Online Request Scheduling for Quality-Aware Diffusion-Based AIGC Services

Han Yang, Ying Zheng, Lei Jiao, Yuedong Xu, Zongpeng Li

***Abstract*—Artificial Intelligence-Generated Content (AIGC) has been gaining significant traction for automatic generation of diverse content. Due to the GPU-intensive generation process and the high costs associated with purchasing and operating GPUs, users often prefer to submit requests to a nearby edge cloud, maintained by an AIGC cloud service provider. Efficiently scheduling AIGC requests in the edge cloud faces non-trivial challenges. First, AIGC requests emphasize the quality of generated content, yet conventional scheduling algorithms often overlook this aspect. Second, when the volume of incoming requests exceeds the capacity of the cloud, the AIGC service provider needs to select appropriate requests to execute, which is further complicated by the online arrival pattern of requests and the constraints imposed by request deadlines. Third, users dynamically submit multiple requests at different times. To manage costs, each user operates within a pre-allocated budget for a given time period. For the AIGC cloud service provider, it is highly non-trivial to identify valuable requests and judiciously balance different user budgets. To tackle the above challenges, we target the online AIGC request scheduling problem with the new objective of maximizing the overall content generation quality. We first conduct real experiments to establish the quality model between inference steps and the quality of generated content. Then, based on this quality model, we formulate the problem into an integer linear program, which is proven NP-hard. Under a primal-dual framework, we carefully design the update of multiple dual variables, to flexibly control the consumption of edge resources and user budgets. We rigorously analyze the performance of the proposed algorithm and prove a theoretical performance guarantee on its competitive ratio. Extensive real-world trace-driven experiments manifest that our proposed method improves the state-of-the-art by up to 25.3% in overall content generation quality.**

***Index Terms*—AIGC, Online optimization, scheduling algorithms**

I. Introduction

Artificial Intelligence-Generated Content (AIGC) refers to content that is generated by advanced machine learning models, especially deep neural networks [1]. A user request, referred to as a "prompt", is fed into the neural network as input, and the desired content is generated through a model inference process. Recent years have witnessed a widespread use of AIGC in various domains [2]–[4]. Despite the promises, it is impractical for most users to directly deploy and utilize these models locally, due to the substantial computational and storage resources required by AIGC models. For instance, the popular Stable Diffusion model requires 20 GB of memory to generate results. Furthermore, the inference process is highly GPU-intensive. It takes only 30 seconds to run the inference process with 20 steps on an Nvidia A100 GPU, while it requires nearly 8 minutes on an Intel Xeon Gold 6348 CPU. As a result, it is more cost-effective for users to offload their AIGC requests to a cloud service provider who provides AIGC-as-a-Service (AaaS) over GPUs.

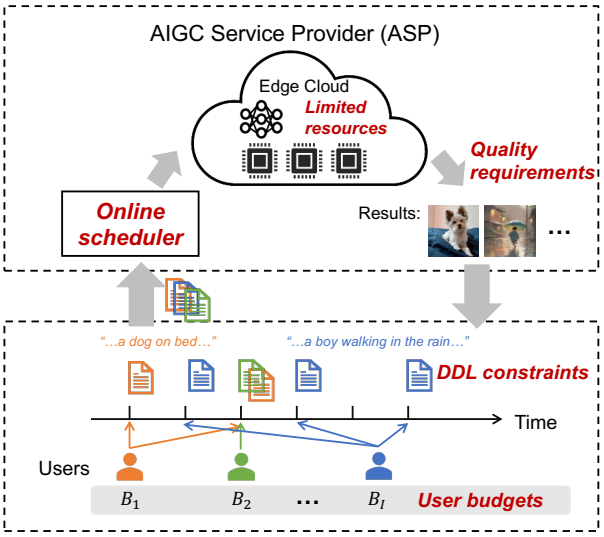


Fig. 1: AIGC request serving in an edge cloud.

AaaS represents a new computing paradigm that promise on-demand access to powerful computation resources equipped with a variety of AIGC models [5], [6]. As shown in Figure 1, in the AaaS ecosystem, an AIGC service provider (ASP) deploys models on the edge cloud, offering online services to users over edge networks, often wireless. Users dynamically submit AIGC requests to the ASP for processing, willing to pay monetary remuneration for the service. For the service provider, an efficient request scheduling algorithm is of crucial importance, since it directly impacts the AIGC service provisioned. Such a scheduling algorithm is challenging to design, due to the following challenges.

This work was supported in part by the Quan Cheng Lab (QCL20250108, QCL20250202 and QCL20250205), in part by Shandong Provincial Natural Science Foundation (ZR2024LZH011, ZR2025LZH008), in part by the U.S. National Science Foundation (CNS-2047719, CNS-2225949), and in part by the Natural Science Foundation of China under Grant Grant 62472103.

Han Yang and Zongpeng Li are with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China, and with the Quancheng Laboratory, Jinan, Shandong 250103, China (e-mail: h-yang23@mails.tsinghua.edu.cn; zongpeng@tsinghua.edu.cn).

Ying Zheng is with the School of Computer Science, Fudan University, Shanghai 200438, China (e-mail: zhengy18@fudan.edu.cn).

Lei Jiao is with the Center for Cyber Security and Privacy, University of Oregon, Eugene, OR 97403, USA (e-mail: ljiao2@uoregon.edu).

Yuedong Xu is with the Artificial Intelligence Innovation and Incubation Institute, Fudan University, Shanghai 200438, China (e-mail: ydxu@fudan.edu.cn).

First, AIGC requests emphasize content generation quality, while existing scheduling algorithms are often quality-agnostic. These algorithms are developed to optimize the commonly-adopted Quality of Service (QoS) metrics, such as job completion time and throughput, and hence are incapable of addressing user requirements on generation quality. Furthermore, measuring generation quality is a challenging problem in it own right, given to the subjective nature of user preferences. For text-to-image tasks based on diffusion models, the number of inference steps affects the quality of the generated result. Typically, the more inference steps, the higher the computational cost, and the better the generated results. Unfortunately, the relationship between generation quality and number of inference steps is not simply linear. This requires us to dive into the AIGC system, establish new quality-aware models, and design new scheduling algorithms that work in concert with it.

Second, resources in an edge cloud are limited. When incoming requests exceed the service capacity of the edge cloud, the ASP can only admit and execute a subset of the submitted requests. If the scheduling algorithm admits requests aggressively, it might prematurely exhaust resources, unable to accommodate later requests with potentially high valuation; if the algorithm behaves too conservatively, it faces the risk of wasting resources instead. The challenge further escalates when we consider the strict deadlines imposed on AIGC requests. We aim to design an online scheduling algorithm that strikes a judicious balance between immediate and future system welfare in an AIGC-as-a-Service ecosystem.

Third, users dynamically submit multiple requests at different times, hoping to keep monetary expenses in a given period within a predefined budget [7]–[9]. However, users are not clear about the quality of the generated results until they receive them. Therefore, the ASP should help users identify more valuable requests through the admission control mechanism, and maximize overall quality of the generated results while ensuring budget compliance.

To our knowledge, existing studies fail to address the aforementioned challenges to satisfaction. Xu *et al.* [10], Liu *et al.* [11] and Du *et al.* [12] target optimizing the QoS of AIGC services. They focus on the deployment of AIGC models and fail to consider the request scheduling problem. Zheng *et al.* [13], Lyu *et al.* [14] and Xu *et al.* [15] investigate the AIGC request scheduling problem, while ignoring the issue of generation quality. Du *et al.* [16] leverage a Deep Reinforcement Learning (DRL) method to tackle the quality-aware AIGC service provider selection problem. However, they ignore the constraints imposed by user budgets and request deadlines.

In this work, from the perspective of the AIGC service provider, we study how to schedule dynamically arrived AIGC requests in an online manner, with the goal of maximizing the overall content generation quality. Specifically, our contributions are as follows

- We explore various text-to-image AIGC models with advanced image quality assessment (IQA), and discover a mathematical relation between the number of inference steps and generation quality. Then we establish quality perception models for the relationship via real-world deployment and extensive testing.
- We formulate a long-term optimization problem to maximize *content generation quality* of the AIGC ecosystem. This problem is an integer linear program, computationally intractable even in the offline setting. Our formulation grasps all the aforementioned challenges and is general, with only mild assumptions on input dynamics and heterogeneity.
- As an important algorithmic step, we reformulate the optimization problem into an equivalent form amenable to primal-dual optimization. Then, we derive the dual problem and design an online algorithm through online primal-dual theory. Our primal-dual algorithm dynamically conducts admission control and decides the schedule for each request as it arrives at the AIGC-as-a-Service system.
- We rigorously prove multiple theoretical results, including the NP-hardness of the problem, the feasibility of our approach, the polynomial-time complexity of our algorithm, as well as its competitive performance against the offline optimum.
- We conduct extensive experiments, in which our proposed approach outperforms competing alternatives in various settings. Specifically, in the common scheduling scenario, our approach improves total generation quality by 25.3%, 32.7%, 50.8% and 52.1% compared to four baseline algorithms and achieves a significant algorithm runtime speedup compared to the state-of-the-art DRL-based approach.

II. Related works

Recently, there have emerged studies that aim at improving the QoS of AIGC services. Xu *et al.* [10] propose a least context algorithm for managing cached models at edge servers, meticulously balancing among latency, energy consumption and accuracy. Similarly, Liu *et al.* [11] develop a comprehensive conceptual model to integrate AIGC with semantic communication, offering a more comprehensive approach to system design. In the context of text-to-image tasks, Du *et al.* [12] introduce three architectures for joint inference between edge computing nodes and user devices, successfully reducing the consumption of inference resources by sharing inference steps among users. Meanwhile, Feng *et al.* [17] propose a novel framework to optimize service delay, network resource usage, and computing resources of user devices in AIGC services. While these studies make significant contributions, they primarily concentrate on the model deployment or service delivery schemes, without considering the scheduling challenges in the AIGC service system.

Deng *et al.* [18] study joint AIGC request scheduling and model deployment in wireless edge networks, aiming to minimize the total system cost during task execution and the accuracy loss of generated results. However, they focus on

an offline problem, assuming a fixed number of AIGC tasks rather than accounting for the dynamic nature of online task arrivals. To address the problem of scheduling online AIGC requests, Zheng *et al.* [13] propose an AIGC DAG scheduling algorithm in an edge-cloud environment, while Lyu *et al.* [14] explore the challenge of selecting appropriate ASPs in a collaborative AIGC framework. Similarly, Xu *et al.* [15] investigate an online text-to-image AIGC request scheduling problem in a heterogeneous edge computing scenario. They first formulate the problem into an integer program, and then leverage deep learning methods to balance request latency with the computation resources. They overlook the constraints of user budgets and do not sufficiently investigate the quality of content generation. To improve the quality of generated results, D2SAC [16] focuses on the AIGC service provider selection problem, with the aim of maximizing the overall quality of generated content. This work introduces an innovative mathematical model to evaluate quality, and employs deep reinforcement learning to select service providers for executing AIGC requests. Despite its strengths, the proposed method also fails to consider the constraints imposed by strict deadlines and user budgets, and hence is not applicable to our problem.

For studies on designing online scheduling algorithms, the primal-dual framework offers valuable insights for our algorithm design. Zhou *et al.* [19] develop an efficient social welfare approximation algorithm using the classic primal-dual framework, leveraging both LP duals and Fenchel duals. This approach provides a useful reference for addressing resource constraints and managing both soft and hard deadline limitations in online scheduling scenarios. Similarly, Shi *et al.* [9] demonstrate the application of online optimization algorithms to handle budget constraints. They propose a binary search algorithm to improve average-case performance and enhance the online auction framework by ensuring a minimum budget spending fraction, resulting in a better competitive ratio. However, these approaches, along with other related works [20], [21], primarily focus on cost and resource constraints, while neglecting the quality of the generated results. This omission makes them unsuitable for addressing the unique challenges of the diffusion-based AIGC request scheduling problem, where ensuring the quality of generated content is necessary.

Therefore, none of the existing works address the online request scheduling problem for quality-aware diffusion-based AIGC services under pre-defined user budget constraints. Both a new formulation and an online solution are needed.

III. Background

A. Diffusion Model

Mainstream models for text-to-image generation tasks, *e.g.*, Stable Diffusion [22] and DALL·E [23], are trained over diffusion models [24]. Diffusion models consist of a diffusion process and a denoise process, also referred to as the inference process. As shown in Figure 2, during the diffusion process, Gaussian noise is progressively added into the original image, eventually transforming it into a completely noisy image after T steps. The sequence from $\mathbf{x}_0$ to $\mathbf{x}_T$ represents the images

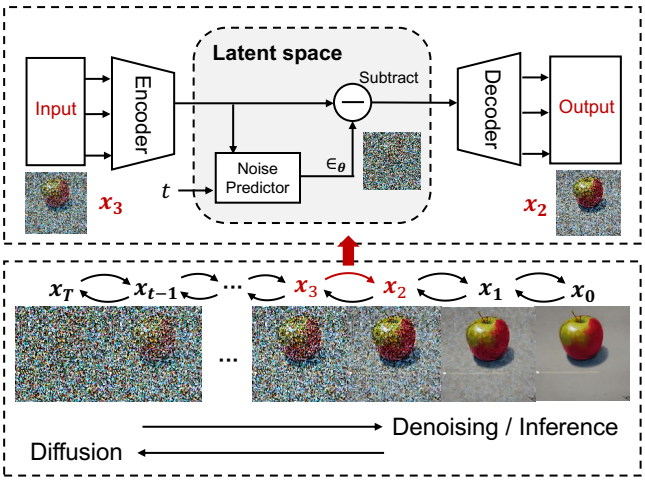


Fig. 2: Illustration of how a diffusion-based AIGC model works: the lower half of the image depicts the overall workflow, while the upper half focuses on a specific denoising step.

generated throughout the diffusion process, each containing varying degrees of noise. Specifically, $\mathbf{x}_0$ corresponds to the original image without any added noise while $\mathbf{x}_T$ corresponds to the completely noisy image.

The training process of a diffusion model is designed to train a noise predictor, typically implemented as a neural network, which estimates the noise added to the image based on the noisy image and its corresponding noise step. Let θ denote the parameters of the noise predictor. For each sampled original image data point, the parameters of the predictor network are trained and updated using the equation

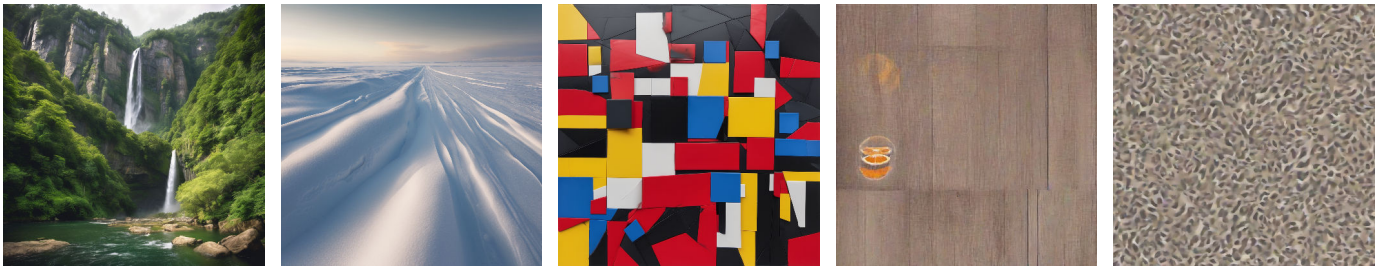
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \boldsymbol{\epsilon}, t \right) \right\|^2, \tag{1}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ represents the sampled random noise; t denotes the number of noising step during sampling. The core idea of Equation (1) is to add the sampled random noise ϵ into the original image $\mathbf{x_0}$, producing a noisy image $\mathbf{x}$ defined as

$$\mathbf{x} = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \boldsymbol{\epsilon}. \tag{2}$$

The noisy image $\mathbf{x}$, along with the noise step t, is then fed into a noise predictor neural network, which produces the predicted noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}, t)$. The training objective is to minimize the difference between the predicted noise and the true noise label, to ensure the predicted noise is sufficiently accurate.

The denoising process progressively reconstructs the original image from a noisy image using the trained model [25]. In a specific text-to-image generation task, the Stable Diffusion algorithm begins by randomly sampling a Gaussian noise image. Simultaneously, a text encoder is adopted to process user textual input, encoding it into a latent vector. This encoded vector, combined with the current number of inference step t, is passed to the noise predictor to estimate the noise at that step. By iteratively removing the predicted noise, the model gradually synthesizes the final image that aligns with user input [26], [27]. Notably, to enhance efficiency, Stable Diffusion does not operate directly on the original image or



(a) CLIP-IQA Score:0.829 (b) CLIP-IQA Score:0.835 (c) CLIP-IQA Score:0.494 (d) CLIP-IQA Score:0.284 (e) CLIP-IQA Score:0.159

Fig. 3: Examples of generation results and their corresponding CLIP-IQA scores. From left to right, the prompts are: (a) "A towering waterfall surrounded by green forests"; (b) "A snowy landscape stretching to the horizon"; (c) "Multicolored blocks spread out on a table"; (d) "A juicy orange on the wooden desk"; (e) "A patterned kitten with soft fur playing in the courtyard".

its noisy counterpart. Instead, it processes their representations in a compressed latent space. For instance, a 512×512 pixel image is reduced to a 4×64×64 representation in the latent space, drastically reducing computational and storage demands during training and inference [22].

Specifically, as shown in Figure 2, each inference step brings an improvement to the current image. As the number of steps increases, the image becomes progressively clearer and more refined. After T steps, the final result in the latent space is decoded into the final image by the image decoder. Consequently, in typical inference tasks of large text-to-image models, the number of inference steps (*i.e.*, denoising steps) directly affects the computational resource consumption and the quality of the generated results.

B. Image Quality Assessment

In an AIGC ecosystem, users submit requests to the server to generate desired images. Throughout this process, in addition to traditional QoS metrics like latency and cost, a primary concern for users is the quality of the generated images. Apparently, users are more likely to pay for generated results that meet their expectations.

However, evaluating the quality of generated images has been a persistent challenge in the field of computer vision [28]. Various mathematical models have been developed to evaluate image quality [29], [30]. The Fréchet Inception Distance (FID) metric [31] and Inception Score (IS) [32] metrics quantify the quality of generated images by measuring the similarity between the high-level features of generated images and those of real-world reference images. However, these IQA metrics primarily focus on evaluating sets of generated images, whereas our generation objective mandates a focus on the evaluation of individual, specific image produced by the inference service. BRISQUE [29] is a no-reference method for assessing the quality of individual images by extracting mean subtracted contrast normalized (MSCN) coefficients from the image and fitting them to an asymmetric generalized Gaussian distribution (AGGD). Features derived from the fitted Gaussian distribution are then extracted and fed into a support vector machine (SVM) for regression, ultimately producing an image quality assessment score based on objective metrics like texture, clarity, and smoothness. However, the quality we seek to evaluate must also encompass the user's overall perception, i.e., subjective "feel", of the image. In this regard, traditional methods such as BRISQUE and similar approaches [33] often fall short.

To address the aforementioned challenge and provide a more comprehensive evaluation of the quality of generated images, we introduce an advanced and effective image quality assessment (IQA) method, CLIP-IQA [34]. This approach leverages a Contrastive Language-Image Pre-training (CLIP) model to assess both the appearance (quality perception) and feel (abstract perception) of images, all without requiring explicit task-specific training. Specifically, the generated images are processed through the CLIP model for feature extraction. These extracted features are then compared with a pre-defined set of quality-related keywords (e.g., "high quality," "low quality") using cosine similarity shown below

$$u_i = \frac{\mathbf{y} \odot \mathbf{t}_i}{\|\mathbf{y}\| \cdot \|\mathbf{t}_i\|}, \quad i \in \{1, 2\}, \tag{3}$$

where $\mathbf{y}$ represents the image feature representation extracted using the CLIP model; $\mathbf{t}_1$ and $\mathbf{t}_2$ denote the feature vectors of a pair of antonym prompts (e.g., "high quality," "low quality"). The quality score of image $\bar{s}$ is then computed as

$$\bar{s} = \frac{e^{u_1}}{e^{u_1} + e^{u_2}}. \tag{4}$$

The value of $\bar{s}$ ranges from 0 to 1, with larger values indicating better overall image quality.

The existing literature has demonstrated that CLIP-IQA achieves high precision in visual perception assessments and maintains strong consistency with human perception across various tasks [34]. We also conduct real experiments to further validate the effectiveness of CLIP-IQA in text-to-image generation scenarios. As illustrated in Figure 3, we present five generated images using the SDXL model [35]. Figure 3(a) and 3(b) illustrate high-quality images produced through sufficient inference steps, exhibiting excellent image texture features and leaving a positive impression, thus receiving high scores. The objective quality of Figure 3(c), reflected in its color and texture, is also high; however, it leaves only an average subjective impression, leading to a moderate score. In contrast, Figure 3(d) and 3(e), generated with insufficient inference

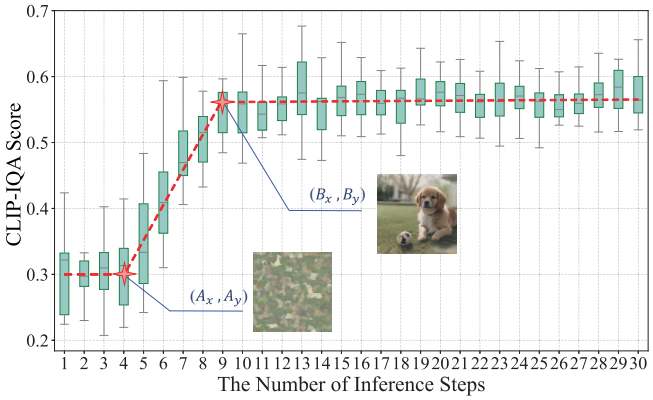


Fig. 4: Relation between number of inference steps and CLIP-IQA scores. Dashed line represents the quality model and the stars mark two important turning points.

steps, exhibit significant discrepancies from the prompts and poor overall quality, resulting in low scores.

CLIP-IQA allows for a holistic assessment covering both technical coherence and subjective impression, providing an accurate quality assessment for individual images without requiring reference images. By adopting the method, we successfully transform the abstract metric of image quality into numerical scores that can be used for subsequent modeling.

IV. Modeling and Problem Formulation

In this section, we first introduce the quality-aware model of generated results in Section IV-A. Then, we present the system models of AIGC services within the edge cloud environment in Section IV-B. Finally, the overall problem formulation is provided in Section IV-C.

A. Quality Model of Generated Results

In traditional image quality assessment methods, including CLIP-IQA, a specific image is essential as input to generate a quality score. However, in our online scheduling task, the system receives user requests rather than images. The image for quality evaluation can only be obtained after the request is processed and the inference task has been completed. By that stage, computation resources have already been consumed, making the quality evaluation ineffective for decision-making. To address this limitation, it is necessary to establish a proactive quality-aware model capable of estimating generation quality in advance, relying solely on the parameters of the user's request. As discussed in Section III-A, the number of inference steps has a direct impact on generation quality. On this basis, literature [5] demonstrates that, the relationship between the inference process and image quality can be approximately characterized using a mathematical function in image restoration context. We further extend to the text-to-image generation scenario and introduce Equation (5) as our quality-aware model represented by the red dashed line in Figure 4.

$$f(x) = \begin{cases} A_y, & \text{if } x < A_x, \\ \frac{B_y - A_y}{B_x - A_x}(x - A_x) + A_y, & \text{if } A_x \le x \le B_x, \\ B_y, & \text{if } x > B_x. \end{cases} \quad (5)$$

In this equation, x denotes the number of inference steps; A_x represents the minimum number of inference steps where the quality of the generated image starts to improve; B_x serves as the threshold beyond which additional inference steps no longer result in significant improvements in image quality. A_y and B_y represent the lower and upper bounds of the quality of the generated images, respectively, reflecting the model's inherent ability to generate images. They are closely tied to the intrinsic properties of the model deployed on the AIGC server, with better-performing and more effective models generally having higher values of A_y and B_y.

The values of A_x, A_y, B_x, and B_y can be obtained through real experiments. Specifically, we set up an experimental platform on an Ubuntu 20.04 server equipped with an NVIDIA RTX A100 (80GB) GPU and an Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz. On this platform, we deploy the *OpenDalleV1.1* [23] for image generation. To derive the four parameters of this model, we utilize it for image generation tasks, keeping the input prompt fixed while varying the number of inference steps. The generated images are then sent to the evaluation module to be scored using CLIP-IQA method [34] as we have introduced. For each inference step, the model repeatedly generates multiple images for quality evaluation. We then compile the results into a box plot as shown in Figure 4. According to the box plot, we fit the red dashed line and obtain the values of the turning points, corresponding to A_x, A_y, B_x and B_y. On this basis, we can establish a quality-aware model in the system. When subsequent AIGC requests arrive, the quality-aware model can estimate the quality of the generation result through Equation (5) and the number of inference steps specified by the user. The calculated quality value will serve as our optimization target, enabling more informed and effective decision making. Notably, the quality-aware model can be equipped with different methods to calculate the generation quality, including conducting any tuning, re-fitting, or probing, for different AIGC requests.

We expand the testing framework on the experimental platform to incorporate additional diffusion-based text-to-image models, such as *stable-diffusion-xl-base-1.0* [35], and *stable-diffusion-2-1* [22]. Quality-aware models can also be constructed utilizing the method described above. At the same time, we test whether the prompts provided by users would impact the quality of the generated content, as users in real-world scenarios rarely make identical requests all the time. We select several specific inference steps and randomly choose 15 types of real user prompts from the DiffusionDB dataset [36] for evaluation. The results as Figure 4 indicate that the content of the prompt does not affect the quality of the generated images. This is primarily due to the presence of the encoder

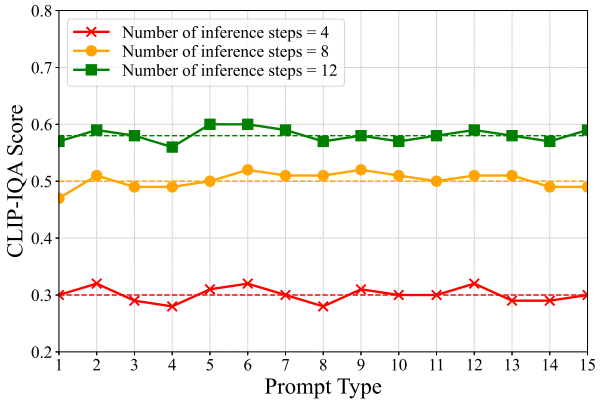


Fig. 5: Relation between prompt types and CLIP-IQA scores. Each point is calculated as the average of 20 generated images.

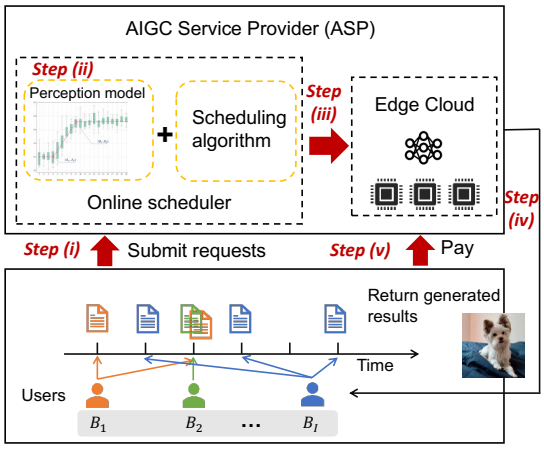


Fig. 6: The system workflow of the request scheduling system.

module in diffusion-based text-to-image models, where different prompts will be encoded into latent space vectors of the same dimension for subsequent diffusion. Although the above conclusion holds true in our tests, we do not rule out the possibility that certain hard prompts may lead to results that deviate from the curve shown in Figure 5. We remark that such cases does not impact our scheduling models, algorithms, and analysis, as long as we are provided with the generation quality of each AIGC request. Additionally, Since our algorithm is designed for AIGC service providers, a practical enhancement for real deployments is to update the step-to-quality curve online. This does not affect our scheduling model or theoretical analysis. The curve can be dynamically adapted based on the evolving distribution of user prompts observed in the system, allowing it to better fit real-time user demand and further mitigate the impact of hard prompts.

B. System Models

AIGC Service Provider (ASP): The ASP deploys its AIGC model on an edge server. Continuous maintenance and updates are required to ensure that the AIGC model remains accurate and effective in generating high-quality content. Users submit requests for content generation and receive the generated content from the ASP. Without loss of generality, assume that the entire system operates in slotted time $[T] = \{1, 2, \cdots, T\}$. Due to resource constraints, the services that can be provided in a specific time slot are limited. We extract this upper-bound as resource capacity C, representing the maximum number of inference steps the ASP can process in a single time slot.

AIGC Users and Requests: We use $[I] = \{1, 2, \cdots, I\}$ to refer to the set of AIGC service users. Each user submits its AIGC requests to the ASP at different time slots. Each user also has a budget B_i, hoping the total monetary expenses do not exceed the pre-defined limit within a certain period [7]–[9]. We use $\mathcal{M}_i = \{M_{i1}, M_{i2}, \cdots, M_{ij}, \cdots\}$ to represent the requests submitted by user i, and the total number of which is $J_i = |\mathcal{M}_i|$. Note that the number of requests may vary for different users. We use $M_{ij} = \{a_{ij}, d_{ij}, s_{ij}\}$ to represent user i's j-th request, where a_{ij} is the arrival time of the request; d_{ij} is the deadline for completing the request; s_{ij} refers to the number of inference steps required for executing the request. We use r_{ij} to represent the required computation resources by user i's j-th request, whose value is a linear function of s_{ij} as stated in [16]. If the ASP executes a submitted request, the corresponding user needs to make a payment of e_p to the ASP to attain the result.

Inference Quality: As mentioned in Section III-B, we select the CLIP-IQA as our image quality assessment method due to its superior performance in assessing image quality. Denote u_{ij} as the quality of the generated image in terms of user i's j-th request. Based on the CLIP-IQA method, we are able to establish the mathematical relationship between the quality of the generated image u_{ij} and the number of inference steps s_{ij}. This also serves as a part of our optimization goal.

Control Decisions: Upon each AIGC request M_{ij} arrival, i.e., user i's j-th request, the ASP responds immediately and makes the following control decisions: (i) Whether to admit the user i's j-th request for execution, denoted by x_{ij}; (ii) Whether to execute the user i's j-th request at time slot t, $\forall t \geq a_{ij}$, denoted by y_{ijt}.

System Workflow: The overall system workflow is shown in Figure 6, consisting of the following steps: (i) Users submit their AIGC requests to the ASP. Note that each user may submit multiple requests, across different time slots; (ii) Upon each request arrival, the ASP conducts quality perception based on the pre-measured perception model; (iii) The ASP makes the scheduling decisions, including whether to execute the current request and, if so, the time slots of execution; (iv) The ASP returns the result to the corresponding user; (v) The user makes a payment to the ASP for executing its request.

C. Optimization Problem Formulation

Optimization Objective: We define the *overall generation quality* as our optimization objective, which is calculated as

$$U = \sum_{i \in [I]} \sum_{j \in [J_i]} u_{ij} x_{ij}, \tag{6}$$

where x_{ij} is a control decision, whose value is 1 if the ASP accepts the user i's j-th request, and 0 otherwise. The value

of u_{ij} depends on the number of required inference steps (i.e., s_{ij}) and the established quality model (Equation (5)).

Optimization Problem: We aim to maximize the overall generation quality, subject to deadline, resource and budget constraints. Some deployments may consider jointly optimize multiple objectives, such as latency, diversity, and safety filters. To handle this, we can perform the weighted sum of the multiple different objectives and use this sum as the single objective for optimization. Such weights can always be tuned and controlled, e.g., depending on the importance of each objective, in order to navigate the final optimization results. This way, our proposed approach still applies.

The diffusion-based AIGC request scheduling problem can be formulated as

$$\text{maximize} \quad \sum_{i\in[I]}\sum_{j\in[J_i]} u_{ij}x_{ij} \tag{7}$$

$$\text{subject to} \quad y_{ijt}\cdot t - d_{ij} \le 0, \forall i\in[I], j\in[J_i], t\in[T], \tag{7a}$$

$$x_{ij} - \sum_{t\ge a_{ij}} y_{ijt} \le 0, \forall i\in[I], j\in[J_i], \tag{7b}$$

$$\sum_{i\in[I]}\sum_{j\in[J_i]} r_{ij}y_{ijt} \le C, \forall t\in[T], \tag{7c}$$

$$\sum_{t\in[T]}\sum_{j\in[J_i]} e_p y_{ijt} \le B_i, \forall i\in[I], \tag{7d}$$

$$x_{ij}, y_{ijt} \in \{0,1\}, \forall i\in[I], j\in[J_i], t\in[T]. \tag{7e}$$

Constraint (7a) ensures each request is executed before its deadline. Constraint (7b) ensures that for each request, at least one time slot will be used to execute it once selected. Constraint (7c) ensures sufficient computation capacity of ASP to complete the inference request. Constraint (7d) guarantees total expenses do not exceed user budgets. Constraint (7e) specifies the domains of the decision variables. Unless otherwise noted, the scopes for our indices are $i\in[I], j\in[J_i]$, and $t\in[T]$. Our problem is provably intractable; see Section VI-A for more details.

TABLE I: Main Notations.

I	number of users
T	number of time slots
J_i	number of requests submitted by user i
B_i	user i's budget
C	computation resource capacity of the edge cloud
e_p	payment to ASP for executing an AIGC request
a_{ij}	arrival time of user i's j-th request
d_{ij}	deadline specified by user i's j-th request
u_{ij}	quality of user i's j-th request
r_{ij}	required computation resources of user i's j-th request
s_{ij}	number of required inference steps of user i's j-th request
$z^{(ij)}$	total amount of resource consumption after scheduling user i's j-th request
$\lambda_t^{(ij)}$	value of dual variable λ_t after processing user i's j-th request
$\varphi_i^{(j)}$	value of dual variable φ_i after processing user i's j-th request
Decisions	Descriptions
x_{ij}	whether to execute(1) user i's j-th request or not(0)
y_{ijt}	whether to execute(1) user i's j-th request at time slot t or not(0)

V. Online Algorithm Design

In this section, we describe how to design our online algorithm. We first introduce the overall idea in Section V-A. In Section V-B, we reformulate the optimization problem. Then, the online algorithm is presented in Section V-C.

A. Overview and Rationale

The original problem (7) is an integer linear program, which contains a standard 0-1 knapsack problem and hence is NP-hard. To solve problem (7), we first reformulate and simplify it into problem (8). However, we are still unable to obtain the solution directly due to the unknown job information, *i.e.*, the coefficients of problem (8) are unknown until the corresponding requests appear. In other words, problem (8) is an *online* problem and we need to develop an *online* algorithm to solve it based only on current and past information.

To design such an online algorithm with provably-guaranteed performance, we leverage the online primal-dual framework. The core idea is that, as the task arrives and the constraints in problem (8) appear dynamically, we always keep a feasible solution for the primal problem (8) and a feasible solution for the dual problem (9) through a carefully-designed request scheduling mechanism and update rules of dual variables. Specifically, we introduce dual variable λ_t to track the computation resource consumption and implicitly reflect the marginal price of computation resources in the system. Based on the value of dual variable λ_t, the ASP is able to strategically identify and admit valuable requests among incoming requests. Meanwhile, we also introduce dual variable φ_i for each user i to manage its budget consumption. Our goal is to maximize each user's dwell time in the system, thereby increasing the likelihood of occurring more valuable requests. To achieve this, we design an update mechanism of φ_i such that when user i's budget consumption is relatively low, the ASP is more inclined to execute tasks submitted by the user. In contrast, when user i's budget consumption is high, the ASP is more inclined to reject the requests submitted by the user.

In addition, we also keep the objective values incurred by the primal feasible solution and the dual feasible solution satisfying a certain relationship. The weak duality indicates that the objective value incurred by a dual feasible solution always serves as an upper bound for the optimal objective value of the primal problem. Therefore, the performance of the online solution is guaranteed.

B. Problem Reformulation

Primal Problem: We first reformulate the optimization problem (7) into an equivalent yet simpler integer linear program (ILP) as

$$\text{maximize} \sum_{i\in[I]}\sum_{j\in[J_i]}\sum_{t\in[a_i,d_i]} u_{ij}y_{ijt} \tag{8}$$

$$\text{subject to} \sum_{t\in[T]} y_{ijt} \le 1, \forall i \in [I], j \in [J_i], \tag{8a}$$

$$\sum_{i\in[I]}\sum_{j\in[J_i]} r_{ij}y_{ijt} \le C, \forall t \in [T], \tag{8b}$$

$$\sum_{t\in[T]}\sum_{j\in[J_i]} e_p y_{ijt} \le B_i, \forall i \in [I], \tag{8c}$$

$$y_{ijt} \in \{0,1\}, \forall i \in [I], j \in [J_i], t \in [T]. \tag{8d}$$

In problem (8), we eliminate the variable x_{ij} by incorporating the constraint (7b) into the objective function. Constraint (8a) ensures that each request can be executed at most once. Constraints (8b) and (8c) are equivalent to (7c) and (7d), respectively. Constraint (8d) specifies the domain of the control variable y_{ijt}. We remark that a feasible solution to problem (8) has a corresponding feasible solution to problem (7), and the optimal objective value of problem (8) is equal to that of problem (7).

C. Online Scheduling

Dual Problem: To design the online scheduling algorithm, we relax the domain of the integer variable y_{ijt} to $y_{ijt} \in [0,1]$. The dual problem of (8) can be written as

$$\text{minimize} \sum_{i\in[I]}\sum_{j\in[J_i]} \delta_{ij} + \sum_{t\in[T]} C\lambda_t + \sum_{i\in[I]} B_i\varphi_i \tag{9}$$

$$\text{subject to } \delta_{ij} \ge u_{ij} - e_p\varphi_i - r_{ij}\lambda_t, \forall i \in [I], j \in [J_i], t \in [T], \tag{9a}$$

$$\delta_{ij} \ge 0, \lambda_t \ge 0, \varphi_i \ge 0, \forall i \in [I], j \in [J_i], t \in [T], \tag{9b}$$

where δ_{ij}, λ_t and φ_i are the dual variables associated with Constraints (8a), (8b) and (8c), respectively.

We define the value of dual variable λ_t as

$$\lambda_t^{(ij)} = \frac{1}{e}\cdot\theta^{\frac{z_t^{(ij)}}{C}},\ \forall i \in [I], \forall j \in [J_i], \tag{10}$$

where $\lambda_t^{(ij)}$ represents the value of λ_t after processing user i's j-th request; $z_t^{(ij)}$ denotes the total amount of resource consumption at time slot t after scheduling user i's j-th request. Next, we explain the design principles of updating dual variable λ_t. In the primal-dual theory, a dual variable is a "shadow price" [37] representing the increment of the objective value incurred by a unit increase in the resource of the original problem. In other words, λ_t, which is associated with Constraint (8b), can be interpreted as a marginal price function of edge resource. At each time slot t, if edge resources are sufficient, the resource price should be kept as low as possible to encourage the ASP to execute more AIGC requests. When resources are scarce, the resource price is high, allowing only more valuable requests to be executed. Note that the resource price will be updated after each request's scheduling. In this way, the ASP can decide whether to execute incoming requests based on the value of λ_t, enabling flexible resource management.

Meanwhile, denote $\varphi_i^{(j)}$ as the value of dual variable φ_i after processing user i's j-th request. The value of $\varphi_i^{(j)}$ is computed as

$$\varphi_i^{(j)} = \varphi_i^{(j-1)}(1+\frac{e_p}{B_i}) + \beta(\frac{u_{ij}}{B_i}),\ \forall i \in [I], \forall j \in [J_i]\setminus\{0\}. \tag{11}$$

We remark that the value of φ_i is only updated when a request submitted by user i is accepted and executed. Note the initial value of $\varphi_i^{(j)}$ is $\varphi_i^{(0)} = 0$. The coefficient β determines the magnitude of growth in the dual variable φ_i at each update. A larger β leads to faster updates of φ_i, while a smaller β results in slower updates. The value of β is chosen by design, and it affects the competitive ratio of the proposed online algorithm, as shown in Theorem 4. The rationale for designing the update rule of dual variable φ_i is that, we try to make each user's budget last for as long as possible in the T time slots, thus the ASP can explore more valuable requests over the entire time span.

For ease of representation, we define the right hand side of Constraint (9a) as

$$F(ijt) = u_{ij} - e_p\varphi_i - r_{ij}\lambda_t,\ \forall i \in [I], j \in [J_i], t \in [T]. \tag{12}$$

We also define

$$t^* = \underset{t\in[a_{ij},d_{ij}]}{\arg\max}\{F(ijt)\}. \tag{13}$$

Then we set the dual variable δ_{ij} as

$$\delta_{ij} = \max\{0, \max_t\{F(ijt)\}\},\ \forall i \in [I], j \in [J_i]. \tag{14}$$

In the online scheduling process, for each arrived user i's j-th request M_{ij}, if $F(ijt^*) > 0$, the ASP accepts the request M_{ij} and set $\delta_{ij} = F(ijt^*)$; otherwise the ASP rejects the request and set $\delta_{ij} = 0$.

Algorithm 1 is our online scheduling algorithm. Line 1 initializes the decision variable x_{ij}, y_{ijt} and dual variable λ_t, φ_i to zero. Upon the arrival of each request M_{ij}, Line 3 selects the time slot t^* in which $\lambda_t^{(ij-1)}$ returns the minimum value. Line 4 calculates the value of $F(ijt^*)$ according to (12). Line 5 checks whether $F(ijt^*) > 0$ and if so, line 6 updates dual variables λ_t and φ_i; otherwise, we reject user i's j-th request in line 11. Line 7 checks whether there are sufficient resources to execute the request and whether the user has a sufficient budget to be charged. Formally, we check the capacity constraint $\sum_i\sum_j r_{ij}y_{ijt} \le C, \forall t$, and the budget constraint $\sum_j\sum_t e_p y_{ijt} \le B_i, \forall i$. If there are sufficient resources and enough budget, then Line 8 admits the request and executes it at time t^*. Line 9 updates the consumed resources and users' budgets.

Algorithm 1: Online AIGC Request Scheduling Algorithm

Input: $\{M_{ij}\}$, C, e_p,$\{B_i\}$

1 Initialize $x_{ij}, y_{ijt}, \lambda_t, \varphi_i, \forall i, j, t$;
2 **for** *each incoming request* M_{ij} **do**
    `// find the request allocation decision that achieves the maximum` $F(ijt)$
3 Select the time slot $t^* = \arg\min_{t\in[a_{ij},d_{ij}]}\{\lambda_t^{(ij-1)}\}$;
4 Calculate $F(ijt^*)$ according to (12);
5 **if** $F(ijt^*) > 0$ **then**
6 Update λ_{t^*} and φ_i according to (10) and (11);
7 **if** *enough resources and user* i*'s budget* **then**
8 Admit user i's j-th request, execute it at time slot t^*, set $x_{ij} = 1$;
9 Update consumed resources and users' budgets;
10 **else** Reject user i's j-th request;
11 **else** Reject user i's j-th request;

VI. Performance Analysis

A. Intractability

Theorem 1. *The AIGC Request Scheduling Problem* (7) *is NP-hard.*

Proof. We prove by showing that problem (7) subsumes the well-known 0-1 knapsack problem as a special case. To see this, let $x_{ij} = \sum_t y_{ijt}$, thus Constraint (7b) always holds. Next, we sum Constraint (7c) over the entire horizon T and obtain $\sum_i \sum_j r_{ij}x_{ij} \le CT$. After removing Constraints (7a) and (7d), the remaining problem is a 0-1 knapsack problem, where the "item" is the user i's j-th request; the "weight" is r_{ij}; and the "value" is u_{ij}. Therefore, the more complex problem (7) is NP-hard. □

B. Feasibility

Theorem 2. *Our proposed approach produces a feasible solution to the problem* (7)*.*

Proof. If user i's j-th request is rejected, all constraints in problem (7) are satisfied. If user i's j-th request is accepted, then we have Constraint (7a) satisfied due to the domain of variable y_{ijt} defined in (8). Line 8 in Algorithm 1 indicates that if we set variable $x_{ij} = 1$, then user i's j-th request would be executed at time t^*, *i.e.*, $y_{ijt^*} = 1$. Therefore, Constraint (7b) holds. Line 7 in Algorithm 1 ensures that the total processed requests would not exceed the resource capacity and users' budgets. Then, Constraints (7c) and (7d) are satisfied. As a result, all constraints in problem (7) hold, indicating our proposed approach produces a feasible solution to problem (7). □

C. Time Complexity

Theorem 3. *Our proposed online algorithm runs in polynomial time.*

Proof. We consider the key steps in Algorithm 1. Let J represent the maximum number of requests generated by a single user. Then, Line 2 iterates at most IJ times. Line 3 iterates at most T times. Thus, overall, the Algorithm runs in $O(IJT)$. □

D. Competitive Ratio

Competitive ratio is a metric that characterizes the multiplicative gap between the objective function value achieved by an online solution and that achieved by the offline optimal solution. The online solution is generated by an online algorithm on the fly without knowing future information, and the offline optimal solution is computed by solving the problem optimally given all the input information over the entire time horizon. Next, we present the formal definition of the competitive ratio in the context of our problem.

Definition 1. ***Competitive Ratio****: Let OPT denote the offline optimal objective value of the problem* (7) *and its equivalent problem* (8)*. Let P^I be the objective value of problem* (8) *incurred by our online approach after processing all requests (either admitted or rejected). The competitive ratio of our approach is the upper bound on the ratio of the offline optimal objective value OPT to the objective value P^I achieved by Algorithm 1, i.e., the upper bound of OPT/P^I. The competitive ratio is greater than or equal to 1 in this definition.*

Theorem 4. *Our proposed online approach has the competitive ratio $\gamma = \epsilon(\alpha + \beta)$, where $\epsilon = 1 + \frac{2u_{max}r_{max}}{u_{min}r_{min}}$, $\alpha = \ln\theta$. The variable β controls the magnitude of growth in the updates of dual variable φ_i, as defined in* (11)*.*

Proof. To obtain the competitive ratio, we follow the roadmap below. The related lemmas and the details of this proof are placed in the Appendix (included in the supplemental material).

$$P^{(IJ)} \ge \frac{1}{\epsilon}\tilde{P}^{(IJ)} \tag{15}$$

$$\ge \frac{1}{\epsilon}\frac{1}{\alpha+\beta}D^{(IJ)} \tag{16}$$

$$\ge \frac{1}{\epsilon(\alpha+\beta)}OPT. \tag{17}$$

To assist the proof, we use $P^{(ij)}$ to denote the objective value of the original problem P (i.e., optimization problem (7)) after processing the j-th request of the i-th user. Similarly, we use $D^{(ij)}$ to represent the objective value of the dual problem D (i.e., optimization problem (9)) after processing the j-th request of the i-th user. Therefore, $P^{(IJ)}$ and $D^{(IJ)}$ denote the objective values of the original problem P and the dual problem D after processing all arrived requests, respectively. We also use $\tilde{P}^{(IJ)}$ to denote the objective value of the almost-feasible problem, which is introduced as an auxiliary problem to facilitate the proof; The inequality in (17) holds due to weak duality. see the Appendix for details. □

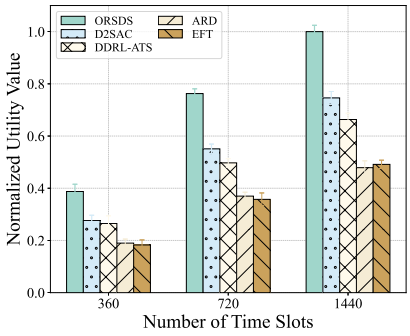


Fig. 7: Impact of Number of Time Slots.

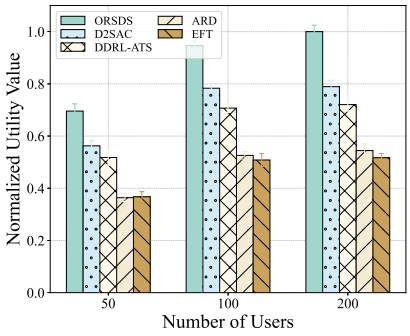


Fig. 8: Impact of Number of AIGC Users.

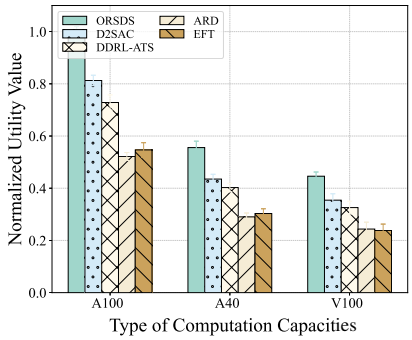


Fig. 9: Impact of Computation Capacities.

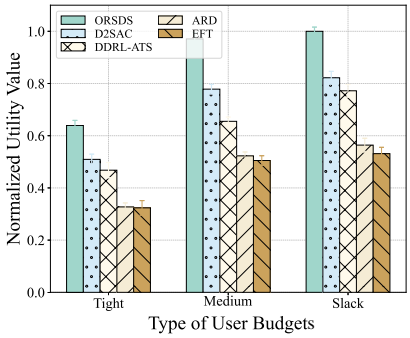


Fig. 10: Impact of Budgets.

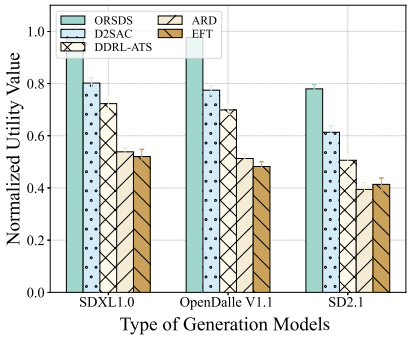


Fig. 11: Impact of Generation Models.

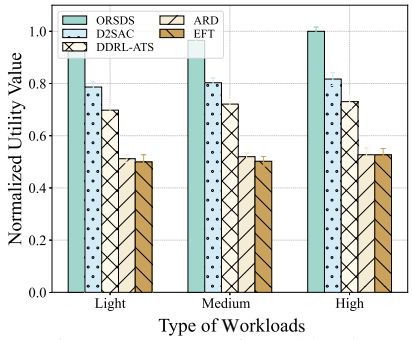


Fig. 12: Impact of Workloads.

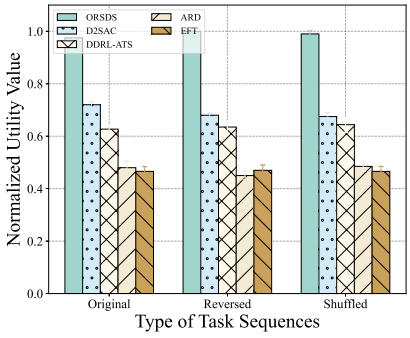


Fig. 13: Impact of Task Sequences.

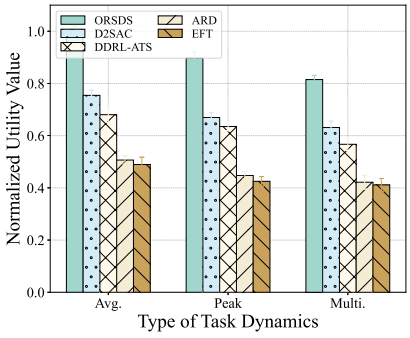


Fig. 14: Impact of Task Dynamics.

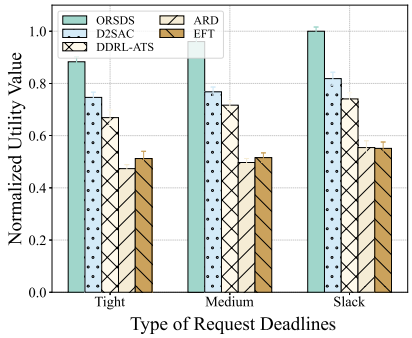


Fig. 15: Impact of Request Deadlines.

VII. Experimental Evaluations

A. Evaluation Settings

AIGC Users and Requests: We consider the scheduling system operating in a day, which has $T = 720$ consecutive time slots. Each time slot is of 2-minute length based on the average inference time of service model. During this period, a total of 50 to 200 AIGC users, with varying budgets submit their generation requests to the service provider. Each user has a specific budget ranging from 10 to 50, indicating the maximum number of requests that can be executed per day. AIGC requests of these users are generated in both real-world and synthesized patterns. Specifically, we extract prompts from DiffusionDB dataset [36], and request arrival time from Azure's public dataset [38]. DiffusionDB is a text-to-image prompt dataset containing 14 million prompts and hyperparameters specified by real users. We randomly select prompts and corresponding numbers of inference steps from the dataset to simulate user-generated requests. Azure's public dataset is a publicly available dataset provided by Microsoft Azure that includes anonymized traces of real-world cloud workloads. We analyze the dataset to collect statistical samples of the number of requests received over a specific time period, thereby simulating the volume of requests during that interval. We also generate multiple synthesized request arrival patterns based on the Possion process, which aligns with most existing work [14] [39]. In addition to its arrival time, each request is assigned a deadline, calculated as its arrival time plus a random number of time slot between 1 and 100.

AIGC Service Provider: The inference server is equipped with an Intel(R) Xeon(R) Gold 6348 CPU @ 2.60GHz and an NVIDIA RTX A100 (80GB) GPU. We deploy three types of widely used diffusion-based text-to-image models including *Stable-diffusion-xl-base-1.0* [35], *OpenDalleV1.1* [23], and *Stable-diffusion-2-1* [22] on the server. These models are invoked through Diffusers 0.32.1 based on Python 3.10 to gen-

TABLE II: Parameter values of quality perception models w.r.t. three widely-used AIGC models.

Parameters of Perception Model	SDXL [35]	OpenDalleV1.1 [23]	SD2.1 [22]
A_x	4	4	3
A_y	0.26	0.30	0.22
B_x	15	9	20
B_y	0.64	0.57	0.53

erate images. The hyperparameters of the generation models are set as follows: the value of batch size is set to 1, the value of guidance scale is set to 7.5. Key parameter values of the quality-aware models are obtained through real experiments, as shown in Table II. In our AIGC requests serving system, user prompts are sent to the inference server through FastAPI [40]. Additionally, the system capacity C is set to 10, which corresponds to the maximum number of requests the AIGC service provider can handle within a single time slot.

Baselines: We implement and compare our approach, **ORS-DS** (**O**nline **R**equest **S**cheduling for **D**iffusion-based **S**ervices), against the following alternatives:

- **D2SAC [16]**: D2SAC schedules AIGC requests based on the DRL method. It combines an AI-Generated optimal decision algorithm with Soft Actor-Critic method, improving overall service quality by enhancing the efficiency and effectiveness of AIGC Service Providers. However, D2SAC targets mainly on ASP selection issues, thus cannot be applied in our problem directly. We adjust the reward function according to our quality-aware model, partition the resources among multiple ASPs, and further train a new scheduling policy tailored for our problem. Additionally, we modify the action state to enhance the performance of D2SAC within our experimental settings.
- **DDRL-ATS [15]**: DDRL-ATS schedules AIGC requests using a reinforcement learning algorithm. It incorporates a diffusion model to prioritize requests and leverages a DRL framework to make scheduling decisions. Similarly, it cannot be directly applied to our problem due to differences in the experimental setup and optimization objectives. To address this, we modify the reward function to ensure it operates effectively in our scenario.
- **EFT (Earliest Finish Time)**: EFT schedules AIGC requests with the aim of minimizing the delay for each request as much as possible. For each incoming request, EFT attempts to find the nearest available time slot to execute it. If all feasible time slots before its deadline have already been occupied, EFT rejects the request. The scheduling priorities of requests are determined based on their order of arrival, with requests that arrive earlier being scheduled first.
- **ARD (Adaptive Random Discard)**: ARD introduces a random rejection mechanism when there are too many requests waiting to be scheduled in the current system. Incoming requests are randomly rejected when the system is under a heavy load, while accepted requests are scheduled to the time slot with the most abundant resources before their deadlines. ARD can effectively prevent the system from being overwhelmed by densely arriving low-reward requests.
- **Offline Optimal Algorithm**: We utilize the Gurobi solver [41] to compute the offline optimal solution. In this scenario, it is assumed that all the values of coefficients in problem (7) are known to the service provider in advance, and decisions are made by solving this optimization problem directly.

Each group of experiments is repeated 10 times, with the basic settings unchanged while the user's prompts and the random seed vary. We compute the mean and standard deviation for each group as our result.

B. Evaluation Results

Impact of System Scale: Figure 7 illustrates the impact of the number of time slots on normalized total utility value, *i.e.*, quality of generated images. As the number of time slots increases, users are able to submit more requests to the ASP, leading to a higher utility value as more requests are scheduled and executed. Across various experiments, Algorithm ORSDS consistently outperforms other methods. When the number of time slots is set to 1440, corresponding to two days, the performance improvements over D2SAC, DDRL-ATS, EFT and ARD are 25.3%, 32.7%, 50.8% and 52.1%, respectively.

In Figure 8, we vary the number of users submitting AIGC requests for service. As the number of users increases, the total number of requests grows, providing the ASP with more options to select and execute. Consequently, the total utility value of each scheduling algorithm increases. However, as the number of users continues to rise, the ASP's processing capacity gradually reaches its limit, preventing it from handling additional requests. Our algorithm slightly improve the overall utility value by selecting and executing more high-reward requests. In contrast, EFT schedules requests solely based on the earliest arrival, while ARD lacks the ability to assess the quality of requests. As a result, their total utility values remain almost the same.

Figure 9 describes the total utility value of five algorithms as the ASP resource capacity varies. The experiments are conducted using different GPU configurations, including the NVIDIA A100, A40, and Tesla V100 GPU. The computational power of the GPU affects the maximum number of inference steps that the scheduling system can execute within a single timeslot, which corresponds to the system's resource capacity. In our experiments, the A100 demonstrates significantly better performance than the A40 for diffusion model inference, while the A40 performs slightly better than the V100. Among these, Algorithm ORSDS consistently achieves the highest utility value and shows improvements of 22.1%, 24.9%, and 23.8% over the state-of-the-art method in the three respective cases.

Figure 10 presents the results as the user budget varies. In the "Slack" budget scenario, the user budget is set to 10, while it is increased to 25 and 50 in the "Medium" and "Tight" scenarios, respectively. As the budget increases, more requests

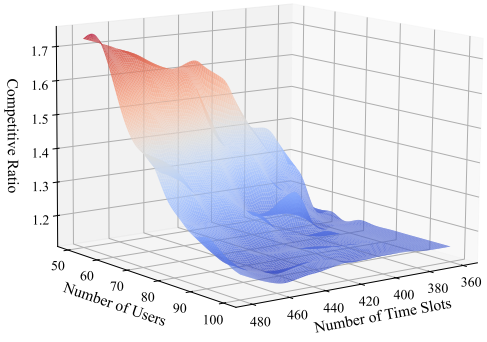


Fig. 16: Empirical Competitive Ratio.

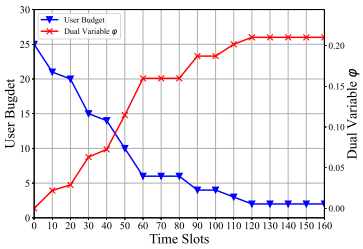


Fig. 17: Relationship between User Budget and dual variable φ.

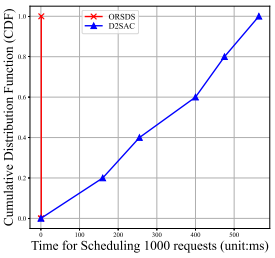


Fig. 18: Algorithm Runtime.

can be executed, resulting in a rise in the total utility value. However, this growth eventually stabilizes due to the system's computational capacity limitations. Due to the incorporation of targeted design for user budgets, our algorithm performs well in various scenarios, particularly when the user budget is "Tight". In Figure 11 we change the type of AIGC model deployed on the ASP, and the premeasured quality function thus changes as Table II. The overall generation capability of "SD2.1", as reflected by its parameters A_y and B_y, is inferior to that of the other two models. This results in a noticeable decline in the scheduling performance of all algorithms under these conditions. However, our algorithm effectively adapts to different text-to-image models and achieves improvements of 20.7%, 21.3% when the deployed model is changed to "OpenDalleV1.1" and "SD2.1", respectively.

Impact of Task Dynamics: Besides real-world traces extracted from Azure's public, we conduct the synthetic traces to simulate more complicated situations. In this scenario, we generate the number of requests for each user at the beginning of each time slot following the Poisson process. The "Light", "Medium", and "High" workloads in Figure 12 correspond to Poisson process intensities of 0.5, 1, and 1.5, respectively, which means the total arrival rates are 50, 100, and 150 for 100 users. The proposed ORSDS algorithm achieves greater performance improvements as workload increases.

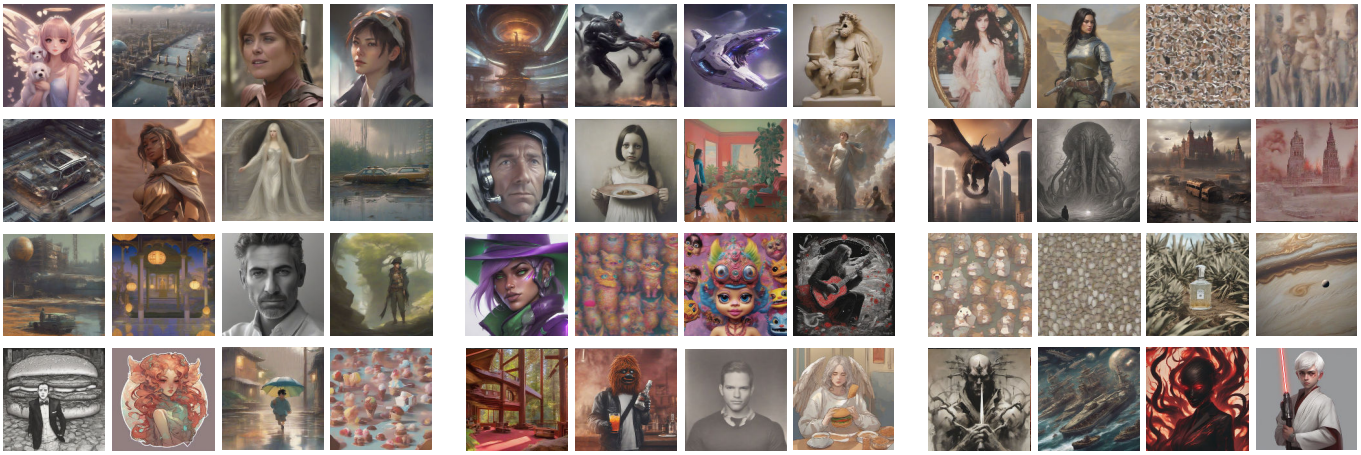
Figure 13 evaluates the impact of task arrival sequences on the normalized utility value. In this experiment, we simulate three different arrival orders for the same batch of requests. The label "Original" corresponds to the default arrival order of requests; "Reversed" indicates that the order is inverted; and "Shuffled" represents a randomly permuted order. Across all three experiments, our proposed algorithm consistently demonstrates a clear performance advantage over the baselines. Moreover, we observe that although different task arrival sequences lead to slight fluctuations in the overall utility, the impact remains limited, as the randomness in arrivals is averaged out over the long term. Instead, we find that the pattern of AIGC request arrivals, rather than their specific sequence, exerts a more substantial influence on the results, as shown in Figure 14. The label "Avg." denotes a constant average arrival rate; "Peak" represents a scenario with a single arrival peak; and "Multi." indicates multiple arrival peaks throughout the process. In the "Peak" scenario, a sudden surge in request volume poses considerable challenges to scheduling stability. Nevertheless, our algorithm maintains strong performance, outperforming the comparison algorithms by 25.2%, 29.2%, 52.6%, and 50.1%, respectively.

Figure 15 depicts the performance of the algorithms with different strategies to generate deadlines. We assume the request deadline to be its arrival time plus a random number of time slots between 1 and 100 (inclusive) in the "Medium" situation, indicating that the ASP must complete the request within this time slot. In the "Tight" scenario, the upper limit of the random range is reduced, while the lower limit is increased in the "Slack" scenario. As the deadline becomes tighter, the total utility value of five algorithms decreases, as many requests cannot be executed before the deadline. Notably, ORSDS consistently delivers the best performance across all scenarios.

Analysis of Budget: Figure 17 visualizes the relationship between user budget and dual variable φ in our proposed algorithm. With the user budget consuming gradually, φ keeps increasing, enabling the ASP to refuse requests from users which is not so valuable judged by quality-aware model. The "value" is related to the shadow price of the requests. For example, requests with excessively high inference deployment requirements but low expected returns will be deemed not valuable. This acts as a perfect threshold as we mentioned in V-A and is an internal reason accounting for why our ORSDS-based scheduling system performs well.

Algorithm Runtime: Figure 18 illustrates the runtime of ORSDS and D2SAC when scheduling 1000 requests. It shows that ORSDS has a much shorter algorithm runtime than D2SAC, with a latency of 1.78 ms versus 564 ms. This enables our proposed ORSDS algorithm to respond to AIGC requests instantly. In real-world scheduling scenarios, faster algorithm runtime brings significant benefits.

Competitive Ratio: Figure 16 evaluates the empirical competitive ratio, which is the ratio of the total utility value achieved by the offline optimum to that achieved by our online solution. We obtain the offline optimum via Gurobi solver [41] as introduced in VII-A. Results demonstrate that the proposed algorithm ORSDS achieves a good competitive ratio bounded by 1.8 with variations in the number of time slots and users. Furthermore, in most cases, the competitive ratio stabilizes at approximately 1.2, indicating strong performance.



(a) Generated images from ASP with ORSDS. (b) Generated images from ASP with D2SAC. (c) Generated images from ASP with EFT.

Fig. 19: Generated images from the ASPs adopting three different types of scheduling algorithms.

Visualization of Generated Images: To further evaluate the performance of five different scheduling algorithms, we track the generated results returned by each system. From these results, we randomly selected 16 images for visualization, as shown in Figure 19. Figure 19(a) illustrates the images generated from the ASP adopting our proposed ORSDS algorithm, where most of the results are clear and of high quality. Figure 19(b) shows the results of D2SAC, which are generally decent but include noticeably low quality outputs. Finally, Figure 19(c) presents the results of EFT. Due to its reliance on task arrival order and the absence of a quality perception mechanism, EFT produces outputs with inconsistent and often unreliable quality, including multiple low-quality images. Meanwhile, we calculate the average CLIP-IQA score of the images returned from the three ASPs. The results of ORSDS, D2SAC and EFT are 0.662, 0.583 and 0.465, respectively.

VIII. Conclusion

With the increasing number of diffusion-based AIGC requests offloaded to the edge cloud, developing an efficient scheduling algorithm is crucial for improving provisioned AIGC services. In this work, we investigate the online request scheduling problem for quality-aware diffusion-based AIGC services. We formulate a long-term optimization problem to maximize overall content generation quality of the AIGC ecosystem, subject to the constraints of limited edge resources and user budgets. To solve this problem, we conduct real experiments on multiple AIGC models to establish quality models, and carefully design the update of multiple dual variables to flexibly control the consumption of edge resources and user budgets. Meanwhile, we theoretically analyze the optimization problem and the proposed method, including the NP-hardness of the problem, the feasibility of our approach, the polynomial-time complexity of our algorithm, as well as a competitive ratio against the offline optimum. Extensive real-world trace-driven experiments are conducted to validate our proposed method.

References

[1] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, and L. Yu, "A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from Gan to Chatgpt," *arXiv preprint arXiv:2303.04226*, 2023.

[2] OpenAI: Gpt-4 Technical Report. [Online]. Available: https://cdn.openai.com/papers/gpt-4.pdf

[3] Z. Xue, G. Song, Q. Guo, B. Liu, Z. Zong, Y. Liu, and P. Luo, "Raphael: Text-to-image Generation via Large Mixture of Diffusion Paths," *Advances in Neural Information Processing Systems*, 2024.

[4] J. Wu, W. Gan, Z. Chen, S. Wan, and H. Lin, "Ai-Generated Content (AIGC): A Survey," *arXiv preprint arXiv:2304.06632*, 2023.

[5] H. Du, Z. Li, D. Niyato, J. Kang, Z. Xiong, X. S. Shen, and D. I. Kim, "Enabling AI-Generated Content Services in Wireless Edge Networks," *IEEE Wireless Communications*, 2024.

[6] Y. Liu, H. Du, D. Niyato, J. Kang, Z. Xiong, A. Jamalipour, and X. Shen, "ProSecutor: Protecting Mobile AIGC Services on Two-Layer Blockchain via Reputation and Contract Theoretic Approaches," *IEEE Transactions on Mobile Computing*, 2024.

[7] Y. He, L. Ma, R. Zhou, C. Huang, and Z. Li, "Online Task Allocation in Mobile Cloud Computing with Budget Constraints," *Computer Networks*, vol. 151, pp. 42–51, 2019.

[8] N. Buchbinder, K. Jain, and J. Naor, "Online Primal-Dual Algorithms for Maximizing Ad-Auctions Revenue," in *European Symposium on Algorithms*. Springer, 2007, pp. 253–264.

[9] W. Shi, L. Zhang, C. Wu, Z. Li, and F. C. Lau, "An Online Auction Framework for Dynamic Resource Provisioning in Cloud Computing," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 1, pp. 71–83, 2014.

[10] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, and Z. Mao, "Sparks of GPTs in Edge Intelligence for Metaverse: Caching and Inference for Mobile AIGC Services," *arXiv preprint arXiv:2304.08782*, 2023.

[11] G. Liu, H. Du, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, and X. Shen, "Semantic Communications for Artificial Intelligence Generated Content (AIGC) toward Effective Content Creation," *IEEE Network*, 2024.

[12] H. Du, R. Zhang, D. Niyato, J. Kang, Z. Xiong, D. I. Kim, X. Shen, and H. V. Poor, "Exploring Collaborative Distributed Diffusion-based AI-generated Content (AIGC) in Wireless Networks," *Ieee network*, vol. 38, no. 3, pp. 178–186, 2023.

[13] Y. Zheng, L. Jiao, Y. Xu, B. An, X. Wang, and Z. Li, "Scheduling Generative-AI DAGs with Model Serving in Data Centers," in *IEEE/ACM International Symposium on Quality of Service*, 2024.

[14] X. Lyu, S. Rani, and Y. Feng, "Optimizing AIGC Service Provider Selection Based on Deep Q-Network for Edge-enabled Healthcare Consumer Electronics Systems," *IEEE Transactions on Consumer Electronics*, pp. 1–1, 2024.

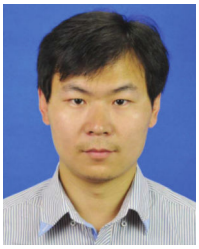
[15] C. Xu, "Diffusion-based Task Scheduling for Efficient AI-Generated Content in Edge Networks," in *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 2024, pp. 333–334.

[16] H. Du, Z. Li, D. Niyato, J. Kang, Z. Xiong, H. Huang, and S. Mao, "Diffusion-based Reinforcement Learning for Edge-enabled AI-Generated Content Services," *IEEE Transactions on Mobile Computing*, 2024.
[17] W. Feng, R. Zhang, Y. Zhu, C. Wang, C. Sun, X. Zhu, X. Li, and T. Taleb, "Exploring Collaborative Diffusion Model Inferring for AIGC-enabled Edge Services," *IEEE Transactions on Cognitive Communications and Networking*, 2024.
[18] T. Deng, D. Chen, J. Jia, M. Dong, K. Ota, Z. Yu, and D. Yuan, "Optimizing resource allocation and request routing for ai-generated content (AIGC) services in mobile edge networks with cell coupling," *IEEE Transactions on Vehicular Technology*, 2024.
[19] R. Zhou, Z. Li, C. Wu, and Z. Huang, "An Efficient Cloud Market Mechanism for Computing Jobs with Soft Deadlines," *IEEE/ACM Transactions on networking*, vol. 25, no. 2, pp. 793–805, 2016.
[20] F. Hoseiny, S. Azizi, M. Shojafar, F. Ahmadiazar, and R. Tafazolli, "PGA: A Priority-aware Genetic Algorithm for Task Scheduling in Heterogeneous Fog-Cloud Computing," in *IEEE conference on computer communications workshops*. IEEE, 2021, pp. 1–6.
[21] Y. Sun, C. Lin, J. Ren, P. Wang, L. Wang, G. Wu, and Q. Zhang, "Subset Selection for Hybrid Task Scheduling with General Cost Constraints," in *IEEE Conference on Computer Communications*. IEEE, 2022, pp. 790–799.
[22] "Stable-diffusion-2-1," 2023. [Online]. Available: https://huggingface.co/stabilityai/stable-diffusion-2-1
[23] "OpenDalleV1.1," 2024. [Online]. Available: https://huggingface.co/dataautogpt3/OpenDalleV1.1
[24] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.
[25] P. Dhariwal and A. Nichol, "Diffusion Models Beat Gans on Image Synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.
[26] A. Q. Nichol and P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," in *International conference on machine learning*. PMLR, 2021, pp. 8162–8171.
[27] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, and L. Van Gool, "Repaint: Inpainting Using Denoising Diffusion Probabilistic Models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 461–11 471.
[28] S. Cheng, H. Hu, X. Zhang, and Z. Guo, "Rebuffering but not Suffering: Exploring Continuous-Time Quantitative QoE by User's Exiting Behaviors," in *IEEE Conference on Computer Communications*, 2023.
[29] A. Mittal, A. K. Moorthy, and A. C. Bovik, "No-reference Image Quality Assessment in the Spatial Domain," *IEEE Transactions on image processing*, vol. 21, no. 12, pp. 4695–4708, 2012.
[30] J. Ke, Q. Wang, Y. Wang, P. Milanfar, and F. Yang, "Musiq: Multi-scale Image Quality Transformer," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 5148–5157.
[31] M. Heusel, H. Ramsauer, T. Unterthiner, and S. Nessler, "Gans Trained by A Two Time-scale Update Rule Converge to a Local Nash Equilibrium," *Advances in neural information processing systems*, 2017.
[32] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved Techniques for Training Gans," *Advances in neural information processing systems*, vol. 29, 2016.
[33] X. Zhang, Y. Zhang, W. Yu, L. Nie, N. Jiang, and J. Gong, "Qs-hyper: A Quality-sensitive Hyper Network for the No-reference Image Quality Assessment," in *Neural Information Processing: 28th International Conference (ICONIP)*. Springer, 2021, pp. 311–322.
[34] J. Wang, K. C. Chan, and C. C. Loy, "Exploring Clip for Assessing the Look and Feel of Images," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 2, 2023, pp. 2555–2563.
[35] "Stable-diffusion-xl-base-1.0," 2023. [Online]. Available: https://huggingface.co/stabilityai/stable-diffusion-xl-base-1.0
[36] Z. J. Wang, E. Montoya, D. Munechika, H. Yang, B. Hoover, and D. H. Chau, "DiffusionDB: A Large-Scale Prompt Gallery Dataset for Text-to-Image Generative Models," *arXiv:2210.14896 [cs]*, 2022. [Online]. Available: https://arxiv.org/abs/2210.14896
[37] T. Gal, "Shadow Prices and Sensitivity Analysis in Linear Programming under Degeneracy: State-of-the-Art-Survey," *Operations-Research-Spektrum*, vol. 8, no. 2, pp. 59–71, 1986.
[38] Azure Inference Dataset. Accessed: 2024-12-25. [Online]. Available: https://github.com/Azure/AzurePublicDataset/blob/master/AzureLLMInferenceDataset2023.md
[39] V. Srivatsa, Z. He, R. Abhyankar, D. Li, and Y. Zhang, "Preble: Efficient Distributed Prompt Scheduling for LLM Serving," in *International conference on learning representations*, 2024.
[40] FastAPI. [Online]. Available: https://fastapi.tiangolo.com/
[41] T. Achterberg, "What's New in Gurobi 9.0," *Webinar Talk url: https://www. gurobi. com/wp-content/uploads/2019/12/Gurobi-90-Overview-Webinar-Slides-1. pdf*, vol. 5, no. 9, pp. 97–113, 2019.

Han Yang received his B.S. degree in Electronic Engineering from Tsinghua University, in 2023. He is currently pursuing his M.S. degree at the Institute for Network Sciences and Cyberspace, Tsinghua University. His research interest focuses on the request scheduling problems and scheduling algorithms related to advanced AIGC service.

Ying Zheng received her Ph.D. degree in Computer Science from Fudan University in 2024. She is currently a postdoctoral researcher at Inria (National Institute for Research in Digital Science and Technology), France. Her research interests include AI infrastructure, and machine learning for optimization. She has published papers in INFOCOM, JSAC, IWQoS, and ICPP.

Lei Jiao received his Ph.D in CS from the University of Göttingen, Germany. He is with the University of Oregon, and was a technical staff at Nokia Bell Labs, Ireland. He researches AI infrastructures, cloud/edge networks, energy systems, cybersecurity and multimedia. He has published 80+ papers in journals such as IEEE JSAC, IEEE/ACM ToN, IEEE TMC, and IEEE TPDS, and conferences such as INFOCOM, MOBIHOC, ICDCS, SECON, and ICNP. He is a U.S. National Science Foundation CAREER awardee, and a recipient of the Ripple Faculty Fellowship, the Alcatel Lucent Bell Labs UK and Ireland Recognition Award, and Best Paper Awards of IEEE CNS 2019 and IEEE LANMAN 2013.



Yuedong Xu received B.S. degree from Anhui University, MSc. Huazhong University of Science & Technology, and Ph.D from The Chinese University of Hong Kong. From 2009 to 2012, he was a Post-Doctoral Researcher with INRIA Sophia Antipolis and Universite d'Avignon, France. He is currently a Professor with the School of Information Science and Technology, Fudan University. His research interests include performance evaluation, optimization, security, data analytics and economic analysis of communication networks, and mobile computing.

Zongpeng Li received his BSc in CS from Tsinghua University in 1999 and his Ph.D from University of Toronto in 2005. His research interests include computer networks, network coding, network algorithms, and cyber security. He received the Outstanding Young Computer Science Researcher Prize from the Canadian Association of Computer Science, and the Research Excellence Award from the Faculty of Science, University of Calgary. He is a senior member of the IEEE.