Network (DQN) [27] and its improved versions [16, 32, 33] are very popular in solving sub-MDPs with discrete actions. The Q-network is the essential structure of these models, $Q_{\theta}(s, a)$. To adapt to various CR allocation scenarios, we design a novel deep

Q-Network with multiple separate networks (As Figure 3 shows). In particular, the state space of each phase is defined as follows:

- In the Elastic Channel phase, the candidate action space is the retrieval strategy numbers. For a recommender system with N_r retrieval channels, the number of retrieval strategies is $N_c = 2^{N_r}$ and the candidate action space is $\{1, \ldots, N_c\}$. For example, for three candidate retrieval channels $\{A, B, C\}$, retrieval strategy (0, 1, 1) indicates that channel A is not retrieved, and channels B and C are retrieved. We convert the indicator vector as a binary value, then the strategy number of the strategy (0, 1, 1) is the integer 3.
- In the Elastic Queue phase, the action space is the truncation length. To reduce the candidate action space, we can put the candidate actions into buckets, e.g., set every ten adjacent truncation lengths as one bucket. Then the candidate action space is $\{10, 20, \dots\}$.
- In the Elastic Model phase, the candidate action space is the prediction model numbers $\{1, \ldots, N_m\}$.

Each phase of CR allocation has its own action spaces. As Figure 3 shows, we model each phase using separate networks to adapt the different action spaces. In the last layer of the Q-Network, we use the selection unit to select the q-logits of a specific phase based on phase number t.

4.1 Constraint Layer

For any phase t, suppose that we have the optimal policy $\pi_{\neg t}^*$ that satisfies the constraints of all phases except t. Then the decision problem for current phase t can be modeled separately as the following single-phase CR allocation problem with a single constraint:

$$\max_{a_t} \sum_{i=1}^{M} \sum_{a_t=1}^{N_t} x_{i,a_t} Value_{i,a_t}$$
 (7)

$$s.t. \sum_{i=1}^{M} \sum_{a_t=1}^{N_t} x_{i,a_t} Cost_{i,a_t} \le C_t$$
 (8)

$$\sum_{a_t=1}^{N_t} x_{i,a_t} \le 1, \quad \forall i \tag{9}$$

$$x_{i,a_t} \in \{0,1\}, \ \forall i, a_t$$
 (10)

By constructing and solving the Lagrange dual problem, we have the optimal solution to this problem. The proof is provided in Appendix B. For request i, the optimal action of phase t is a_t^* :

$$a_t^* = \underset{a_t}{\arg\max}(Value_{i,a_t} - \lambda_t Cost_{i,a_t})$$
 (11)

where $\lambda_t \geq 0$ is the Lagrange multiplier.

Further, we use $Q^{\pi_{\neg t}^*}(s_t, a_t)$ to represent the expected cumulative reward for taking action a_t in state s_t and subsequent actions are decided following policy $\pi_{\neg t}^*$.

$$Q^{\pi_{\neg t}^*}(s_t, a_t) = \mathbb{E}_{\tau \sim \pi_{\neg t}^*}[R_t | s_t, a_t]$$
 (12)

$$R_t = \sum_{i=t+1}^{\infty} \gamma^i r(s_i, a_i, s_{i+1})$$
 (13)

For phase t of request i, we have:

$$Q^{\pi_{\neg t}^*}(s_t, a_t) = Value_{i, a_t}$$
 (14)

$$Cost(s_t, a_t) = Cost_{i,a_t} \tag{15}$$

where $Cost(s_t, a_t)$ is the computation cost for taking a_t in s_t , determined by (s_t, a_t) , and independent of both prior and subsequent strategies. Thus, for phase t, the optimal action for request i in state s_t is a_t^* :

$$a_t^* = \arg\max_{a_t} (Q^{\pi_{\neg t}^*}(s_t, a_t) - \lambda_t Cost(s_t, a_t))$$
 (16)

Compared to the action selection formula in original DON [27] networks, we only need to add a layer (Constraint Layer) to obtain the optimal action that satisfies the CR constraints.

$$Q_{\lambda_{+}}^{\pi_{-t}^{*}}(s_{t}, a_{t}) = Q_{-t}^{\pi_{-t}^{*}}(s_{t}, a_{t}) - \lambda_{t} Cost(s_{t}, a_{t})$$
 (17)

Then the optimal action is:

$$a_t^* = \arg\max_{a_t} Q_{\lambda_t}^{\pi_{-t}^*}(s_t, a_t)$$
 (18)

Adaptive-λ in Offline Model Training. As mentioned in DCAF [21] and CRAS [38], Assumptions (4.1) and (4.2) usually hold in general recommender systems.

Assumption 4.1. $Value_{i,a_t}$ is monotonically increasing with $Cost_{i,a_t}$.

Assumption 4.2. $\frac{Value_{i,a_t}}{Cost_{i,a_t}}$ is monotonically decreasing with $Cost_{i,a_t}$.

From our observations, they also hold for most requests in Meituan advertising system. However, it is worth noting that our assumptions differ from those of CRAS. CRAS uses the queue length to represent the computation cost, while we make no assumptions about the relationship between computation cost and queue length (or other actions).

Given a fixed $\{Value_{i,a_t}\}_{i=1}^M$ and variable λ_t , the optimal action of phase t is a_t^* (Eq. 11). For each request i, its optimal action a_t^* varies with λ_t , then the total computation cost $\hat{C}_t(\lambda_t)$ (Eq. 19) and the total revenue $\hat{R}_t(\lambda_t)$ (Eq. 20) of phase t vary with λ_t .

$$\hat{C}_t(\lambda_t) = \sum_{i=1}^{M} Cost_{i,a_t^*}$$
(19)

$$\hat{R}_t(\lambda_t) = \sum_{i=1}^{M} Value_{i, a_t^*}$$
(20)

We can obtain the optimal λ_t which satisfies the CR constraint and maximizes $\hat{R}_t(\lambda_t)$ through updating λ_t iteratively based on $\hat{C}_t(\lambda_t)$.

Lemma 4.1. Suppose Assumptions (4.1) and (4.2) hold, for any λ_t^k , let λ_t^{k+1} be:

$$\lambda_t^{k+1} \leftarrow \lambda_t^k + \alpha \left(\frac{\hat{C}_t(\lambda_t^k)}{C_t} - 1 \right)$$
 (21)

where C_t is the computation budget of phase t and $\alpha \in \mathbb{R}^+$ is learning rate of λ . Then, the following conclusion holds:

- Conclusion 1. $\hat{C}_t(\lambda_t^{k+1}) \leq \hat{C}_t(\lambda_t^k)$ will holds if $\hat{C}_t(\lambda_t^k) > C_t$. Conclusion 2. $\hat{R}_t(\lambda_t^{k+1}) \geq \hat{R}_t(\lambda_t^k)$ will holds if $\hat{C}_t(\lambda_t^k) < C_t$. Conclusion 3. $\lambda_t^{k+1} = \lambda_t^k$ will holds if $\hat{C}_t(\lambda_t^k) = C_t$.

PROOF. Suppose Assumptions (4.1) and (4.2) hold, $\hat{C}_t(\lambda_t)$ is monotonically decreasing with λ_t (see more details in [21]). Further, under

Assumption 4.2, $\frac{\hat{R}_t(\lambda_t)}{\hat{C}_t(\lambda_t)}$ is monotonically decreasing with λ_t , and under Assumption 4.1, $\hat{R}_t(\lambda_t)$ is monotonically decreasing with λ_t .

- when $\hat{C}_t(\lambda_t^k) > C_t$, we have $\lambda_t^{k+1} > \lambda_t^k$, then $\hat{C}_t(\lambda_t^{k+1}) \le$
- when $\hat{C}_t(\lambda_t^k) < C_t$, we have $\lambda_t^{k+1} < \lambda_t^k$, then $\hat{R}_t(\lambda_t^{k+1}) \ge$
- when $\hat{C}_t(\lambda_t^k) = C_t$, we have $\lambda_t^{k+1} = \lambda_t^k$ holds.

In summary, Lemma (4.1) specifies that it is feasible to update λ with formula (21). Initially, conclusion 1 of Lemma (4.1) indicates that when the total computation cost exceeds the computation budget, updating λ_t with formula (21) will obtain less total computation cost. It helps to avoid violating the constraint. Furthermore, conclusion 2 of Lemma (4.1) indicates that when the total computation cost is less than computation budget, updating λ_t with formula (21) will obtain a better total revenue. Finally, conclusion 3 of Lemma (4.1) indicates that when the total computation cost equals to computation budget, updating λ_t with formula (21) will obtain the original value of λ_t . Updating λ_t with formula (21) until convergence, we will obtain the optimal λ_t^* , where $\hat{C}_t(\lambda_t^*) = C_t$.

Algorithm 1 Offline Training of RL-MPCA (Based on DDQN)

Input: Dataset \mathcal{D} , number of iteration I, mini-batch size N, adaptive- λ update times K

- 1: Initialize Q-network Q_{θ} , target Q net $Q_{\theta'}$ ($\theta' \leftarrow \theta$), Lagrange multipliers $\lambda = (\lambda_1, \dots, \lambda_T)$
- 2: **for** i = 1, ..., I **do**
- Sample batch \mathcal{D}^i of N transitions (s_t, a_t, r_t, s_{t+1}) from \mathcal{D}
- $a_{t+1} = \arg\max_{a_{t+1}} \left(Q_{\theta}(s_{t+1}, a_{t+1}) \lambda_{t+1}^{i} Cost(s_{t+1}, a_{t+1}) \right)$
- $\theta \leftarrow \arg\min_{\theta} \sum_{\mathcal{D}^i} (r_t + \gamma Q_{\theta'}(s_{t+1}, a_{t+1}) Q_{\theta}(s_t, a_t))^2$ **for** k = 1, ..., K **do** For $s_t \in \mathcal{D}^i$, take a_t^k with (22)

- For $t \in \{1, ..., T\}$, update $\lambda_t^{i,k+1}$ with (23)

- Every N_{target} steps reset $\theta' \leftarrow \theta$
- 12: end for

Output: Q_{θ} , $\lambda = (\lambda_1, ..., \lambda_T)$

As described in Algorithm 1, we dynamically update the λ in the offline training phase. At iteration step i, we take a mini-batch of samples \mathcal{D}_i (a bigger batch is generally taken here, e.g., 8192 samples per batch), and update $\lambda = (\lambda_1, \dots, \lambda_T) K$ times. At the k-th update, for each s_t in \mathcal{D}_i , take action a_t^k with:

$$a_t^k = \underset{a_t}{\arg\max} \left(Q_{\theta}(s_t, a_t) - \lambda_t^{i,k} Cost(s_t, a_t) \right)$$
 (22)

and for each phase $t \in \{1, ..., T\}$ at the k-th update, update

$$\lambda_t^{i,k+1} \leftarrow \max \left\{ 0, \lambda_t^{i,k} + \alpha \left(\frac{\sum_{(s_t, a_t^k) \in \mathcal{D}^{i,k}} Cost(s_t, a_t^k)}{C_t(\mathcal{D}^i)} - 1 \right) \right\}$$
 (23)

where $\alpha \in \mathbb{R}^+$ is the learning rate of adaptive- λ , and $C_t(\mathcal{D}_i)$ is the maximum CR budget that the system can allocate for dataset \mathcal{D}_i at phase t. $C_t(\mathcal{D}_i)$ can be calculated through an offline fixed rule, which is designed by stress testing and practical experience.

Algorithm 1 describes the training process of DDQN-based RL-MPCA. Essentially, the Constraint Layer module of RL-MPCA only modifies the Q-network, so it can also apply to other Q-learning methods. Take a popular offline RL method with Q-network, REM [3], for example. The only difference between Algorithm 1 and the REM-based RL-MPCA approach is Q-network Q_{θ} . Specifically, for REM model with H heads, we replace Q_{θ} with $Q_{\theta}^{REM} = \sum_h \beta_h Q_{\theta}^h$, where $\boldsymbol{\beta} = (\beta_1, \dots, \beta_H)$ is categorical distribution, which is randomly drawed for each mini-batch (see more details in [3]).

4.1.2 λ Correction in Offline Model Evaluation . After the offline model is trained, the λ -calibrated Q-value guarantees that the agent's decisions satisfy the CR constraints on all training datasets. However, when applying λ to the real online system, it still faces the following problems: (1) The online and offline data distributions are inconsistent because the behavioral policy of collecting offline data differs from the target policy, which leads to the possibility that λ may not satisfy the CR constraints in the online system. (2) The traffic of the recommender system varies over time, and the existing λ cannot satisfy the CR constraints on each time slice.

To solve problem (1), we build an offline simulation system, which interacts with the agent and gives feedback on the computation cost and revenue in imitation of the real online environment. Through the evaluation in the simulation system, we select the optimal λ^* that satisfies the CR constraints in order of the decision phases. When both Assumptions (4.1) and (4.2) hold, we can find optimal λ^* through bisection search in each phase (please refer to [21] for the detailed proof). Otherwise, we can find optimal λ^* through grid search [6].

To solve problem (2), we select the optimal λ^* for each time slice based on the offline simulation system. We observe that the traffic of the recommender system generally varies periodically except for special holidays. Taking Meituan advertising system as an example, its traffic variation cycle is one day. Therefore, we can divide a day into multiple time slices with similar traffic distribution in the same time slice. Considering that the cost of training a separate model for each time slice is expensive and not easy to maintain, we first train a uniform model for all time slices, and then solve a separate λ for each time slice. Alternatively, for systems with non-periodic traffic, a possible solution is to use the traffic from the previous time slice to represent the current time slice. Specifically, we can update λ in near real-time, thus allowing λ to automatically adapt to irregular traffic changes.

4.2 System Architecture

We illustrate the overview of the architecture in Figure 4. In each phase, for instance, in the Elastic Queue phase, we need to allocate and control CRs through Computation Allocation System and Computation Control System. Computation Allocation System aims to maximize the total business revenue under the CR constraints. Computation Control System aims to guarantee system stability by means of feedback control. Dynamic allocation of CRs poses a significant challenge in guaranteeing the stability of recommender

systems. We use Flink [8] to collect real-time system load information, such as failure rate, CPU utilization, etc., and then use PID [5] control algorithms to achieve feedback control. When the system load exceeds the target value of the PID, the PID will control the consumption of CRs. For instance, in the Elastic Queue scenario, when the system's failure rate rises above the target value, the PID Controller will reduce the upper bound of the queue for all requests. The result of online A/B tests shows that the Computation Control System reduces the degradation rate by 0.1 percentage point, provides automatic and timely responses to unexpected traffic, and guarantees the stability of recommender systems.

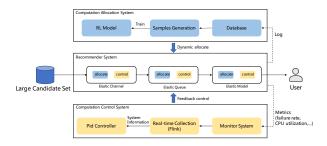


Figure 4: The Overview of System Architecture.

5 EXPERIMENTS

Our experiments aim to study four questions: (1) Does adaptive- λ of constraint layer help to avoid violating the global CR constraints in the training process? (2) After λ correction, does the model with constraint layer satisfy the global CR constraints and improve business revenue? (3) How does RL-MPCA approach perform in comparison to other state-of-the-art CR allocation approaches and RL algorithms? (4) How do different hyper-parameter settings affect the performance of RL-MPCA?

To answer these questions, we conduct various experiments in a three-phase joint modeling CR allocation situation, which contains one Elastic Channel phase, one Elastic Queue phase, and one Elastic Model phase.

5.1 Offline Experiments

To demonstrate the performance of the proposed RL-MPCA, we evaluate and compare various related approaches for CR allocation on a real-world dataset. In offline experiments, we use the simulation system to evaluate these approaches.

- 5.1.1 Dataset. We run random exploratory policies and superior policies (see more details about behavioral policies in Appendix F) to collect the dataset on Meituan advertising system during July and August 2022. Finally, we sample 568,842,204 requests from 101,368,290 users as the dataset, which includes user profile features, context features, time slice features, etc.
- 5.1.2 Offline Simulation System. It is dangerous to deploy a model to an online system when its effect is unknown, which may significantly damage the online revenue of the recommender system and cause the online service to crash. To solve this problem, we build an offline simulation system, which can imitate the online real-world environment to interact with the model (agent) and give feedback

on the computation consumption and revenue. More details about the offline simulation system are described in Appendix A.

5.1.3 Evaluation Metrics. We use computation (cost) and revenue (return) to evaluate the performance of approaches in offline experiments. The computation cost of each phase is defined as the sum of the CR consumption of all requests in that phase (see more details in Appendix C). To facilitate analysis, we define total computation cost as (cost = $\sum_t (\hat{C}_t - 1)$). return is defined as the total revenue of all requests, specifically, return = $\sum fee_{ad} + \sum price_{o}$. With reference to D4RL [15], to facilitate the analysis of the effectiveness of different approaches while ignoring the impact of our application scenarios, we normalize scores by:

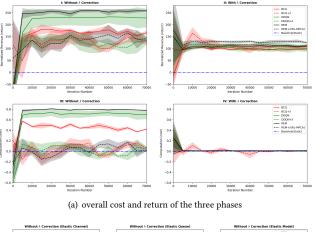
$$normalized_score = 100 * \frac{score - random_score}{expert_score - random_score}$$
 (24)

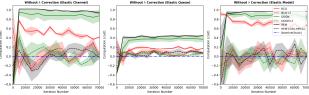
5.1.4 Hyper-parameters Settings. RL-MPCA contains several hyper-parameters. We employed the grid search [6] to determine the hyper-parameter values. Appendix D provides the hyper-parameters of experiments.

5.1.5 Baselines. We compare RL-MPCA with several baselines. Our situation has only one Elastic Queue phase (the two other phases are Elastic Channel and Elastic Model). In the situation containing only one Elastic Queue phase, the modeling methods of DCAF and CRAS are consistent. Therefore, in the later experiments, we only show the details of DCAF.

- **Static**. Static approach allocates CRs with global fixed rules, including fixed retrieval channels, fixed truncation length of candidate items, and fixed prediction models.
- DCAF. DCAF [21] formulates the CR allocation problem as an optimization problem with constraints, then solves the optimization problem with linear programming algorithms. In online A/B tests, we use fixed rules in Elastic Channel phase and Elastic Model phase, and DCAF is deployed in the Elastic Oueue phase.
- ES-MPCA. Before RL-MPCA, we designed an evolutionary strategies based multi-phase computation allocation approach (ES-MPCA, see more details in Appendix E), which has been deployed on Meituan advertising system.
- Ex-RCPO. RCPO [31] solves a CMDP problem by introducing the penalized reward functions (i.e., calibrate rewards with Lagrange multiplier λ). We replace adaptive-λ of RL-MPCA with the penalized reward functions when training the model, and name it Ex-RCPO.
- **Ex-BCORLE**(λ). BCORLE [40] solves a single-constraint budget allocation problem with λ -generalization. It cannot be directly applied to the multi-constraint CR allocation. We extend BCORLE from single- λ to multi- λ , and name it Ex-BCORLE.
- **Ex-BCRLSP**. BCRLSP [9] solves the single-constraint budget allocation problem by calibrating Q-value in near real-time. It cannot be directly applied to the multi-constraint CR allocation. We extend BCORLE from single-λ to multi-λ, and name it Ex-BCORLE.
- Ex-CrossDQN. CrossDQN [23] solves a single-constraint ads allocation problem by introducing auxiliary batch-level

loss when training the model. We replace adaptive- λ of RL-MPCA with auxiliary batch-level loss when training the model, and name it Ex-CrossDQN.





(b) cost at each phase without λ correction

Figure 5: Offline experiment results for adaptive- λ and λ correction on multiple Deep Q-Network models. Agents are evaluated every 5,000 steps, and averaged over 5 seeds.

5.1.6 Offline Experiment Results. To answer question (1) and question (2), we train multiple models: DDQN, BCQ, REM, and their improved versions of introducing adaptive- λ , then use the simulation system to evaluate them. As shown in Figure 5, during the training process, introducing adaptive- λ can control the CRs of the model always around the target constraints for each phase (Figure 5.a.III and Figure 5.b). However, the CRs of models swing around the target constraints due to the inconsistent distribution of the mini-batch sampled during training and the evaluation dataset (see more details in Section 4.1.2). After the λ correction, for each phase, the CRs of models strictly satisfy the constraints except for BCQ+ λ (BCQ with adaptive-λ), and Figure 5.a.IV shows the total CRs of all phases. Adaptive- λ allows the model to learn the Q-value under the case that CRs conform to the constraint (or in the near range of the constraints) at each phase. Thus, we can observe that the effectiveness of all three models improves after introducing adaptive- λ (Figure 5.a.II). An interesting phenomenon is that after introducing adaptive-λ, the CRs of BCQ instead cannot be stably calibrated to conform to the constraint. A potential reason is that the BCQ model contains an imitation component, which causes the BCQ model to imitate the behavioral strategy. As a result, the value of λ changes in an unknown direction during the training process, and eventually, the Q value cannot be calibrated to satisfy the target constraints.

	Before Calibration		After Calibration		
	cost	return	ConstraintSat	cost	return
Static	100%	-	Yes	100%	-
DCAF	-	-	Yes	$100(\pm 0.5)\%$	$52.7(\pm0.4)$
ES-MPCA	-	-	Yes	$100(\pm 0.5)\%$	$77.8(\pm 0.1)$
Ex-RCPO	$109.7(\pm 20.0)\%$	$146.9(\pm 65.4)$	Yes	$100(\pm 0.5)\%$	$111.8(\pm 8.4)$
Ex-BCORLE(λ)	-	-	Yes	$100(\pm 0.5)\%$	$74.2(\pm 36.1)$
Ex-CrossDQN	97.4(±5.6)%	$98.4(\pm 32.8)$	Yes	$100(\pm 0.5)\%$	$112.0(\pm 10.3)$
DDQN	172.6(±12.5)%	$227.6(\pm 16.2)$	Yes	$100(\pm 0.5)\%$	$100.0(\pm 12.9)$
BCQ	146.4(±7.6)%	$169.1(\pm 16.2)$	Yes	$100(\pm 0.5)\%$	$97.1(\pm 5.5)$
REM(Ex-BCRLSP)	179.2(±2.0)%	254.1(±11.8)	Yes	$100(\pm 0.5)\%$	$108.9(\pm 9.0)$
DDQN+ λ	102.7(±17.8)%	$125.3(\pm 31.8)$	Yes	$100(\pm 0.5)\%$	$115.4(\pm 7.4)$
BCQ+ λ	$101.5(\pm 17.4)\%$	$119.3(\pm 42.8)$	No/Yes	-/100(±0.5)%	$-/108.7(\pm 12.4)$
REM+ λ (RL-MPCA)	$103.4(\pm 18.3)\%$	$135.4(\pm 37.0)$	Yes	$100(\pm 0.5)\%$	$126.2(\pm 8.7)$

Table 1: The offline results in the simulation system. All numbers of return are the normalized score calculated by Eq. (24), where random_score and expert_score are the scores of Static and DDQN.

To answer question (3), we compare RL-MPCA to the state-ofthe-art CR allocation approach DCAF and other related approaches. The results are shown in Table 1. Experiment results show that RL-MPCA outperforms other approaches in *return* when the CR constraints are satisfied.

5.1.7 Hyper-parameter Analysis. To answer question (4), We compare the effect of two critical parameters, α and K, on the performance of RL-MPCA. α is the learning rate of adaptive- λ . K is λ update times in one global step.

Hyper-parameter α . Like the learning rate of the model's common parameters, the learning rate of adaptive- λ α cannot be too big or too small. Too small a learning rate will lead to slow learning, while too big a learning rate will cause λ to swing around the optimal value. Table 2 shows the model performance at different learning rates, and we finally choose 0.1 as the parameter value.

Hyper-parameter K. A bigger K indicates more updates to the λ at once update of the model parameter during training, which will make constraints easier to be satisfied. As seen in Table 2, the *return* increases with the increase of K. However, a bigger K also means more time consumption for training. To trade off the revenue and time, we choose 10 as the parameter value.

α		K			
α	return	K	return	training-time	
0.001	117.2(±6.5)	1	118.8(±14.1)	100%	
0.01	$122.9(\pm 13.2)$	5	$120.5(\pm 10.4)$	115%	
0.05	$123.8(\pm 8.1)$	10	$126.2(\pm 8.7)$	153%	
0.1	$126.2(\pm 8.7)$	15	$126.3(\pm 10.7)$	200%	
0.5	$122.1(\pm 10.2)$	20	$125.4(\pm 10.2)$	231%	
1.0	$113.0(\pm 7.1)$	30	127.1(±11.1)	253%	

Table 2: Experiment results of hyper-parameters α and K.

5.2 Online A/B test Results

We also evaluate the RL-MPCA approach for two weeks in the online environment. In online A/B tests, we compare our proposed RL-MPCA approach with several previous strategies deployed on Meituan advertising system. Table 3 lists the performance of several primary online metrics, including gross merchandise volume per mille (GPM, i.e., $GPM = \text{avg}(price_o) * 1000$, where $\text{avg}(price_o)$ is the average of $price_o$), cost per mille (CPM, i.e., $CPM = \text{avg}(fee_{ad}) * 1000$, where $\text{avg}(fee_{ad})$ is the average of fee_{ad}), click-through rate (CTR), and post-click conversion rate (CVR). RL-MPCA outperforms all other approaches, and ES-MPCA and DCAF take second and third place, respectively.

	cost	GPM	СРМ	CTR	CVR
Static	+0.00%	+0.00%	+0.00%	+0.00%	+0.00%
DCAF	-0.77%	+1.38%	0.01%	+0.26%	+0.53%
ES-MPCA	-0.3%	+2.25%	+0.12%	+0.83%	+0.89%
RL-MPCA	-1.5%	+3.68%	+0.90%	+1.09%	+2.86%

Table 3: The online A/B test results.

6 CONCLUSION AND FUTURE WORK

This paper proposes a Reinforcement Learning based Multi-Phase Computation Allocation approach, RL-MPCA, for recommender systems. RL-MPCA creatively formulates the computation resource (CR) allocation problem as a Weakly Coupled MDP problem and solves it with an RL-based approach. Besides, RL-MPCA designs a novel multi-scenario compatible Q-network adapting to various CR allocation scenarios, and calibrates Q-value by introducing multiple adaptive Lagrange multipliers (adaptive- λ) to avoid violating the global CR constraints when maximizing the business revenue. Both offline experiments and online A/B tests validate the effectiveness of our proposed RL-MPCA approach.

In future work, we plan to explore more general CR allocation approaches and more CR allocation application scenarios. Moreover, we plan to explore a new simulation scheme to capture the stochastic variation of response time and system load and then jointly model the response time constraint and the CR constraint to improve the system's availability.

REFERENCES

- Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained policy optimization. In *International conference on machine learning*. PMLR, 22–31
- [2] Daniel Adelman and Adam J Mersereau. 2008. Relaxations of weakly coupled stochastic dynamic programs. Operations Research 56, 3 (2008), 712–727.
- [3] Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. 2020. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 104–114.
- [4] Eitan Altman. 1999. Constrained Markov decision processes. Routledge.
- [5] Kiam Heong Ang, Gregory Chong, and Yun Li. 2005. PID control system analysis, design, and technology. IEEE transactions on control systems technology 13, 4 (2005) 550-576.
- [6] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. Journal of machine learning research 13, 2 (2012).
- [7] Craig Boutilier and Tyler Lu. 2016. Budget allocation using weakly coupled, constrained Markov decision processes. (2016).
- [8] Paris Carbone, Asterios Katsifodimos, Stephan Ewen, Volker Markl, Seif Haridi, and Kostas Tzoumas. 2015. Apache flink: Stream and batch processing in a single engine. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 36, 4 (2015).
- [9] Fanglin Chen, Xiao Liu, Bo Tang, Feiyu Xiong, Serim Hwang, and Guomian Zhuang. 2022. BCRLSP: An Offline Reinforcement Learning Framework for Sequential Targeted Promotion. arXiv preprint arXiv:2207.07790 (2022).
- [10] Yi Chen, Jing Dong, and Zhaoran Wang. 2021. A primal-dual approach to constrained markov decision processes. arXiv preprint arXiv:2101.10895 (2021).
- [11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems. 191–198.
- [12] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified conversational recommendation policy learning via graph-based reinforcement learning. In Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 1431–1441.
- [13] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Herve Jegou, and Armand Joulin. 2020. Training with quantization noise for extreme model compression. arXiv preprint arXiv:2004.07320 (2020).
- [14] Yufei Feng, Fuyu Lv, Weichen Shen, Menghan Wang, Fei Sun, Yu Zhu, and Keping Yang. 2019. Deep session interest network for click-through rate prediction. arXiv preprint arXiv:1905.06482 (2019).
- [15] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. 2020. D4rl: Datasets for deep data-driven reinforcement learning. arXiv preprint arXiv:2004.07219 (2020).
- [16] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking batch deep reinforcement learning algorithms. arXiv preprint arXiv:1910.01708 (2019).
- [17] Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International conference on machine learning*. PMLR, 2052–2062.
- [18] Rong Gao, Haifeng Xia, Jing Li, Donghua Liu, Shuai Chen, and Gang Chun. 2019. DRCGR: Deep reinforcement learning framework incorporating CNN and GAN-based for interactive recommendation. In 2019 IEEE International Conference on Data Mining (ICDM). IEEE, 1048–1053.
- [19] Yue He, Xiujun Chen, Di Wu, Junwei Pan, Qing Tan, Chuan Yu, Jian Xu, and Xiaoqiang Zhu. 2021. A Unified Solution to Constrained Bidding in Online Display Advertising. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining. 2993–3001.
- [20] Farah Tawfiq Abdul Hussien, Abdul Monem S Rahma, and Hala Bahjat Abdul Wahab. 2021. Recommendation systems for e-commerce systems an overview. In Journal of Physics: Conference Series, Vol. 1897. IOP Publishing, 012024.
- [21] Biye Jiang, Pengye Zhang, Rihan Chen, Xinchen Luo, Yin Yang, Guan Wang, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. 2020. DCAF: A Dynamic computation resource allocation Framework for Online Serving System. arXiv preprint arXiv:2006.09684 (2020).
- [22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning. Advances in Neural Information Processing Systems 33 (2020), 1179–1191.
- [23] Guogang Liao, Ze Wang, Xiaoxu Wu, Xiaowen Shi, Chuheng Zhang, Yongkang Wang, Xingxing Wang, and Dong Wang. 2022. Cross dqn: Cross deep q network

- for ads allocation in feed. In *Proceedings of the ACM Web Conference 2022.* 401-409
- [24] Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 1557–1565.
- [25] Yongshuai Liu, Jiaxin Ding, and Xin Liu. 2020. IPO: Interior-point policy optimization under constraints. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 34. 4940–4947.
- [26] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. 1998. Solving very large weakly coupled Markov decision processes. In AAAI/IAAI. 165–172.
- [27] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. nature 518, 7540 (2015), 529–533.
- [28] Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668 (2018).
- [29] Reuven Y Rubinstein and Dirk P Kroese. 2004. The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning. Vol. 133. Springer.
- [30] Pingzhong Tang, Xun Wang, Zihe Wang, Yadong Xu, and Xiwang Yang. 2020. Optimized Cost per Mille in Feeds Advertising. In Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems. 1359–1367.
- [31] Chen Tessler, Daniel J Mankowitz, and Shie Mannor. 2018. Reward constrained policy optimization. arXiv preprint arXiv:1805.11074 (2018).
- [32] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence, Vol. 30.
- [33] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In International conference on machine learning. PMLR, 1995–2003.
- [34] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. 2014. Natural evolution strategies. The Journal of Machine Learning Research 15, 1 (2014), 949–980.
- [35] Di Wu, Xiujun Chen, Xun Yang, Hao Wang, Qing Tan, Xiaoxun Zhang, Jian Xu, and Kun Gai. 2018. Budget constrained bidding by model-free reinforcement learning in display advertising. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management. 1443–1451.
- [36] Ruobing Xie, Shaoliang Zhang, Rui Wang, Feng Xia, and Leyu Lin. 2021. Hierarchical reinforcement learning for integrated recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 4521–4528.
- [37] Chaoqi Yang, Junwei Lu, Xiaofeng Gao, Haishan Liu, Qiong Chen, Gongshen Liu, and Guihai Chen. 2020. MoTiAC: Multi-objective actor-critics for real-time bidding. arXiv preprint arXiv:2002.07408 (2020).
- [38] Xun Yang, Yunli Wang, Cheng Chen, Qing Tan, Chuan Yu, Jian Xu, and Xiaoqiang Zhu. 2021. Computation Resource Allocation Solution in Recommender Systems. arXiv preprint arXiv:2103.02259 (2021).
- [39] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. 2021. Combo: Conservative offline model-based policy optimization. Advances in neural information processing systems 34 (2021), 28954–28967.
- [40] Yang Zhang, Bo Tang, Qingyu Yang, Dou An, Hongyin Tang, Chenyang Xi, Xueying Li, and Feiyu Xiong. 2021. BCORLE (λ): An Offline Reinforcement Learning and Evaluation Framework for Coupons Allocation in E-commerce Market. Advances in Neural Information Processing Systems 34 (2021), 20410– 20422
- [41] Xiangyu Zhao, Changsheng Gu, Haoshenglun Zhang, Xiwang Yang, Xiaobing Liu, Jiliang Tang, and Hui Liu. 2021. Dear: Deep reinforcement learning for online advertising impression in recommender systems. In Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35. 750–758.
- [42] Xiangyu Zhao, Xudong Zheng, Xiwang Yang, Xiaobing Liu, and Jiliang Tang. 2020. Jointly learning to recommend and advertise. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. 3319–3327
- [43] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining. 1059–1068.
- [44] Sijin Zhou, Xinyi Dai, Haokun Chen, Weinan Zhang, Kan Ren, Ruiming Tang, Xiuqiang He, and Yong Yu. 2020. Interactive recommender system via knowledge graph-enhanced reinforcement learning. In Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval. 179– 188

A SIMULATION SYSTEM

The offline simulation system contains two modules: the request simulation module and the revenue estimation module. For a given request, the request simulation module is responsible for interacting with an agent and generating interaction results. The revenue estimation module is a deep neural network model based on supervised learning, which evaluates the simulation results and predicts the user views, clicks, and purchases for each request. Although the offline simulation system requires a lot of time and computation resources, the prediction results of the revenue estimation module are relatively accurate because the request simulation module can generate detailed information about the requests. Finally, after calibrating the output of the revenue estimation model, our offline simulation system can achieve fairly confident revenue estimation results.

As Figure 6 shows, for each request *i*, the interaction of the simulation system and the agent involves multiple steps.

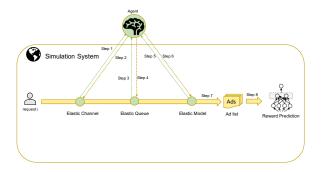


Figure 6: The structure of simulation system.

- Step 1. The simulation system constructs and feeds the initial state s_i¹ to the agent.
- **Step 2**. The agent takes Elastic Channel action a_1^i based on state s_1^i .
- **Step 3**. The simulation system retrieves the ads with action a_1^i , and feeds state s_2^i (including the retrieval ad list) to the agent
- Step 4. The agent takes Elastic Queue action aⁱ₂ based on state sⁱ₂.
- Step 5. The simulation system simulates the truncation operation with the truncation length corresponding to action aⁱ₂, and feeds state sⁱ₃ (including the truncated ad list) to the agent.
- **Step 6**. The agent takes Elastic Model action a_3^i based on state s_3^i .
- Step 7. The simulation system provides the prediction service for ads with the prediction model corresponding to action a_3^i , and outputs state s_4^i (including the truncated ad list and its prediction scores).
- Step 8. The simulation system takes state s₄ⁱ as input features, and predicts the final revenue (i.e., user views, clicks, and purchases) with a supervised learning based deep neural network model (see the architecture in Figure 7).

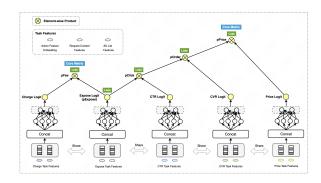


Figure 7: Architecture of revenue prediction model.

B PROOF

To slove the single-phase computation resource (CR) allocation problem in Section 4.1, we introduce a Lagrange multiplier λ_t , and construct the dual problem:

$$\min_{\lambda_t} \max_{a_t} \sum_{i=1}^{M} \sum_{a_t=1}^{N_t} x_{i,a_t} Value_{i,a_t} \\
- \lambda_t \left(\sum_{i=1}^{M} \sum_{a_t=1}^{N_t} x_{i,a_t} Cost_{i,a_t} - C_t \right)$$
(25)

s.t.
$$\sum_{a_t=1}^{N_t} x_{i,a_t} \le 1, \ \forall i,t$$
 (26)

$$x_{i,a_t} \in \{0,1\}, \ \forall i, a_t$$
 (27)

$$A_t \ge 0 \tag{28}$$

In phase t, for request i, there is one and only one action a_t can be taken. Then the dual problem above can be further transformed as:

$$\min_{\lambda_t} \sum_{i=1}^{M} \max_{a_t \in \{1, \dots, N_t\}} \{Value_{i, a_t} - \lambda_t Cost_{i, a_t}\} + \lambda_t(C_t)$$
 (29)

$$s.t \ \lambda_t \ge 0 \tag{30}$$

Thus, we have the global optimal solution to original problem, $x_{i,a_t^*} = 1$ when:

$$a_t^* = \underset{a_t}{\arg\max}(Value_{i,a_t} - \lambda_t Cost_{i,a_t})$$
 (31)

Note that a similar proof has been provided in [9], but the constraint definition of our optimization problem is different from it.

C COMPUTATION COST ESTIMATION

Essentially, CRs include computing resources, memory resources, network transmission resources, etc. In real industrial applications, computation cost estimation aims to find a metric that is easy to calculate and can be directly mapped to the amount of computation consumed. CRAS uses *queue length* as the computation cost metric, which is simple and feasible in Elastic Queue scenarios, and we have verified this in Meituan advertising system. However, *queue length* does not apply to Elastic Channel and Elastic Model scenarios. Specifically, in Elastic Channel, the primary metric affecting the CR consumption of the retrieval service is the number of requests entering the service. In Elastic Model, the primary metrics affecting

the resource consumption of the prediction service are the number of requests and the total number of ads entering the model. During the model training, we use the number of requests entering the retrieval channel and the number of requests entering the complex prediction model as the computation cost evaluation metrics to facilitate the evaluation of system computation. Because the Elastic Queue guarantees the number of ads entering the prediction model, it is reasonable to ignore the number of ads in Elastic Model when training the model.

In the offline experiments and online A/B tests, we also ensured that the number of ads entering the complex prediction model did not exceed the target value.

D HYPER-PARAMETERS

Table 4 lists the hyper-parameters of experiments.

Hyper-parameters	Value
Adaptive- λ update times K	10
Learning rate of adaptive- $\lambda \alpha$	0.1
Number of phases	3
Sizes of action spaces (N_c, N_q, N_m)	(2, 26, 2)
Number of heads in the network	64
Size of hidden layer in the network	[128, 64]
Optimizer	Adam
Learning rate	$3*10^{-4}$
Discount factor <i>γ</i>	0.99
Batch size	8192
Activation function	ReLU
BCQ threshold $ au$	0.3
Update frequency of target net N_{target}	100
Learning rate of λ in Ex-RCPO	$1*10^{-4}$
Temperature coefficient in Ex-CrossDQN	40

Table 4: The hyper-parameters of experiments.

E ES-MPCA

Same as RL-MPCA (see more details in Section 3.2), ES-MPCA also formulates the multi-phase CR allocation problem as a Weakly Coupled MDP problem. The difference is that ES-MPCA solves it with an evolutionary strategies based (ES-based) approach. To solve the Weakly Coupled MDP problem, we consider it as a black-box optimization problem, aiming to maximize the total business revenue under the CR constraints.

In this paper, we use Cross-Entropy Method (CEM) [29] to solve the black-box optimization problem. ES-MPCA designs the actions as:

$$channelQuota = f_c (\theta_c x_c)$$
 (32)

$$queueLen = f_q(\theta_q x_q) \tag{33}$$

$$modelQuota = f_m(\theta_m x_m)$$
 (34)

where *channelQuota*, *queueLen* and *modelQuota* are retrieval strategy number, truncation length and prediction model number, respectively. $(\theta_c, \theta_q, \theta_m)$ and (x_c, x_q, x_m) are parameters and features, respectively.

Algorithm 2 Offline Training of ES-MPCA (Based on CEM)

Input: Number of iteration I, the number of parameters $N_{all} = N_{channel} + N_{queue} + N_{model}$, the number of parameters sampled N_{sample} , the number of parameters retained N_{retain} .

- 1: Initialize mean $\pmb{\mu}^0=(\mu_1^0,\ldots,\mu_{N_{all}}^0)$ and variance $\pmb{\sigma}^0=(\sigma_1^0,\ldots,\sigma_{N_{all}}^0)$ of parameters.
- 2: **for** i = 1, ..., I **do**
- 3: Draw sample $\{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_{N_{sample}}\} \sim N(\boldsymbol{\mu}^{i-1}, \boldsymbol{\sigma}^{i-1})$
- 4: Evaluate $\{\theta_1, \dots, \theta_{N_{sample}}\}$ by simulation system (Reward Evaluation)
- 5: Sort $\{\theta_1, \dots, \theta_{N_{sample}}\}$ by the reward $reward = \sum Value(\theta) \sum_t \lambda_t \min\{C_t \sum Cost_t(\theta), 0\})$
- 6: Take top- N_{retain} parameters $\{\theta_1, \dots, \theta_{N_{retain}}\}$, then calculate their mean μ^i and variance σ^i
- 7: end for

Output: The best parameter $\theta^* = (\theta_1^*, \dots, \theta_{N_{all}}^*)$

Algorithm 2 describes the training process of CEM-based ES-MPCA. By imposing an extremely large penalty on the parameters that violate the constraint (λ_t is generally an extremely large value, e.g., for each phase t, $\lambda_t = 10^8$ in our experiments), ES-MPCA always guarantees that the final output optimal parameters θ^* are those that satisfy the CR constraints.

Experiment results show that the optimal parameters θ^* outputted by ES-MPCA always exactly satisfy the CR constraints (i.e., for each phase t, $\sum Cost_t(\theta^*) = C_t$ holds), which is consistent with the assumptions and conclusions in Section 4.1.

F BEHAVIORAL POLICIES

In this section, we provide a detailed introduction to behavioral policies. Random exploratory policies randomly make decisions in each phase to explore the revenues under different actions, including randomly selecting retrieval channels, truncation lengths, and prediction models. Superior policies include ES-based policies and RL-based policies. We train them on a random dataset collected by random exploratory policies. More details of ES-based policies are provided in Appendix E.

G ONLINE SERVING

After model training (Algorithm 1) and λ -correction (see more details in Section 4.1.2), we obtain the trained network Q_{θ} and trained constraint parameter $\lambda = (\lambda_1, \dots, \lambda_T)$. Algorithm 3 shows the process of online serving for a given request.

Algorithm 3 Online Serving of RL-MPCA

Input: Trained Network Q_{θ} , trained constraint parameter $\lambda =$

- $(\lambda_1,\ldots,\lambda_T)$
- Initialize state s₀
- 2: **for** t = 1, ..., T **do**
- Take action $a_t^* = \arg\max_{a_t} (Q_{\theta}(s_t, a_t) \lambda_t Cost(s_t, a_t))$
- 4: Execute allocation following a_t^*
- 5: Observe the next state from system
- 6: end for