Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Special Track on AI for Good

GreenFlow: A Computation Allocation Framework for Building Environmentally Sound Recommendation System

Xingyu Lu[1] , **Zhining Liu**[1] , **Yanchu Guan**[1] , **Hongxuan Zhang**[2] , **Chenyi Zhuang**[1] , **Wenqi Ma**[1] , **Yize Tan**[1] , **Jinjie Gu**[1] and **Guannan Zhang**[1]
[1]Ant Group
[2]Nanjing University
chenyi.zcy@antgroup.com

Abstract

Given the enormous number of users and items, industrial cascade recommendation systems (RS) are continuously expanded in size and complexity to deliver relevant items, such as news, services, and commodities, to the appropriate users. In a real-world scenario with hundreds of thousands requests per second, significant computation is required to infer personalized results for each request, resulting in a massive energy consumption and carbon emission that raises concern.

This paper proposes GreenFlow, a practical computation allocation framework for RS, that considers both accuracy and carbon emission during inference. For each stage (e.g., recall, pre-ranking, ranking, etc.) of a cascade RS, when a user triggers a request, we define two actions that determine the computation: (1) the trained instances of models with different computational complexity; and (2) the number of items to be inferred in the stage. We refer to the combinations of actions in all stages as *action chains*. A reward score is estimated for each action chain, followed by dynamic primal-dual optimization considering both the reward and computation budget. Extensive experiments verify the effectiveness of the framework, reducing computation consumption by 41% in an industrial mobile application while maintaining commercial revenue. Moreover, the proposed framework saves approximately 5000kWh of electricity and reduces 3 tons of carbon emissions per day.

1 Introduction

With the rapid development of AI, especially after the 2012 breakthrough, i.e., the deep learning revolution, computation has grown substantially. The computations required for deep learning research have been doubling every few months, resulting in an estimated 300,000x increase from 2012 to 2018[1]. This significant increase in computation has led to a huge energy consumption and carbon emissions. Facing the explosive growth of the demand for computation, Green AI [Schwartz *et al.*, 2019; Xu *et al.*, 2021; Yigitcanlar *et al.*, 2021], which aims to maximize energy efficiency and minimize environmental impact, has attracted a lot of research interest.

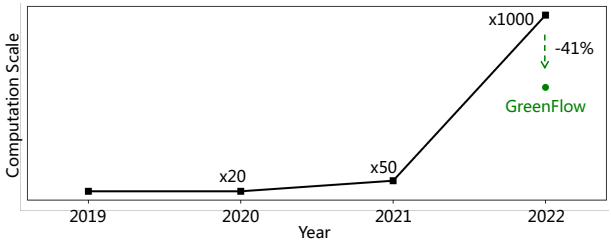

Figure 1: The growth trend of computation consumed by RSs on an industrial mobile application. With GreenFlow, the growth trend has been substantially slowed down.

RS serves as one of the most widely used AI-based application in various domains, such as content recommenders for multimedia services [Covington *et al.*, 2016], product recommenders for e-commerce [Zhou *et al.*, 2017], or tweet recommenders for social media platforms [Chen *et al.*, 2012]. Due to large-scale and wide-range of RS, it contributes to a large percentage of AI computation. As shown in Figure 1, we show the growth trend of computation in recent years for RSs of an industrial mobile application. The rapidly growing curve poses a great challenge to the computation overhead. However, we found that the difficulty of identifying users' preferences is significantly different, which suggests that the demand for computation of each user is different and an equal allocation strategy would easily lead to an inefficient use of computation. In addition, industrial RSs are often equipped with multiple cascade AI models of varying levels of computation to satisfy the need for low latency (typically within a few hundred milliseconds) and limited computation budget. However, the computation budget allocated to each stage is usually fixed, which also results in inefficiencies.

To maximize the utility of computation consumed by the RS, two critical problems arise in computation allocation: (1) how to estimate the reward curve for each user given different levels of computation; (2) how to allocate the limited computation to arriving requests effectively based on personalized demand. To address these challenges, we propose a computation allocation framework for the industrial cascade RS as

[1]https://openai.com/research/ai-and-compute/

shown in Figure 2. Specifically, we first define the two actions that determine the computation as (1) the trained instances of models with different computational complexity; and (2) the number of items to be inferred in each stage. Moreover, we design action chains as the allocation unit, which denote the combinations of model instances and the number of items of all the stages. Next, we estimate a reward score and computation cost for each action chain, followed by a dynamic primal-dual optimization considering both the reward and computation budget. This approach results in an optimal action chain for computation allocation.

Scientific Contribution. In summary, our contributions mainly include the following three aspects:

- **Problem.** We formally define the problem of computation allocation with the goal to maximize its revenue given a limited computation budget in an industrial RS.
- **Method.** We propose a generic computation allocation framework, which maximizes the revenue of computation through (1) building an personalized reward estimation model of action chains, and (2) conducting dynamic primal-dual optimization for online allocation.
- **Evaluation.** We perform extensive experiments showing that GreenFlow consistently achieves improvements. By deploying GreenFlow on several industrial RSs, we are able to save approximately 5000kWh of electricity and reduce carbon emissions by 3 tons per day. In addition, we present an evaluation methodology named PFEC, which thoroughly evaluates the effectiveness of the computation allocation method from four aspects: **P**erformance, **F**LOPs, **E**nergy, and **C**arbon emission.

Impact to the SDGs. We believe that such a computation allocation framework can greatly improve the resource-use efficiency (SDG 9), which can make the RS environmentally sound and also contributes a lot to the combat climate change (SDG 13) because it can greatly reduce carbon emission without harming the commercial revenue.

2 Related Work

2.1 Green AI

Green AI is an emerging research field that aims to develop strategies to reduce energy consumption and mitigate the climate impact of AI systems to maximize energy efficiency. We summarize its research directions as follows:

- Data center: This direction mainly focuses on developing more energy-efficient algorithms and hardware, optimizing data center cooling and power usage [Khosravi *et al.*, 2017], and using renewable energy sources to reduce the power usage effectiveness (PUE).
- Computing cluster and engine: It focuses on improving the conversion efficiency of energy to computation via optimizing the scheduler [Tirmazi *et al.*, 2020], parallel computing [Shukur *et al.*, 2020], and other techniques.
- AI/ML application: Through optimizing the all stages of deep learning [Xu *et al.*, 2021] towards lower computation costs, the conversion efficiency of computation to revenue is improved. Among existing directions, architecture design [Kitaev *et al.*, 2020; Howard *et al.*, 2017], training [Liu *et al.*, 2021; Huang *et al.*, 2018] and inference [Molchanov *et al.*, 2016; Han *et al.*, 2016] are major research directions.

In this paper, we mainly focus on RS from the perspective of AI/ML application, and achieve the goal of Green AI via allocating the computation in an efficient way.

2.2 Computation Allocation in RS

Multi-stage cascade ranking systems is a common solution in the industry [Wang *et al.*, 2011; Qin *et al.*, 2022] to achieve ranking of a great amount of items with low latency. Its main idea is to deploy simple rankers in the early stages for pre-ranking on the scale of the entire candidate set, and use complex rankers for a much smaller set of items for better accuracy, so that we can achieve a well trade-off between efficiency and effectiveness. To our knowledge, [Jiang *et al.*, 2020] is the first paper to propose the allocation of computation in RS at the granularity of online requests, where the size of candidate items is adjusted for each user. However, this approach only focuses on one specific stage of RS and may not perform well on a cascade architecture. [Yang *et al.*, 2021] extends the method to cascade RS, assuming that the revenue of each stage is independent of the others and can be multiplied to obtain the total revenue of the cascade RS.

However, these studies do not consider the joint effects across stages and different models, which results in a failure to model the personalized reward model at a fine-grained level and allocate the computation effectively.

3 Framework

3.1 Overview

In a multi-stage cascade RS, we introduce the GreenFlow framework to maximize the efficiency of the whole RS under the limited computation budget. Figure 2 illustrates the whole process of a RS after introducing the GreenFlow. In order to take computation budget into consideration, we introduce the new concept called *action chain*. Since the computation cost of a certain ranking stage is determined by the computational complexity of ranking models and the number of items to be inferred, so that we adopt an action chain to assemble them for all stages. Then, GreenFlow aims to allocate an optimal action chain for each incoming request prior to the RS. In details, the framework is consist of three steps.

In the first step, the *action chain generator* constructs candidate action chains through cartesian product between models and item scales.

Then, in the second step, we develop a computation measure module and a reward model to estimate the computation costs and rewards of each candidate action chain, respectively. For a comprehensive computation cost estimation, several metrics, such as FLOPs, carbon emissions, and CPU usage, are calculated using static tools (e.g., experiment-impact-tracker [2], carbontracker[3], etc.). For the reward estimation task, in the context of a specific request, our model uses

[2]https://github.com/Breakend/experiment-impact-tracker
[3]https://github.com/lfwa/carbontracker

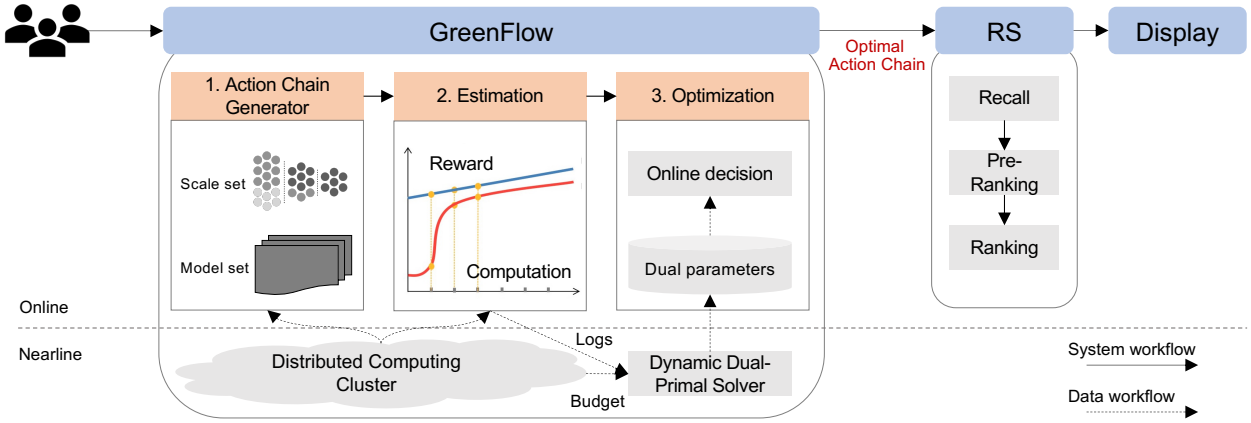

Figure 2: System overview of the proposed GreenFlow.

the actual revenues (e.g., click rate, conversion rate, etc.) as labels for training. In general, an action chain with a higher computation cost tends to generate better revenue. However, it should be noted that the uplift of the reward curve varies with (1) different requests and (2) the corresponding stages as the computation cost increases.

In the third step, we propose a hybrid online-nearline architecture to obtain the optimal action chain for allocation under the global computation budget. In the nearline module, a streaming job collects the candidate action chains' reward scores and computation costs of each request. Then, a dynamic primal-dual solver is proposed to do a matching task using the collected samples and the current budget periodically. Finally, the solved dual parameters are stored into an online storage. The entire procedure of updating the dual parameters takes minutes or seconds, depending on the response time requirements of the RS. In the online module, the framework returns an optimal action chain for each arriving request to the RS.

However, it should be noted that introducing GreenFlow into RS will increase the computational overhead. We will further discuss this trade-off in the experiment section.

3.2 Evaluation Methodology

To quantitatively measure the effectiveness of the computation allocation framework in increasing computation utilization and combating climate change, we propose an evaluation methodology named **PFEC**, which thoroughly evaluates the effectiveness of the framework from the following four aspects, i.e., **P**erformance, **F**LOPs, **E**nergy, and **C**arbon emission.

Performance is typically used to measure the revenue generated by RS, with common metrics including clicks, conversions, and other similar measures. FLOPs, which measures how many operations are required to run a single instance of a given neural network, can quantify how much the computation RS consumes.

As for energy and carbon emission, following the method proposed in [Lacoste *et al.*, 2019], we first calculate the energy consumption (EC) mainly taking the RAM, CPU and GPU into consideration:

$$\mathrm{EC} = \mathrm{PUE} \cdot (p_{\mathrm{ram}} e_{\mathrm{ram}} + p_{\mathrm{cpu}} e_{\mathrm{cpu}} + p_{\mathrm{gpu}} e_{\mathrm{gpu}}), \quad (1)$$

where PUE is a constant determined by the energy efficiency of a data center (the worldwide average is 1.67[4]), $p_{(\cdot)}$ and $e_{(\cdot)}$ denote the rated power of certain device and device usage, respectively. With the EC, we can further calculate the carbon emission (CE) via:

$$\mathrm{CE} = \mathrm{EC} \cdot \mathrm{CI}, \quad (2)$$

where CI is the carbon intensity that varies with different countries, and it is set to $615 gCO_2 ekWh^{-1}$ in this paper [5].

With PFEC, we can compare different allocation frameworks on resource utilization and impact on the environment.

4 Methodology

4.1 Notations and Definitions

In this section, we will introduce the formal definition of the action chain and the computation allocation problem for the whole RS. We assume that x or X denotes a scalar; $\vec{x}$ denotes a vector; $\mathbf{X}$ denotes a matrix; and $\mathcal{X}$ denotes a set.

Let $\mathcal{I}_t := \{1, 2, ..., I_t\}$ be a set of arriving requests in time t, K be the number of stages in the cascade RS and C be the computation budget of the whole RS. In each stage k, the ranking model m_k and ranking item scale n_k are selected from the *Model Pool* $\mathcal{M}$ and the *Item Scale* set $\mathcal{N}$ as shown in Figure 2, respectively. A tuple of m_k and n_k constitute the key parameters $s_k = \{m_k, n_k\}$ of the k-th stage. Then, we define the action chain of a whole cascade ranking RS as $a = (s_1, s_2, ..., s_K)$ and let $\mathcal{A}$ be the action chain set constructed by the *action chain generator* where $|\mathcal{A}| = J$. For each action chain a_j, let c_j be the estimated computation cost. For each request i, let $R_{ij} := R(a_j, f_i)$ be the estimated reward of a_j with the context feature f_i which is related to the arriving request $i \in \mathcal{I}_t$. Then we define the allocation strategy $\mathbf{x}_i := (x_{i,1}, x_{i,2}, ..., x_{i,J}) \in \{0, 1\}^J$ that x_{ij} represents the decision

[4] https://journal.uptimeinstitute.com/is-pue-actually-going-up/
[5] https://app.electricitymaps.com/zone

variable of allocating action chain j to request i. We construct the computation allocation problem as following:

$$\max_{\mathbf{x}_i} \sum_{i\in\mathcal{I}_t, 1\le j\le J} R_{ij}x_{ij}, \tag{3a}$$

$$s.t. \quad \sum_{j\le J} x_{ij} = 1, \forall i \in \mathcal{I}_t, \tag{3b}$$

$$\sum_{i\in\mathcal{I}_t, 1\le j\le J} c_j x_{ij} \le C, \tag{3c}$$

$$x_{ij} \in \{0,1\}, \forall i \in \mathcal{I}_t, 1 \le j \le J \tag{3d}$$

In the next two subsections, we first introduce how to accurately estimate the reward function R_{ij}, and then propose a method for quickly solving the Problem (3).

4.2 Personalized Reward Model

In general, increasing computation is likely to result in higher revenue. However, the same action chain may generate different revenues for different requests. Thus, for each request i with context features f_i and action chain a_j, it is necessary to design a personalized model to accurately estimate the reward R_{ij}. In this regard, we encounter the following three challenges:

- How to model the dependence of cascade stages in RS?
- How to deal with the data sparsity problem?
- How to ensure the monotonicity between computation and reward?

Accordingly, we introduce the key three mechanisms of the model in the remainder of this section.

Recursive Multi-Stage Design. The structure of our proposed model is designed in a recursive style to match the cascade design of RS. As illustrated in Figure 3, the chunk of neural network is the detailed implementation of the recursive function g_k. Given the output embedding of the previous stage $\vec{h}_{k-1}$, the action $\{m_k, n_k\}$ from k-th stage, and the context feature f_i, g_k outputs the reward uplift Δr_k and $\vec{h}_k$ for the next stage. Hence, R_{ij} is calculated by:

$$R_{ij} = \sum_{k=1}^{K} \Delta r_k, \tag{4a}$$

$$(\Delta r_k, \vec{h}_k) = g_k(\vec{h}_{k-1}, \vec{f}_i, \vec{m}_k, \vec{n}_k), \tag{4b}$$

where K is the number of stages; $\vec{f}_i, \vec{m}_k, \vec{n}_k$ are the encoded embeddings of f_i, m_k, n_k, respectively. Meanwhile, each g_k should deal with data sparsity problem and obey the monotonicity constraint.

Multi-Basis Functions

In an industrial RS, collecting feedback for all action chains on the same user can be expensive, which presents a challenge in learning the function g_k using a network with high degrees of freedom. So we propose a novel structure of consisting multi-basis functions in the yellow chunks of Figure 3. The reward uplift Δr_k is given by:

$$\Delta r_k = \sum_{\phi_p\in\mathcal{B}} w_p \phi_p(\mathbf{1}_Q^\top v_p), \tag{5}$$

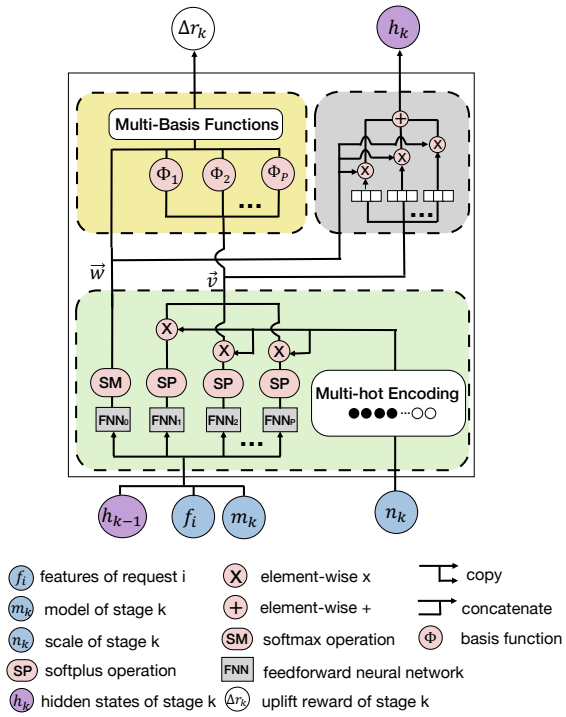

Figure 3: The structure of the recursive function g_k.

where $\mathcal{B}$ is a set of basis functions and $|\mathcal{B}| = P$, $\mathbf{1}_Q$ is a vector of all ones; $w_p \in \vec{w} = (w_1, w_2, ..., w_P)$ and $v_p \in \vec{v} = (v_1, v_2, ..., v_P)$ are two vectors calculated by the feedforward neural networks FNN$(\cdot)$ in the green chunk of Figure 3. $\vec{w}$ and $\vec{v}$ are defined as:

$$\begin{aligned} v_p &= softplus(\text{FNN}_p(\vec{h}_{k-1}, \vec{f}_i, \vec{m}_k)) * \vec{n}_k, \forall p \le P, \\ \vec{w} &= softmax(\text{FNN}_0(\vec{h}_{k-1}, \vec{f}_i, \vec{m}_k)), \end{aligned} \tag{6}$$

where Q is the dimension of the embedding $\vec{n}_k$ and $*$ is the element-wise multiply. In the current implementation, we define the multi-basis functions set as:

$$\mathcal{B} = \{tanh(x), ln(x), \frac{x}{\sqrt{1+x^2}}, sigmoid(x), x\}. \tag{7}$$

This design has the following advantages:

- The use of multi-basis functions can better capture the reward curves of users with varying levels of activity.
- The design of different basis functions can make the reward model obtain different mathematical properties. For instance, if the second derivative of the basis functions we design is non-positive, then the marginal revenue of the reward curves will be non-increasing. That is, as FLOPs become larger, the slope of the reward curve will become smaller.

In the experiment section, we will conduct ablation experiments to demonstrate the advantages of the aforementioned design and the specific effects it achieves.

Monotonic Constraint. To obey the monotonicity between computation (especially the item scale n_k) and reward Δr_k,

we use a multi-hot embedding $\vec{n}_k$ for the item scale n_k in each stage k. In the multi-hot encoding module, the item scale set $\mathcal{N}_k$ is divided into Q groups and each n_k in the same group shares the same embedding. The larger item scale n_k attains more 1s in $\vec{n}_k$ during the encoding process, and the larger $\vec{v}$ computed by Equation (6) will be. Since all the basis functions are monotonically increasing and $\vec{w}$ are non-negative, the monotonically increasing relationship between the item scale n_k and reward Δr_k is preserved.

4.3 Dynamic Primal-Dual Optimization

With known requests $\mathcal{I}_t$, estimated revenues R_{ij} and resource cost c_j in time t, one can regard the problem defined in Equation (3) as a bipartite matching problem [Aggarwal *et al.*, 2011] following a primal-dual framework. By optimizing this problem, the dual variables can be solved directly. Specifically, the Lagrangian function of Equation (3) is given by

$$L_t(\lambda, \mathbf{X}) = \sum_{i \in \mathcal{I}_t, j \leq J} R_{ij} x_{ij} + \lambda (C - \sum_{i \in \mathcal{I}, j \leq J} c_j x_{ij}), \quad (8)$$

and the dual form of Equation (3) leads to:

$$\min_{\lambda \geq 0} \max_{X} L_t(\lambda, \mathbf{X}), \quad (9)$$

where λ is the dual variable. In the literature of matching problem, λ is called *dual price* which represents the increased revenues of per resource cost given the budget. With the help of strong duality and K.K.T conditions, we can derive the optimal decision $\vec{x}_i$ from solved dual price λ^* by:

$$x_{ij} = \begin{cases} 1, & \text{if } j = \arg\max_j \{R_{ij} - c_j \lambda^*\} \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

We propose a dual descent algorithm to solve the optimal dual price λ_t in time t and apply the λ_t to make the online decision $\vec{x}_i$ for requests $i \in \mathcal{I}_{t+1}$ periodically. The pseudo-code of the whole procedure is summarized in Algorithm 1. In each time t, a gradient descent approach is adopted to solve λ_{t+1} by step 6∼8 with global convergence guarantee [Li *et al.*, 2020]. Moreover, [Agrawal *et al.*, 2014] shows that the online decision in time $t+1$ using the previous λ_t is near-optimal when the requests adopt a stochastic arriving model and the requests $\mathcal{I}_t$ is enough for solving λ_t. By adopting a seconds or minutes period of Algorithm 1 in an industrial RS, our algorithm is effective that the distribution of arriving users is i.i.d. in a short time and the number of requests per second are generally thousands.

Algorithm 1 Dynamic Primal-Dual Algorithm

1: **Input:** computation budget C, action chain set $\mathcal{A}$, computation cost c_j ($\forall a_j \in \mathcal{A}$), max iterations L, step size η and initialized dual price λ_0.
2: **for** $t = 1, 2, \ldots$ **do**
3: Collect the reward R_{ij} for each request $i \in \mathcal{I}_t$.
4: Let $\lambda_t^0 = \lambda_{t-1}$.
5: **for** $l = 0, 1, 2, ..., L$ **do**
6: Update $\vec{x}_i, \forall i \in \mathcal{I}_t$ given λ_t^l by Equation (10).
7: $\nabla L = C - \sum_{i \in \mathcal{I}_t, 1 \leq j \leq J} c_j x_{ij}$.
8: $\lambda_t^{l+1} = \lambda_t^l - \eta \nabla L$.
9: **end for**
10: Let $\lambda_t = \lambda_t^L$, and solve $\vec{x}_i$ for $i \in \mathcal{I}_{t+1}$ with λ_t by Equation (10).
11: **end for**

5 Experiments

In this section, we will evaluate the proposed framework through various offline and online experiments to demonstrate its effectiveness. Before introducing the experimental results, we will first describe the experiment settings, including the datasets used, details of training, comparison methods, and evaluation metrics.

5.1 Experimental Setup

Dataset. For reproducibility, we use one public dataset named Ali-CCP[6] for offline evaluation. It is a public click and conversion prediction dataset collected from traffic logs of Taobao's RS [Ma *et al.*, 2018], and consists of 85 million samples. We randomly sample 50% of users in the dataset for training models in the cascade RS, 25% for building the validation dataset, 22.5% for training sample generation of the reward model, and the rest 2.5% for the final evaluation, which includes 9016 users.

Implementation of Action Chain. In the experiments, we consider a three-stage (i.e., recall, pre-ranking and ranking) cascade RS with four trained instances of models for allocation, of which detailed statistics are shown in Table 1. To simplify the comparison, only one model for each of the first two stages, which are DSSM [Huang *et al.*, 2013] and YoutubeDNN [Covington *et al.*, 2016] (abbreviated as YDNN for brevity), respectively. DSSM is used for ranking the entire candidate items (its size is n_1), and YDNN performs pre-ranking on the top-n_2 scored candidate items ranked by DSSM. As for the ranking stage, the most two powerful models, DIN [Zhou *et al.*, 2017] and DIEN [Zhou *et al.*, 2018], are available for selection to generate top-n_3 scored candidate items feed by the previous stage. Finally, we choose top scored items to expose to users. Therefore, an action chain is denoted as $a = (\{\text{YDNN}, n_2\}, \{\text{DIN or DIEN}, n_3\})$, where $n_2 \in \mathcal{N}_2 = [800, 900, 1000, ..., 1500]$ and $n_3 \in \mathcal{N}_3 = [60, 80, 100, ..., 200]$ in this paper ($\{\text{DSSM}, n_1\}$ is omitted due to its fixed computation cost).

Evaluation Metrics. To compare different methods thoroughly, we construct various computation budgets C by simulating different action chains. Given a specific budget constraint C, we focus on the following offline metrics:

$$revenue@e = \sum_{i \in \mathcal{I}_t, j \in \mathcal{P}} \hat{R}_{ij}, \quad (11)$$

where $\mathcal{P}$ denotes the top-e scored candidate items in the ranking stage generated by an action chain (we set $e = 20$ in this experiment), and the reward $\hat{R}_{ij}$ of the user i is set to the number of the clicked items. Since it is difficult to simulate the energy consumption in an offline way, we mainly consider the first two indicators of PFEC, i.e., $revenue@e$ and

[6] https://tianchi.aliyun.com/dataset/408

Stage	Model	FLOPs	AUC
Recall	DSSM	13K	0.525
Pre-ranking	YDNN	123K	0.581
Ranking	DIN	7020K	0.639
Ranking	DIEN	7098K	0.641

Table 1: Trained instances of models for allocation of each stage.

C, in the offline experiments, and a full PFEC evaluation is conducted on the online experiments.

Training Sample Generation of Reward Model. To train the personalized reward model, we simulate different action chains for each user and calculate the reward of each action to build the training dataset. Specifically, the click-through rate on $\mathcal{P}$ is used to represent the reward (i.e., label) of the action chain $a = (\{\text{YDNN}, n_2\}, \{\text{DIN or DIEN}, n_3\})$ for supervising the training of the reward model.

Comparison Methods. Two methods are compared in the experiments:

- **EQUAL**: Computation is allocated to each user equally based on a given fixed action chain.
- **CRAS** [Yang *et al.*, 2021]: A method that decomposes the computation allocation into independent subproblems on each stage of RS.

5.2 Offline Experiments

In this part, the offline experiments are designed to answer the following questions:

- **Q1**: Does our proposed GreenFlow outperform other approaches on the task of computation allocation?
- **Q2**: Does multi-stage modeling in reward estimation outperform single-stage modeling?
- **Q3**: Is it necessary to provide multi-models in one stage for allocation rather than single-model?
- **Q4**: How does each variant of the proposed reward model contribute to the improvement?

Q1: Effectiveness of GreenFlow

We report the results given different computation budgets in Figure 4. Due to the limitation that EQUAL and CRAS only consider single model for allocation in the same stage, we design two variants of EQUAL and CRAS, respectively, i.e. EQUAL-DIN, EQUAL-DIEN, CRAS-DIN and CRAS-DIEN, for comparison as there are two available models in the ranking stage. From Figure 4, we can draw the following conclusions:

- With the decreasing budget, the number of clicks is accordingly decreasing, which implies that a well-designed cascaded RS is necessary due to its improvement over the revenue of computation.
- Our proposed method outperforms all baseline methods with a large margin, which verifies the superiority of dynamic action chain and multi-stage reward modeling.

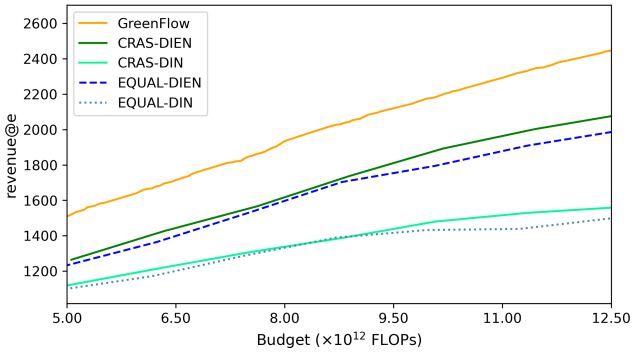

Figure 4: Results of different methods with different budgets.

Setup			Methods	
Strategy	Actions	$\times 10^{12}$FLOPs	CRAS	Ours
Single-Stage	$\{ m_3$=DIEN, $n_3 \in \mathcal{N}_3\}$	8.68	1634	**1665**
		9.96	1740	**1784**
		11.24	1833	**1873**
	$\{m_2$=YDNN, $n_2 \in \mathcal{N}_2\}$	7.33	1223	**1248**
		7.55	**1310**	1301
		7.88	1516	**1524**
Multi-Stage	($\{m_2$=YDNN, $n_2 \in \mathcal{N}_2\}$, $\{ m_3$=DIEN or DIN, $n_3 \in \mathcal{N}_3\}$)	8.68	1624	**2023**
		9.96	1765	**2174**
		11.24	1845	**2321**
		7.33	1225	**1820**
		7.55	1309	**1851**
		7.88	1522	**1903**

Table 2: Results with different computation budgets.

Q2: Single-Stage vs. Multi-Stage

To demonstrate the effectiveness of multi-stage modeling, we compare GreenFlow and CRAS with one action fixed, i.e., our reward model degrades into single-stage. From Table 2, we observe that CRAS and GreenFlow achieve comparable performance under one action fixed, but GreenFlow outperforms CRAS by a large margin when conducting multi-stage modeling, which verifies the superiority of multi-stage modeling.

Q3: Single-Model vs. Multi-Model

As presented in Table 1, both DIN and DIEN models have similar FLOPs and perform comparably in terms of AUC. This naturally raises the question of whether introducing different models in the same stage is necessary. We argue that this necessity based on the fact that no single model can outperform all others on all users. To evaluate this, we divided the user set into three groups: those better suited to DIN, those better suited to DIEN, and those equally suited to both. The distribution of these groups is approximately 1:3:6, respectively. Then, we conducted three tests on GreenFlow using only DIN, only DIEN, and both of them. The results are shown in Table 3, and it is evident that using multiple models yields gains ranging from 1% to 6% on $revenue@e$ compared to using any single model.

Budget C	Methods		
$\times 10^{12}$FLOPs	Only DIN	Only DIEN	Both
6.2	1345	1572	**1667**
7.4	1467	1814	**1834**
7.5	1471	1823	**1848**
8.7	1581	2007	**2026**
10.1	1686	2079	**2181**

Table 3: Results of GreenFlow with different models available for selection in the ranking stage.

Variants of Reward Model		Metrics	
Recursive Mechanism	Multi-Basis Functions	Field-RCE	$revenue@e$
✓	✓	**0.137**	**2174**
✓	✗	0.148	2082
✗	✓	0.150	2065
✗	✗	0.177	1875

Table 4: Results of variants of reward models under 9.96×10^{12} FLOPs.

Metrics		A	B	C
PFEC	P: Clicks	+2.1%	-0.2%	+0.3%
	FLOPs	-61%	-20%	-15%
	Energy	-4869kWh	-168kWh	-124kWh
	CO_2	-2995kg	-103kg	-76kg
Extra Cost	Latency	-20ms	+10ms	+5ms
	FLOPs	+3%	+8%	+8%

Table 5: Online A/B testing results (per day) of three RSs.

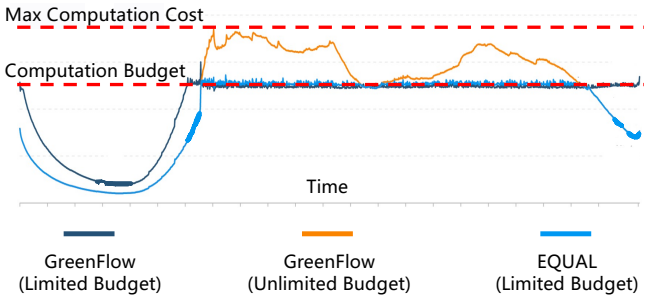

Figure 5: Computation cost of three different allocation strategies.

Q4: Variants of Reward Modeling

To verify the effectiveness of the proposed reward model, we test different combinations of the two mechanisms (i.e., the recursive mechanism and the introduction of basis functions) in the reward model. Since well-calibrated models are important for optimizing Equation (3), we additionally introduce another metric for evaluating the reward model, which is field-level relative calibration error [Pan *et al.*, 2019]:

$$\text{Field-RCE} = \frac{1}{|\mathcal{D}|} \sum_{f \in \mathcal{F}} \frac{|\sum_{i \in \mathcal{D}^f} (y_i - \hat{y}_i)|}{\frac{1}{|\mathcal{D}^f|} \sum_{i \in \mathcal{D}^f} y_i}, \tag{12}$$

where $\mathcal{D}$ is the test dataset, $\mathcal{F}$ is the specified feature field, $\mathcal{D}^f$ is the set whose feature value is f, y^i is the label and $\hat{y}_i$ is the predicted value. This metric measures the deviation between the predicted value of reward curve and posterior probability.

Table 4 shows the results, and we conclude that both recursive mechanism and multi-basis functions contribute to improvement of Field-RCE and $revenue@e$.

5.3 Online Experiments

To further demonstrate the effectiveness of GreenFlow, we conduct online A/B testing on three RSs with different levels of computational overhead, and the results are reported in Table 5. We observe significant decrease in FLOPs (i.e., greatly reduce the demand for model inferring servers), a slight improvement in the accuracy of RS, and negligible latency due to the introduction of GreenFlow. Specifically, we report the extra computation cost and latency brought by GreenFlow in Table 5, which can be ignored compared to the RS. Note that for RS A, the latency even decreases because the large reduction in FLOPs leads to fewer request packets, greatly reducing the time of model inference.

Due to a large number of requests, it is also necessary to ensure that the constraint of the computation budget can be precisely satisfied in an industrial environment. When it fails, RS cannot meet all requests for recommendation and has to resort to computation downgrade, resulting in harm to revenue. In Figure 5, we show computation cost of three different allocation strategies, and conclude that GreenFlow can handle traffic spikes well under the constraint of the computation budget and save a lot of computation in the same time.

To date, GreenFlow has been deployed in several RSs of an industrial mobile application with about one billion users for one year. This deployment has resulted in a significant reduction of 1,000 tons of carbon emission and a saving of approximately 2,000,000 kWh of electricity. Moreover, GreenFlow has achieved a remarkable balance among revenue, computation, carbon emission, and energy consumption.

6 Conclusion

In this paper, we propose a computation allocation framework for industrial RS to enhance resource utilization efficiency. The framework first estimates a reward score for each action chain, and dynamic primal-dual optimization is conducted to search for the optimal action chain while considering both the reward score and computation budget. Through offline experiments and online A/B testing, we demonstrate the effectiveness of GreenFlow in maximizing the revenue of a RS given a particular computation budget. Additionally, GreenFlow has been deployed in the production environment of a mobile application, resulting in significant economic returns and a substantial reduction in carbon emissions. In the future work, we will focus on improving the reward model since it serves as the core component that determines the computation allocation.

Contribution Statement

Xingyu Lu and Zhining Liu contributed equally to this research. Hongxuan Zhang completed this work at Ant Group.

References

[Aggarwal *et al.*, 2011] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the twenty-second annual ACM-SIAM symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.

[Agrawal *et al.*, 2014] Shipra Agrawal, Zizhuo Wang, and Yinyu Ye. A dynamic near-optimal algorithm for online linear programming. *Operations Research*, 62(4):876–890, 2014.

[Chen *et al.*, 2012] Kailong Chen, Tianqi Chen, Guoqing Zheng, Ou Jin, Enpeng Yao, and Yong Yu. Collaborative personalized tweet recommendation. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2012.

[Covington *et al.*, 2016] Paul Covington, Jay K. Adams, and Emre Sargin. Deep neural networks for youtube recommendations. *Proceedings of the 10th ACM Conference on Recommender Systems*, 2016.

[Han *et al.*, 2016] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark Horowitz, and William J. Dally. Eie: Efficient inference engine on compressed deep neural network. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 243–254, 2016.

[Howard *et al.*, 2017] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[Huang *et al.*, 2013] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, 2013.

[Huang *et al.*, 2018] Yanping Huang, Yonglong Cheng, Dehao Chen, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Z. Chen. Gpipe: Efficient training of giant neural networks using pipeline parallelism. *ArXiv*, abs/1811.06965, 2018.

[Jiang *et al.*, 2020] Biye Jiang, Pengye Zhang, Rihan Chen, Xinchen Luo, Yin Yang, Guan Wang, Guorui Zhou, Xiaoqiang Zhu, and Kun Gai. Dcaf: A dynamic computation allocation framework for online serving system. *arXiv preprint arXiv:2006.09684*, 2020.

[Khosravi *et al.*, 2017] Atefeh Khosravi, Lachlan L. H. Andrew, and Rajkumar Buyya. Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data centers. *IEEE Transactions on Sustainable Computing*, 2:183–196, 2017.

[Kitaev *et al.*, 2020] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *ArXiv*, abs/2001.04451, 2020.

[Lacoste *et al.*, 2019] Alexandre Lacoste, Alexandra Sasha Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *ArXiv*, abs/1910.09700, 2019.

[Li *et al.*, 2020] Xiaocheng Li, Chunlin Sun, and Yinyu Ye. Simple and fast algorithm for binary integer and online linear programming. *arXiv preprint arXiv:2003.02513*, 2020.

[Liu *et al.*, 2021] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, and Dongrui Fan. Sampling methods for efficient training of graph convolutional networks: A survey. *IEEE/CAA Journal of Automatica Sinica*, 9:205–234, 2021.

[Ma *et al.*, 2018] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1137–1140, 2018.

[Molchanov *et al.*, 2016] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv: Learning*, 2016.

[Pan *et al.*, 2019] Feiyang Pan, Xiang Ao, Pingzhong Tang, Min Lu, Dapeng Liu, Lei Xiao, and Qing He. Field-aware calibration: A simple and empirically strong method for reliable probabilistic predictions. *Proceedings of The Web Conference 2020*, 2019.

[Qin *et al.*, 2022] Jiarui Qin, Jiachen Zhu, Bo Chen, Zhirong Liu, Weiwen Liu, Ruiming Tang, Rui Zhang, Yong Yu, and Weinan Zhang. Rankflow: Joint optimization of multi-stage cascade ranking systems as flows. *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

[Schwartz *et al.*, 2019] Roy Schwartz, Jesse Dodge, Noah Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63:54 – 63, 2019.

[Shukur *et al.*, 2020] Hanan M. Shukur, Subhi R. M. Zeebaree, Abdulraheem Jamil Ahmed, Rizgar R. Zebari, Omar M. Ahmed, Bareen Shams Aldeen Tahir, and Mohammed A. M. Sadeeq. A state of art survey for concurrent computation and clustering of parallel computing for distributed systems. *Journal of Applied Science and Technology Trends*, 2020.

[Tirmazi *et al.*, 2020] Muhammad Tirmazi, Adam Barker, Nan Deng, Md Ehtesam Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. Borg: the next generation. In *EuroSys'20*, Heraklion, Crete, 2020.

[Wang *et al.*, 2011] Lidan Wang, Jimmy J. Lin, and Donald Metzler. A cascade ranking model for efficient ranked retrieval. *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 2011.

[Xu *et al.*, 2021] Jingjing Xu, Wangchunshu Zhou, Zhiyi Fu, Hao Zhou, and Lei Li. A survey on green deep learning. *ArXiv*, abs/2111.05193, 2021.

[Yang *et al.*, 2021] Xun Yang, Yunli Wang, Cheng Chen, Qing Tan, Chuan Yu, Jian Xu, and Xiaoqiang Zhu. Computation resource allocation solution in recommender systems. *ArXiv*, abs/2103.02259, 2021.

[Yigitcanlar *et al.*, 2021] Tan Yigitcanlar, Rashid Mehmood, and Juan Manuel Corchado. Green artificial intelligence: Towards an efficient, sustainable and equitable technology for smart cities and futures. *Sustainability*, 2021.

[Zhou *et al.*, 2017] Guorui Zhou, Cheng-Ning Song, Xiaoqiang Zhu, Xiao Ma, Yanghui Yan, Xi-Wang Dai, Han Zhu, Junqi Jin, Han Li, and Kun Gai. Deep interest network for click-through rate prediction. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2017.

[Zhou *et al.*, 2018] Guorui Zhou, Na Mou, Ying Fan, Qi Pi, Weijie Bian, Chang Zhou, Xiaoqiang Zhu, and Kun Gai. Deep interest evolution network for click-through rate prediction. In *AAAI Conference on Artificial Intelligence*, 2018.