Efficient Multi-User Offloading of Personalized Diffusion Models: A DRL-Convex Hybrid Solution

Wanting Yang, Zehui Xiong, *Senior Member, IEEE*, Song Guo, *Fellow, IEEE*, Shiwen Mao, *Fellow, IEEE*, Dong In Kim, *Fellow, IEEE*, and Merouane Debbah, *Fellow, IEEE*

***Abstract*—Generative diffusion models like Stable Diffusion are at the forefront of the thriving field of generative models today, celebrated for their robust training methodologies and high-quality photorealistic generation capabilities. These models excel in producing rich content, establishing them as essential tools in the industry. Building on this foundation, the field has seen the rise of *personalized content synthesis* as a particularly exciting application. However, the large model sizes and iterative nature of inference make it difficult to deploy personalized diffusion models broadly on local devices with heterogeneous computational power. To address this, we propose a novel framework for efficient multi-user offloading of personalized diffusion models. This framework accommodates a variable number of users, each with different computational capabilities, and adapts to the fluctuating computational resources available on edge servers. To enhance computational efficiency and alleviate the storage burden on edge servers, we propose a tailored multi-user hybrid inference approach. This method splits the inference process for each user into two phases, with an optimizable split point. Initially, a cluster-wide model processes low-level semantic information for each user's prompt using batching techniques. Subsequently, users employ their personalized models to refine these details during the later phase of inference. Given the constraints on edge server computational resources and users' preferences for low latency and high accuracy, we model the joint optimization of each user's offloading request handling and split point as an extension of the Generalized Quadratic Assignment Problem (GQAP). Our objective is to maximize a comprehensive metric that balances both latency and accuracy across all users. To solve this NP-hard problem, we transform the GQAP into an adaptive decision sequence, model it as a Markov decision process, and develop a hybrid solution combining deep reinforcement learning with convex optimization techniques. Simulation results validate the effectiveness of our framework, demonstrating superior optimality and low complexity compared to traditional methods.**

***Index Terms*—Diffusion model, edge offloading, generalized quadratic assignment problem, DRL, AIGC service, hybrid inference.**

Received 24 November 2024; revised 2 March 2025; accepted 7 April 2025. Date of publication 14 April 2025; date of current version 6 August 2025. The work was supported in part by the National Research Foundation, Singapore and in part by Infocomm Media Development Authority under its Future Communications Research & Development Programme. The work was also supported in part by the SUTD-ZJU IDEA under Grant SUTD-ZJU (VP) 202102, in part by the Ministry of Education, Singapore, under its Academic Research Fund Tier 2 under Grant MOE-T2EP20221-0017, in part by SMU-SUTD Joint under Grant 22-SIS-SMU-048, and in part by SUTD Kickstarter Initiative under Grant SKI 20210204. This work was supported in part by the MSIT (Ministry of Science and ICT), Korea, under the ICT Creative Consilience program under Grant IITP-2020-0-01821, and in part by the ITRC support program under Grant IITP-2023-RS-2023-00258639 supervised by the IITP (Institute for ICT Planning & Evaluation). This work was supported in part by the Hong Kong RGC General Research Fund under Grant 152244/21E, Grant 152169/22E, Grant 152228/23E, and Grant 162161/24E, in part by Research Impact Fund under Grant R5011-23F and Grant R5060-19, in part by Collaborative Research Fund under Grant C1042-23GF, in part by NSFC/RGC Collaborative Research Scheme under Grant CRS_HKUST602/24, in part by Areas of Excellence Scheme under Grant AoE/E-601/22-R, and in part by the InnoHK (HKGAI). The work of Shiwen Mao was supported by the NSF under Grant CNS-2107190 and Grant CNS-2148382. Recommended for acceptance by N. Pappas. *(Corresponding author: Zehui Xiong.)*

Wanting Yang is with the Pillar of Engineering Systems and Design, Singapore University of Technology and Design,, Singapore 487372 (e-mail: wanting_yang@sutd.edu.sg).

Zehui Xiong is with the Pillar of Information Systems Technology and Design, Singapore University of Technology and Design,, Singapore 487372 (e-mail: zehui_xiong@sutd.edu.sg).

Song Guo is with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China (e-mail: songguo@cse.ust.hk).

Shiwen Mao is with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36830 USA (e-mail: smao@ieee.org).

Dong In Kim is with the Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea (e-mail: dongin@skku.edu).

Merouane Debbah is with the KU 6G Research Center, Department of Computer and Information Engineering, Khalifa University, Abu Dhabi 127788, UAE, and also with CentraleSupelec, University Paris-Saclay, 91192 Gif-sur-Yvette, France (e-mail: merouane.debbah@ku.ac.ae).

All related code, datasets, and fine-tuned models are available at https://github.com/wty2011jl/E-MOPDM.

Digital Object Identifier 10.1109/TMC.2025.3560582

I. Introduction

The remarkable progress of generative models has ushered in a new era of artificial intelligence technology. Notable advancements include ChatGPT in natural language processing, demonstrating capabilities on par with human experts in a wide range of applications. Similarly, Stable Diffusion Models (SDMs) [1], developed based on Denoising Diffusion Probabilistic Models [2], have gained comparable prominence in computer vision. The impressive generative capabilities and the training robustness of SDMs have established them as a foundational tool in numerous applications, such as *super-resolution*, *image editing*, *text-to-image* and *inpainting*, etc. [3]. They are also pivotal in empowering industries such as telecommunications [4] and medical [5] sectors. Among these advanced applications, personalized content synthesis (PCS) stands out as a particularly important and eagerly anticipated branch of research.

The customizable features of PCS enable users to generate images tailored to objects, styles, faces, and other high-level semantic details. For example, this capability allows users to place themselves in virtual scenes like historic landmarks or future cities, or recreate rare pet behaviors, such as a cat leaping gracefully. As a result, PCS has garnered significant public interest, prompting a surge in research and development efforts within this area. Since the release of *DreamBooth* and *Textual Inversion* in August 2022, the field has rapidly expanded, with over 100 methods developed in a short time. [6]. However, especially within the PCS field, rising privacy concerns have become particularly pronounced due to the personal nature of the content being generated. Additionally, the increasing number of users has put excessive pressure on cloud resources. Consequently, there is a strong demand for on-device deployment of SDMs to address these issues. Yet, these models often come with large model sizes and substantial computational requirements. In this sense, the slow and iterative sampling process during inference limits the widespread deployment of personalized SDMs on local devices, as they must adapt to the diverse computational capabilities of user devices.

Recently, research aimed at enhancing inference speed in diffusion models has been organized into three main categories. The first focuses on *sampling step reduction*, which is best known for its implementation in Denoising diffusion implicit models (DDIMs) [7]. The second approach is based on *model compression* [8], [9], [10]. The third category achieves *hybrid inference* through edge offloading [11], [12]. (More details of these techniques are discussed in Section II.) Among these approaches, the DDIM has been integrated into SDMs. However, generating high-quality and detail-rich images still requires hundreds of denoising steps. In recent work [13], *ByteDance Inc.* combines the aforementioned model compression and both hybrid inference methods for SDMs. The inference process begins with a smaller model on the user side, while the latter denoising steps, which require additional detail, are performed by a large-scale model on the cloud or edge servers. The simulation results confirm the promising gains brought by this *composite hybrid inference* approach. However, unlike shared foundation models, personalized diffusion models are unique to each user. Extending this to a practical multi-user scenario, replicating each personalized model on the edge server for independent inference would place immense storage and computational burdens on the servers used for offloading. This approach deviates from the trend toward on-device applications and could potentially exceed the capabilities of even cloud infrastructure.

To overcome this dilemma and facilitate the widespread adoption of personalized SDMs, this work focuses on a balanced solution for multi-user personalized SDMs. Our goal extends beyond optimizing specific inference algorithms. Instead, we intend to seamlessly integrate inference algorithm optimizations with real-world resource management, relying on the typical edge-device collaborative network framework. Specifically, we propose a novel framework for efficient multi-user offloading of personalized diffusion models. This framework addresses scenarios with a variable number of users, diverse user computational capabilities, and fluctuating edge server resources. To minimize redundancy and alleviate storage burdens, we employ a cluster-wide model that captures common personalized features of the cluster's users through fine-tuning, serving as a shared offloading model. Furthermore, we enhance computational efficiency by enabling batch processing of offloading tasks for multiple users. Nevertheless, these improvements also present two new challenges.

- The hybrid inference approach, which combines the offloaded inference by the cluster-wide model with local inference by the personalized model. While offloading reduces inference latency, it also introduces a degree of personalization loss. Thus, a trade-off between latency reduction and accuracy of personalized feature representation must be carefully considered when optimizing offloading steps.
- Existing approaches often overlook the server's parallel inference capabilities and assume that total latency is proportional to the number of tasks. Unlike these methods, the batching technique presents a unique challenge. As batch size increases, it leads to longer inference latency for each denoising step, thereby exacerbating the interdependence between individual user offloading decisions and overall system performance.

To address the aforementioned challenges, we innovatively develop a comprehensive and precise optimization metric. Using this metric as a guide, we formulate the efficient multi-user offloading for personalized diffusion models as an extension of the Generalized Quadratic Assignment Problem (GQAP). Additionally, we design a low-complexity solution specifically for this extended version of GQAP. The key contributions of our approach are summarized as follows:

- We propose an efficient multi-user hybrid inference manner. In this framework, a cluster-wide SDMs, which is trained to capture common features of personalized demands among users within the cluster, is located at the edge server. Meanwhile, the personalized model of each user is deployed at the local individually. The inference process for each user is split into two phases with an *optimizable split point*. For users participating in offloading, the initial phase of inference is performed on the cluster-wide model using batching techniques. This approach synchronously generates low-level semantic and contour information tailored to each user's prompt. The intermediate results are subsequently returned to each user, who then applies a personalized SDM to add finer details in the later stages of the generation process.
- We propose a tailored metric that balances latency and accuracy of personalized feature, interconnected through a weighting parameter. This parameter captures each user's preference between the two factors and can be customized individually to reflect their specific requirements. Based on rigorous mathematical analysis, we provide the range for this parameter along with insights into the corresponding effects. To quantify accuracy in PCS, we define a metric called Personalized Accuracy Index (PAI), which jointly considers the traditional accuracy, measured by the CLIP, and the variation in final generated images for different users given the same prompt, assessed by LPIPS. We train a cluster-wide model and multiple personalized models using DreamBooth and establish a unified mathematical model describing the relationship between PAI and the split point based on empirical simulations.
- We formulate the joint optimization of offloading request handling and each user's individual split point as an

TABLE I
SUMMARY OF MAIN NOTATIONS

System Setting-Related Symbols	
G	The number of available GPUs at edge
$\mathcal{I}$	The set of user sending requests
$B_{\max}$	The maximum batchsize supported by the edge server
N	The total denoising steps during inference
$\hat{N}$	The maximum number of offloaded denoising steps
$W_{\max}$	The total bandwidth shared by the granted users
s^{p}	The data size of prompt
s^{m}	The data size of intermediate data
η	The spectral efficiency over wireless link
Metric and Decision-Related Symbols	
ℓ^{R}	The round-trip time (waiting latency)
ℓ^{T}	The transmission latency
ℓ^{E}	The computing latency at edge
ℓ^{L}	The transmission latency at local
L	The end-to-end latency
n^*	The optimal split point
$\mathbf{x}$	The results of request handling
b_{g}	The number of granted user for offloading
α_i	The customized emphasis parameter between L and PAI
$\mathscr{L}_e(\cdot)$	The function of inference latency per denoising step at edge on b_{g} and G
$\mathscr{F}(\cdot)$	The function of personalized accuracy index metric on n^*

extended GQAP, aiming to maximize overall performance using the tailored metric as a guiding principle. To meet the real-time decision-making requirements, we are, to the best of our knowledge, the first to apply Deep Reinforcement Learning (DRL) to solve GQAP, proposing a DRL-convex hybrid solution. Specifically, we introduce a novel paradigm that transforms GQAP into a sequential decision-making process and maps it into a Markov Decision Process (MDP) model. Meanwhile, we determine the optimal split point for each user based on convex optimization theory and integrate it into the environment of the MDP. We compare this algorithm with classical algorithms for solving GQAP, such as Branch & Bound and heuristic algorithms, demonstrating its optimality and low complexity, $O(n)$, regarding the number of users.

The rest of this paper is organized as follows. In Section II, we review research in hybrid inference for SDMs and the existing solutions relevant to GQAP. Then, in Section III, we introduce the system overview, and relevant models. Then, we present the problem formulation in Section IV and the DRL-convex hybrid solution is described in Section V. An experimental evaluation is presented in Section VI, followed by the conclusion and future directions in Section VII. For the sake of convenience, the main notations in this work have been summarized in Table I.

II. Related Works

A. Inference Latency Reduction in Diffusion Models

As discussed in Section I, research aimed at overcoming these challenges can be organized into three main directions. The first direction, Sampling Step Reduction, is exemplified by Denoising Diffusion Implicit Models (DDIMs) [7], which reparameterize the diffusion process into a non-Markovian chain. This enables sampling in 50 - 200 steps, or even fewer, instead of the typical 1,000, but at the cost of reduced high-frequency detail generation and increased output variability. Efforts to balance speed with fidelity include adaptive step scheduling and noise-aware sampling, but challenges remain. The second direction, Model Compression, includes architectural redesign, pruning, and quantization to streamline diffusion architectures while preserving semantic coherence. Architectural redesigns, such as replacing U-Net backbones with vision transformers in U-ViT [14], achieve significant parameter reductions while maintaining competitive FID scores. Structured pruning strategies [9] remove redundant elements to reduce computational demands, and quantization techniques demonstrated by Q-Diffusion [10] employ precision adjustments to maintain performance. Despite advancements, compressed models struggle with complex compositions and rare concepts, often requiring task-specific adjustments to avoid semantic drift. The third category, *hybrid inference*, includes (1) combining models of different sizes to offset quality degradation from exclusive reliance on compressed models [11], and (2) leveraging edge server's greater computing power to offload part of the denoising steps, enabling collaborative inference and reducing overall latency [12]. A detailed discussion on hybrid inference will follow in Section II-B.

B. Hybrid Inference for Diffusion Models

Since the remarkable success of diffusion models in generative tasks, research on latency optimization for inference processes has gained momentum. As discussed in Section I, this research initially focused on reducing denoising steps and compressing the model. However, both approaches inevitably lead to some degree of performance degradation; the greater the reduction in inference latency, the more noticeable the performance loss. Beyond these two approaches, hybrid inference has been proposed, offering flexible combinations with the above methods.

In 2022, *NVIDIA Corporation* first proposes the eDiff-I framework, where the inference process is divided into multiple stages. Each stage corresponds to a distinct model, focusing on different generative functions to enhance overall inference performance [15]. Then, the authors in [16] design OMS-DPM, a predictor-based search algorithm, to determine the optimal model schedule given a specific generation time budget and a set of pre-trained models. Recently, in [17], the authors present a new framework called DDSM, which employs a spectrum of neural networks whose sizes are adapted according to the importance of each generative step. In addition to the model's size and functionality, the authors in [12] expand the concept of hybrid inference to incorporate computational heterogeneity within the inference process. Here, the inference process is split, with one part completed by a more powerful edge server and the latter part handled locally by the user device, connected via a wireless transmission link to reduce inference latency. Additionally, they propose an interesting idea: users with similar prompts can share the initial portion of the inference process on the edge server, reducing the edge server's computational load. However, this inevitably comes at the cost of lower image generation quality. To address this, the authors in [18] further optimize the split point under this setting. Although the framework's effectiveness is applicable when prompts from different users are highly similar, in practice, user prompts are often diverse. Therefore, this framework's use cases are quite limited.

In addition, ByteDance further extends hybrid inference to simultaneously incorporate both model heterogeneity and computational heterogeneity, as introduced in [13]. However, they only consider the single-user inference scenario and do not address the inevitable challenges in real-world applications, where cloud server resources available to users are limited. Consequently, they do not explore the necessary handling of offloading requests or the optimization of the split point based on personalized user requirements under such conditions. Overall, optimization of diffusion model inference for real-world applications remains in a very early stage.

C. Generalized Quadratic Assignment Problem

Combinatorial optimization problems (COPs) are widely present in network resource optimization, including the *traveling salesman problem* [19], *knapsack problem* [20], and *multi-armed bandit problem* [21], etc. Among them, quadratic assignment problem (QAP) is considered as one of the most difficult to solve in combinatorial optimization [22]. In addition, various variants of this problem, such as *Quadratic Bottleneck Assignment Problem*, Biquadratic QAP, quadratic semi-assignment problem and the relatively new GQAP, have continuously emerged, attracting significant attention over the past decades. A common approach is to linearize the quadratic terms using linearization techniques, enabling the problems to be solved with general-purpose mixed integer linear programming solvers. Two widely used techniques are standard linearization [23] and the reformulation-linearization technique (RLT) [24]. Meanwhile, the authors in [25] further integrate Lagrangean relaxation procedure over RLT, then combine with the classic Branch & Bound scheme. Nevertheless, the quadratic form of its objective function makes it challenging to find even an approximate optimal solution within limited time, especially when the problem size is slightly larger. To this end, the authors in [22] propose a heuristic algorithm, a parallel iterated tabu search algorithm. Although it outperforms traditional integer linear programming solvers in terms of performance, its high computational complexity makes it challenging to apply in real-time optimization scenarios.

Meanwhile, the success of DRL in solving increasingly complex problems over the past decade has also inspired a wave of research on using DRL to tackle COPs with low complexity. Notable achievements have been made in solving problems such as the TSP, Vehicle Routing Problem, and Maximum Cut [26], [27], [28]. The basic idea is to transform the optimization of a high-dimensional variable into a decision sequence, thus formulating it as an MDP model based on the DRL paradigm. However, unlike ordering problems such as the TSP, the QAP problem hardly permits such a transformation. Thus using DRL to solve QAP has received relatively little attention. To our knowledge, only two studies have explored this approach. In [29], the authors discuss the challenges of generalizing existing frameworks for TSP on QAP. Moreover, in [30] the authors leverage a novel double pointer network, which alternates between selecting a location for the next facility and choosing a facility for the previously selected location. Although the effectiveness of this paradigm has been demonstrated, it is limited to standard QAP that requires an equal number of facilities and locations. Consequently, the algorithm's applicability is restricted. In summary, a low-complexity algorithm for addressing the GQAP remains an open research area.

III. System Model and Pre-Deployment

A. System Overview

In this work, we consider a single-cell scenario with a cluster of users running the *customized* on-device applications based on SDMs, as shown in Fig. 3. Meanwhile, for users within a cluster, we assume their PCS tasks share similarities, such as personalized dog object generation, personalized female face generation, or similar personalized style. Moreover, the devices of the users, such as desktops, laptops, and smart mobile devices, are characterized by diverse computational power. To reduce computational latency and facilitate the application proliferation, users can access edge offloading services. However, user-specific model deployment at the edge leads to considerable storage overhead. To this end, an efficient offloading manner is considered in this work. Specifically, these customized on-device applications are assumed to share a cluster-wide model hosted on the edge server. This model, derived through cluster-oriented fine-tuning, enables efficient parallel processing of tasks with batching techniques. A feasible pre-deployment approach is detailed in Section III-B.

To maximize overall system efficiency and performance, we adopt a periodic & centralized offloading mechanism. We assume that the offloading strategy is executed at fixed intervals of duration Δ, each consisting of K time slots, where each time slot is indexed by k. Moreover, we denote the set of users sending requests during Δ as $\mathcal{I}$, with each user indexed by i. A user $i \in \mathcal{I}$ can send an offloading request at any time slot k, but must wait $(K-k)$ time slots to receive the server's feedback. This time interval is referred to as round-trip time (RTT), denoted by ℓ^{R}. Moreover, we denote the number of available GPUs for the current user set $\mathcal{I}$ by G, which may vary over time. We define that the offloading decisions made by the edge server for user requests include two options: *grant* and *deny* [20]. For the grant decision, the *number of offloaded denoising steps* is required to be identified at the same time. All the decisions for the users in $\mathcal{I}$ are jointly determined by four main factors: 1) the computational

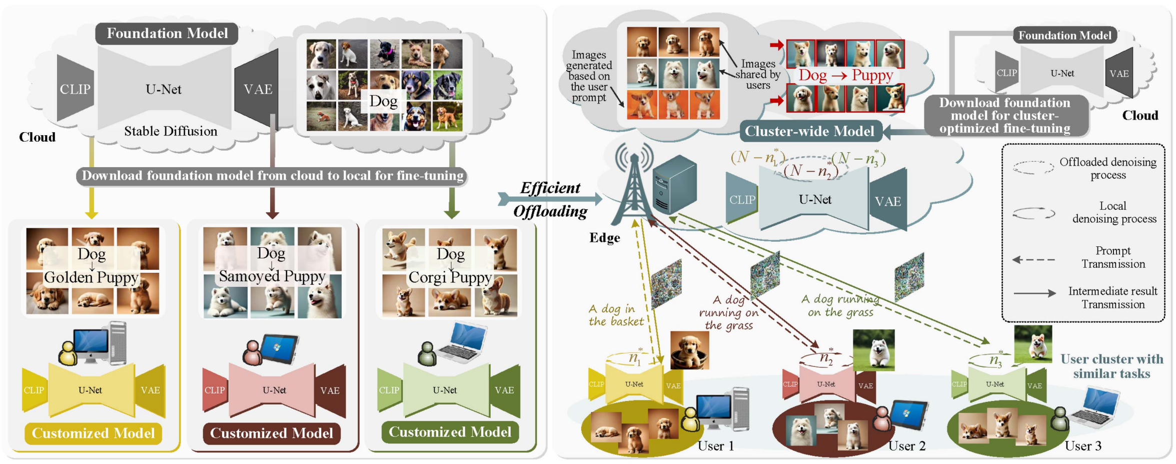


Fig. 1. Illustration of system overview and pre-deployment approach.

capacity of the users' devices; 2) the preferences for personalized accuracy and latency; 3) the time of the request; and 4) the current available resources at the edge. As with any trade-off, the degree of inference offloading inversely impacts the extent of degradation in meeting users' personalized requirements. In this sense, these factors are considered to achieve a desired trade-off between latency and the final personalized accuracy for all users. The metrics about latency and personalized accuracy are specified in Section III-D.

Once the offloading strategy is determined, all denoising steps for users who are denied are performed locally on their devices. For users who are granted, they first need to transmit their *individual* prompts to the edge server. The edge server, based on the offloading strategy results for each user, performs the user-specific number of denoising steps at the edge for each user's prompt, respectively. The intermediate results are then fedback to the users to complete the remaining offloading locally. (The hybrid inference model is detailed in Section III-C.) For the uplink and downlink transmission involved in the process, we assume that the spectral efficiency η (or data rate per unit bandwidth) for user i is fixed, with a given modulation and coding scheme. Given that the transmission latency is remarkably lower than the computational latency, we simplify the bandwidth allocation. We assume that during the transmission process, all users granted for offloading equally share the system's reserved maximum bandwidth $W_{\max}$.

B. Pre-Deployment Approach

Subsequently, we introduce a pre-deployment approach for the considered multi-user scenario as stated in Section III-A. For illustration purposes, we consider the SDM-based on-device applications with *personalized object generation*. This approach, however, is equally applicable to personalization tasks involving *facial features*, *style customization*, etc., supporting a variety of emerging diffusion-based applications such as extended reality, advertising, and remote education.

For example, as shown in Fig. 1, the term "dog" for the available SDM refers broadly to all breeds and ages of dogs. In the absence of additional specific adjectives, the generated image of a dog is random. However, in practice, each user has a specific and largely consistent preference for the appearance of a dog. Therefore, users would prefer the term "dog" to directly generate an image that aligns with their envisioned appearance, without the need to input repetitive and redundant descriptive prompts each time. To achieve customized configurations, we assume the application uses the Dreambooth technique to fine-tune the downloaded SDM with private images. For example, the model can be fine-tuned to identify a dog as the user's specific pet, such as a golden retriever puppy. Meanwhile, the cluster-wide model at the edge is fine-tuned using an image collection representative of a specific user cluster. These images can be either shared by users or generated based on prompts provided by them. For example, as illustrated in Fig. 1, in a cluster focused on various breeds of puppies, the data collection for fine-tuning consists of a selection of Samoyed and Golden puppy images shared by users and images generated by the prompt corgi puppy. Then, the fine-tuned cluster-wide model can represent a dog's image as a shared representation common to different breeds of puppies. *A case study on this, along with the visualization results, are presented in Section VI.*

C. Hybrid Inference Model

Although the SDM consists of three components: *Text encoder*, *U-Net*, and *Variational Autoencoder*, the computational complexity of the U-Net is the most significant, and it repeatedly participates in the denoising process iterations [31]. In this sense, we disregard the latency of the Text Encoder and Variational Autoencoder, assuming that both the inference latency at the edge and local sides are mainly governed by their own number of denoising steps. Notably, the latency per denoising step differs between the edge and local sides, influencing their overall inference latency.

In contrast to traditional DNNs, such as convolutional neural networks, fully connected networks, and Transformer variants, which features a *straightforward*, *single forward pass*

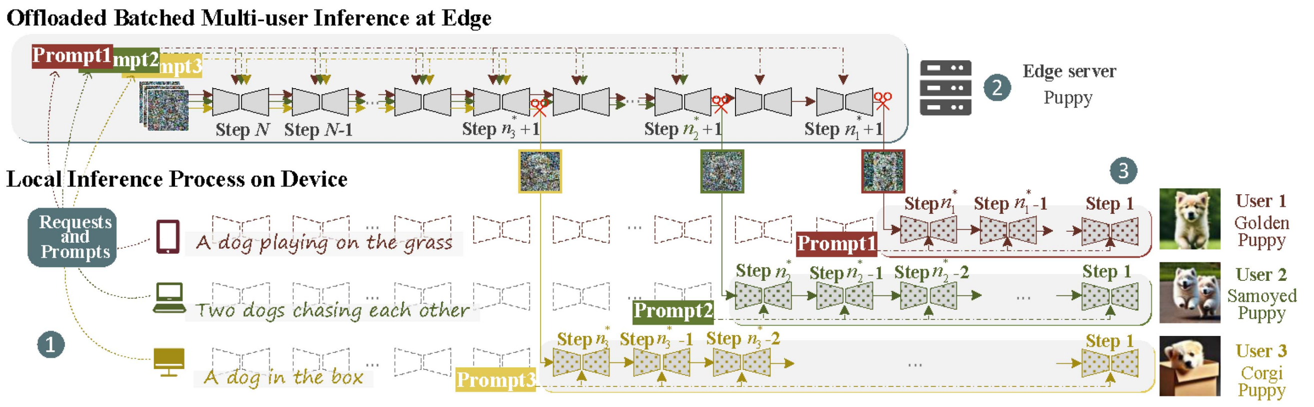


Fig. 2. Illustration of the hybrid inference process with n_i^* optimized individually for each user.

for inference, diffusion models employ an iterative inference process. Specifically, starting with a noise vector, the model progressively applies a series of denoising steps, repeatedly utilizing the same DNN, such as U-Net, to incrementally generate a high-quality output. The inherently recurrent nature of the diffusion model's inference process challenges the effectiveness of existing partition-based offloading strategies for DNNs. To address the associated inference latency, this work proposes a hybrid inference model that partitions the iterative inference process in the temporal domain, as illustrated in Fig. 2.

Numerous simulation experiments indicate that the initial denoising steps primarily capture common low-level features, such as contours and blobs, with minimal inclusion of personalized information. Consequently, as shown in the bottom of Fig. 2, users can offload the *initial* denoising steps to a cluster-wide model, followed by a customized denoising process locally. Let the total number of denoising steps be denoted by N. Since the inference process reverses the sequence from the original data back to pure noise, the denoising steps are indexed in descending order. Thus, step N is executed first, and step 1 is executed at last. Moreover, we assume that the last $\hat{N}$ denoising steps must be executed locally to meet users' personalized requirements. The value of $\hat{N}$ is determined experimentally in Section VI. Thereby, with the split point $n_i^* \in [\hat{N}, N]$, the denoising steps from N to n_i^*+1 are executed at the edge, and the remaining denoising steps from n_i^* to 1 are performed locally. For an extreme case, $n_i^* = N$ indicates that the entire inference process is performed locally.

Moreover, as shown in the top of Fig. 2, To optimize the offloading process at the edge, the batching technique is employed to harness parallel computing capabilities and reduce memory access time. Specifically, at intervals of duration Δ, the edge server assembles the pure noisy data, denoising step indexes, and prompts of all the granted users, and batches them separately. Then, the formed three individual batches are fed as three distinct inputs into the model for parallel and iterative inference. Due to varying offloading task volumes among different users, some tasks may complete earlier, resulting in a reduced batch size.

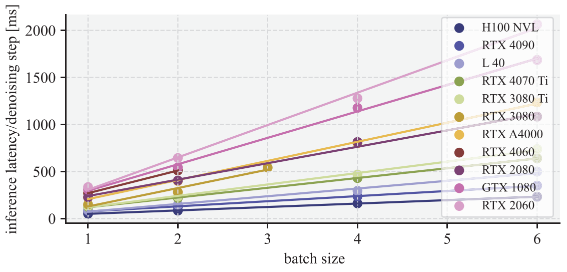


Fig. 3. Illustration of the impact of batch size on latency per denoising step, based on empirical measurements.

According to the NVIDIA report,[1] when utilizing the batching technique, the function of the computational latency on the batch size for a specific neural network is observed to be approximately linear, usually with a large constant term [20]. In this context, we use $\mathscr{L}(b) = \mathrm{k}b + \mathrm{h}$ to characterize the computational latency of *one denoising step*, where b represents the batch size. Meanwhile, k and h are constant terms whose values depend on the adopted models of GPUs, which can be determined experimentally as shown in Fig. 3. Hereinafter, for the sake of simplicity, we use $(\mathrm{k}_i, \mathrm{h}_i)$ and $(\mathrm{k}_e, \mathrm{h}_e)$ to characterize the computation power of user i and edge server, respectively. For the local inference, we can default the value of b to always be 1. Then, we denote the latency per local denoising step for user i by $\mathscr{L}_i(1)$ for short. Furthermore, considering the commonly used data partitioning approach in parallel inference, we assume that with G GPUs on the edge server, the inference latency can be approximately reduced to the latency corresponding to a batch size of $1/G$ of the original, that is, $\mathscr{L}(b)$ can be further modified as $\mathscr{L}_e(b, G) \approx \mathrm{k}_e(b/G) + \mathrm{h}_e$.

D. Quality of Experience Metric

1) End-to-End Latency: The end-to-end (E2E) latency for a stable diffusion task is measured from the moment the user sends

[1]https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-product-literature/t4-inference-print-update-inference-tech-overview-final.pdf

the offloading request to the edge server until the final generated image is obtained. As stated in Section III-A, the edge server responds to requests in two ways, each leading to a different composition of E2E latency. For users whose requests are denied, the E2E latency is equivalent to the sum of RTT and the latency of full local inference, expressed as $L_i = \ell_i^{\mathrm{R}} + \ell_i^{\mathrm{L}}$. For users whose requests are granted, the E2E latency includes two additional components, which is given by $L_i = \ell_i^{\mathrm{R}} + \ell_i^{\mathrm{E}} + \ell_i^{\mathrm{T}} + \ell_i^{\mathrm{L}}$, where ℓ_i^{E} denotes the inference latency at the edge, and ℓ_i^{T} represents the total transmission latency, including both uplink and downlink. The detailed E2E latency calculations under a given strategy policy are provided in Section IV.

2) Personalized Accuracy Index: In the context of efficient diffusion offloading, we introduce a tailored metric called the Personalized Accuracy Index (PAI), which integrates both personalization and accuracy. To optimize the offloading strategy, we establish the relation between PAI and the optimal split point n^*. First, for personalized accuracy measurement, we leverage the Contrastive Language–Image Pre-training (CLIP) model[2] to assess the the similarity between the generated images and the user's prompt *with additional personalized descriptions concerning the targeted object, style, and other relevant aspects*, denoted as $\mathrm{CLIP}(\mathrm{p}_i^+, \mathrm{m}_i)$. To further focus on personalized performance, we use the Learned Perceptual Image Patch Similarity (LPIPS) model[3] to measure differences in the final images generated by individual users, which is expressed by $\overline{\mathrm{LPIPS}} = \mathbb{E}_{1 \leq i < i' \leq |\mathcal{I}|}[\mathrm{LPIPS}(\mathrm{m}_i, \mathrm{m}_{i'})]$. These images denoted by $\mathrm{m}_i, \mathrm{m}_{i'}$ are based on the same intermediate result provided by the cluster-wide model at the edge server, with each user applying their customized models for subsequent denoising inference. In this sense, a greater difference between the images generated by different users indicates stronger personalization. Based on the fine-tuned cluster-wide model and local personalized model presented in Section III-B, we conducted extensive simulation experiments for hybrid inference by offloading varying numbers of denoising steps to the edge server, *which is detailed in Section VI*. From the simulation results, we can observe that, as the split point decreases (the number of offloading steps increases), both the accuracy and personalization metrics exhibit a downward trend, with the decline in personalization being more pronounced. However, this trend is not gradual. As the number of offloaded denoising steps increases, the personalized features of each user remain well-preserved within a certain range. Yet, once a critical threshold–approximately when the offloaded steps reach or exceed half of the total denoising steps–is crossed, the degradation of these features accelerates significantly. With excessive offloading, the generated images across different users become nearly indistinguishable, ultimately losing their distinct personalized characteristics. This phenomenon further confirms the necessity of setting the value of $\hat{N}$ as stated in Section III-C.

Moreover, to capture this effect, we modulate the $\overline{\mathrm{LPIPS}}$ values using a sigmoid function, $\sigma_{\mathrm{P}}(x) = \frac{1}{1+\exp(-a(x-b))}$, where $x = \overline{\mathrm{LPIPS}}$ with varying split point, b represents the threshold at which personalization undergoes a significant shift, and a indicates the slope of this change. Based on the range of $\overline{\mathrm{LPIPS}}$ values and our observations, we set $a = 30$ and $b = 0.1$. Considering the interdependence and mutual reinforcement between accuracy and personalization, the PAI can be expressed by

$$\mathrm{PAI} = \kappa \cdot \mathbb{E}_{i \in I_t}\left[\mathrm{CLIP}\left(\mathrm{p}_i^+, \mathrm{m}_i\right)\right] \cdot \sigma_{\mathrm{P}}\left(\overline{\mathrm{LPIPS}}\right), \tag{1}$$

where $\kappa = 3$. Similarly, we also use the sigmoid fitting approach, denoted by $\mathscr{F}(\cdot)$, with the fitting parameters $a_{\mathscr{F}} > 0$ and $b_{\mathscr{F}} > 0$. This function is utilized in Section IV to formulate the objective function that guides the optimization of the offloading strategy and is detailed in Section VI.

IV. Problem Formulation for Efficient Edge-Assisted Offloading

As stated in Section III-C, we assume that the edge server allocates G GPUs to the set $\mathcal{I}$ of users who have sent requests within the past time period Δ. Therefore, we assume that the current offloading will not impact the performance of previous offloading strategies, and likewise, the performance of the current offloading strategy will not be affected by subsequent offloading. In this sense, we mainly focus on one snapshot of personalized diffusion model offloading without loss of generality.

Before presenting the problem formulation, we first represent the outcome of the offloading strategy by a one-hot matrix $\mathbf{x}^{|\mathcal{I}| \times |\mathcal{D}|}$ and an integer index vector $\mathbf{n}^{*|\mathcal{I}| \times 1}$. Specifically, we define two binary decision variables, x_i^{g} and x_i^{d}, corresponding to *grant* and *deny*, respectively. For simplicity in subsequent notation, we abbreviate as x_i^j, $j \in \mathcal{D} = \{\mathrm{g}, \mathrm{d}\}$. If $x_i^j = 1$, it indicates that the request handling corresponding to j is selected for user i. In this sense, we should ensure

$$\mathrm{C1:} \quad \sum_{j \in \mathcal{D}} x_i^j = 1, \quad \forall i \in \mathcal{I}, \tag{2}$$

which signifies the condition that each request has only one handling approach. Meanwhile, considering computational and communication efficiency as well as stability, we assume that the edge server imposes a concurrent user limit for the number of users performing offloading in a single round, denoted $B_{\max}$. In this sense, we have a common constraint in offloading tasks, as below,

$$\mathrm{C2:} \quad \sum_{i \in \mathcal{I}} x_i^{\mathrm{g}} \leq B_{\max}. \tag{3}$$

In addition, for the spilt point determined by the offloading strategy for user i, we denote it as n_i^*, which is consistent with the definition of n^* in Section III-D. Therefore, we have

$$\mathrm{C3:} \quad \hat{N} \leqslant n_i^* \leqslant N, \quad \forall i \in \mathcal{I}. \tag{4}$$

Moreover, given the coupled relationship between decision variables $\mathbf{x}$ and $\mathbf{n}^*$, an additional constrain is required as below,

$$\mathrm{C4:} \quad x_i^{\mathrm{d}} \cdot (N - n_i^*) = 0, \quad \forall i \in \mathcal{I}, \tag{5}$$

which indicates that for users who do not offload, all denoising steps are completed locally, with the split point determined as $n_i^* = N$.

[2]https://github.com/openai/CLIP.
[3]https://github.com/richzhang/PerceptualSimilarity.

After defining the decision variables and their domains, we proceed to detail the objective function based on the QoE metrics defined in Section III-D. First, we assign each user an individual weight parameter, denoted as α_i, which characterizes their emphasis on latency and PAI. The value of α_i can vary for each user. Therefore, the QoE of user i can be comprehensively evaluated as $\alpha_i \mathcal{F}(n_i^*) - L_i$. According to the definitions of the two forms of E2E latency in Section III-D1, by integrating the decision variables $\mathbf{x}$ into L_i, L_i can be uniformly expressed as

$$L_i = \ell^{\mathrm{R}} + \ell^{\mathrm{T}} \cdot x_i^{\mathrm{g}} + \ell^{\mathrm{E}} \cdot x_i^{\mathrm{g}} + \ell^{\mathrm{L}}. \tag{6}$$

Therein, $\ell^{\mathrm{R}} = K - k$, which is consistent for all users regardless of the different request handling they are assigned with. Meanwhile, ℓ^{L} is also a common term for the E2E latency corresponding to different request handling results, but its representation varies. For $x_i^{\mathrm{d}} = 1$, the expression of ℓ^{L} is fixed as $\ell^{\mathrm{L}} = N\mathscr{L}_i(1)$, while for $x_i^{\mathrm{g}} = 1$, its expression may vary depending on the user. With the constraint (5) in mind, ℓ^{L} can be uniformly expressed as a function of n_i^*, given by $\ell^{\mathrm{L}} = n_i^* \mathscr{L}_i(1)$. In contrast to ℓ^{R} and ℓ^{L}, terms ℓ^{T} and ℓ^{E} only exist in the E2E latency for users whose requests are granted. Specifically, for ℓ^{T}, it consists of the latency for both the uplink transmission of the prompt and the downlink transmission for the intermediate result. We denote the data sizes of the prompt and intermediate noisy data by s_i^{p}, and s_i^{m}, respectively. Although JPEG encoding may cause different split points to produce varying levels of image noisiness, which can slightly affect the data size of intermediate results, these differences have a negligible impact on the E2E latency. Therefore, we assume the data size of the intermediate results to be constant. Then, ℓ_i^{T} for the granted users with $x_i^{\mathrm{g}} = 1$ can be calculated by

$$\ell_i^{\mathrm{T}} = \frac{s_i^{\mathrm{p}}}{\eta W_i^{\mathrm{g}}} + \frac{s_i^{\mathrm{m}}}{\eta W_i^{\mathrm{g}}}, \tag{7}$$

where W_i^{g} represents the bandwidth allocated to user i. According to the straightforward bandwidth allocation described in Section III-A, where granted users share the bandwidth equally, we have

$$W_i^{\mathrm{g}} = \frac{W_{\max}}{\sum_{i\in\mathcal{I}} x_i^{\mathrm{g}}}. \tag{8}$$

The calculation of ℓ_i^{E} shares certain similarity with that of ℓ_i^{T}. However, calculating the computational latency for edge offloading presents more complications. Due to the varying split points n_i^* for each user, the batch size may change during the offloaded inference process, as shown in Fig. 2, leading to potentially different computational latency for each denoising step. Nonetheless, considering that the number of denoising steps predominantly influences the overall latency, we simplify the modeling of the batch size's impact on latency. The batch size during the offloading at the edge is assumed to be fixed as

$$b_{\mathrm{g}} = \sum_{i\in\mathcal{I}} x_i^{\mathrm{g}}. \tag{9}$$

This simplification neglects the impact of certain tasks completing early on the reduction of the denoising step latency for the remaining tasks. With the batch size to the maximum value, it provides a conservative estimate of the worst-case latency. In this context, ℓ_i^{E} can be expressed by

$$\ell_i^{\mathrm{E}} = (N - n_i^*)\, \mathscr{L}_e(b_{\mathrm{g}}, G). \tag{10}$$

In light of above, since we aim to optimize the overall performance of all the users, the optimization problem of the offloading strategy can be formulated as

$$\max_{\mathbf{x},\mathbf{n}^*} \sum_{i\in\mathcal{I}} \alpha_i \mathscr{F}(n_i^*) - L_i \tag{P1}$$

subject to

$$(2), (3), (4), (5).$$

Proposition 1: The multi-dimensional coupled integer programming problem shown in (P1) is NP-hard.

Proof: As shown in (P1), there are two sets of coupled optimization variables, $\mathbf{x}$ and $\mathbf{n}^*$ with high dimensions of $|\mathcal{I}| \times |\mathcal{D}|$ and $|\mathcal{I}| \times 1$, respectively. To demonstrate that (P1) is an NP-hard problem, we first simplify it into the form of a standard NP-hard problem, ensuring that the fundamental complexity and core structure remain unchanged. Then, based on the transitivity of NP-hardness, we infer that the original problem is also NP-hard. Specifically, we fix the value of the decision variable n_i^* by $n_i^* = \hat{N}$ for the granted users, assuming that these users do not have high demands for image quality and are primarily focused on reducing latency. Once users opt for offloading, they offload the maximum possible number of denoising steps, i.e., $(N - \hat{N})$. In this sense, the PAI metric $\sum_{i\in\mathcal{I}} \alpha_i \mathscr{F}(n_i^*)$ can be reduced to a function dependent solely on $\mathbf{x}$, i.e.,

$$\sum_{i\in\mathcal{I}} \alpha_i \mathscr{F}(n_i^*) = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{D}} C_{ij} x_i^j, \tag{11}$$

where $C_i^{\mathrm{d}} = \alpha_i \mathscr{F}(N)$ and $C_i^{\mathrm{g}} = \alpha_i \mathscr{F}(\hat{N})$. Meanwhile, C3 and C4 are removed. Moreover, for the penalty term related to latency, substituting (8) into (7), we have

$$\ell_i^{T} = \sum_{i'\in\mathcal{I}} A_{i'\mathrm{g}i\mathrm{g}}^{\mathrm{T}} \cdot x_{i'}^{\mathrm{g}} \cdot x_i^{\mathrm{g}}, \tag{12}$$

where $A_{i'\mathrm{g}i\mathrm{g}}^{\mathrm{T}} = \frac{s_i^{\mathrm{p}} + s_i^{\mathrm{m}}}{\eta W_{\max}}$. Considering the overall transmission latency of the multi-user system, we have

$$L^{\mathrm{T}} = \sum_{j\in\mathcal{D}} \sum_{j'\in\mathcal{D}} \sum_{i\in\mathcal{I}} \sum_{i'\in\mathcal{I}} A_{i'j'ij}^{\mathrm{T}} \cdot x_{i'}^{j'} \cdot x_i^j, \tag{13}$$

where $A_{i'j'ij}^{\mathrm{T}} = 0$, if $j = \mathrm{d}$ or $j' = \mathrm{d}$. Similarly, substituting (9) into (10), and explicitly expanding the form of the function $\mathscr{L}_e(b, G) = \mathrm{k}_e(b/G) + \mathrm{h}_e$, we have

$$\ell_i^{\mathrm{E}} = D_{i\mathrm{g}}^{\mathrm{E}} \cdot x_i^{\mathrm{g}} + \sum_{i'\in\mathcal{I}} A_{i'\mathrm{g}i\mathrm{g}}^{\mathrm{E}} \cdot x_{i'}^{\mathrm{g}} \cdot x_i^{\mathrm{g}}, \tag{14}$$

where $D_{i\mathrm{g}}^{\mathrm{E}} = (N - \hat{N})\mathrm{h}_e$ and $A_{i'\mathrm{g}i\mathrm{g}}^{\mathrm{E}} = (N - \hat{N})\mathrm{k}_e/G$. When extending to multi-user system, we have

$$L^{\mathrm{E}} = \sum_{i\in\mathcal{I}} \sum_{j\in\mathcal{D}} D_{ij}^{\mathrm{E}} \cdot x_i^j + \sum_{j\in\mathcal{D}} \sum_{j'\in\mathcal{D}} \sum_{i\in\mathcal{I}} \sum_{i'\in\mathcal{I}} A_{i'j'ij}^{\mathrm{E}} \cdot x_{i'}^{j'} \cdot x_i^j, \tag{15}$$

where $D^{\mathrm{E}}_{i\mathrm{d}} = 0$ and $A^{\mathrm{E}}_{i'j'ij} = 0$, except for $A^{\mathrm{E}}_{i'\mathrm{g}i\mathrm{g}}$. In addition, for the latency for local inference, similarly, we have

$$L^{\mathrm{L}} = \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{I}} D^{\mathrm{L}}_{ij}\cdot x^j_i, \tag{16}$$

where $D^{\mathrm{L}}_{i\mathrm{d}} = N(\mathrm{k}_i + \mathrm{h}_i)$ and $D^{\mathrm{L}}_{i\mathrm{g}} = \hat{N}(\mathrm{k}_i + \mathrm{h}_i)$. At last, for L^{R}, we have

$$L^{\mathrm{R}} = \sum_{j\in\mathcal{D}}\sum_{i\in\mathcal{I}} D^{\mathrm{R}}_{ij}\cdot x^j_i, \tag{17}$$

where $D^{\mathrm{R}}_{ij} = K - k_i$, and k_i denotes the moment when user i sends the request to the edge, as stated in Section III-A.

Overall, with (11), (13), (15), (16), and (17), the objective function of (P1) can be rewritten as

$$\max_{\mathbf{x}} \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} D_{ij}\cdot x^j_i - \sum_{j\in\mathcal{D}}\sum_{j'\in\mathcal{D}}\sum_{i\in\mathcal{I}}\sum_{i'\in\mathcal{I}} A_{i'j'ij}\cdot x^{j'}_{i'}x^j_i, \tag{18}$$

where $D_{ij} = C_{ij} - (D^{\mathrm{L}}_{ij} + D^{\mathrm{E}}_{ij} + D^{\mathrm{R}}_{ij})$ and $A_{i'j'ij} = A^{\mathrm{T}}_{i'j'ij} + A^{\mathrm{E}}_{i'j'ij}$. According to [32, Theorem 1], the linear term can be removed from (18) as the diagonal of matrix of quadratic terms. Then, (18) can be reformulated as

$$\max_{\mathbf{x}} \sum_{j\in\mathcal{D}}\sum_{j'\in\mathcal{D}}\sum_{i\in\mathcal{I}}\sum_{i'\in\mathcal{I}} \tilde{A}_{i'j'ij}\cdot x^{j'}_{i'}\cdot x^j_i, \tag{19}$$

where $\tilde{A}_{ijij} = D_{ij} - A_{ijij}$ while all other elements of $\tilde{A}_{i'j'ij} = -A_{i'j'ij}, i' \neq i, j' \neq j$ remain unchanged. If $|\mathcal{D}| = |\mathcal{I}|$, the optimization problem in (19) is is a QAP. However, in the vast majority of cases, $|\mathcal{D}| \ll |\mathcal{I}|$. Additionally, considering the remaining constraint C2 in (3) from the original problem (P1), the simplified version can be more accurately described as a GQAP, widely proven to be NP-hard [33], [34] Since the reduced problem (19) remains NP-hard, the original (P1) includes additional constraints and variables, making it more complex, it is reasonable to infer that (P1) is at least NP-hard.

V. DRL-Convex Hybrid Solution for Efficient Edge-Assisted Offloading

To address the GQAP about $\mathbf{x}$, further complicated by the additional optimization of a coupled decision variable of integer index vector, $\mathbf{n}^*$, in this section, we propose a novel DRL-convex hybrid approach. Specifically, to make the extended version of GQAP tractable, we first transform it into two nested sub-problems with $\mathbf{x}$ and $\mathbf{n}^*$, respectively, which is detailed in Section V-A.

A. Insights for Nested Sub-Problem Formulation

Recall (P1): the optimal values of $\mathbf{x}$ and $\mathbf{n}^*$ are interdependent. The total number of granted users, determined by $\mathbf{x}$, influences the batch size during the inference at the edge server, which in turn affects the latency of each offloaded denoising step. This leaves a customized trade-off between PAI and E2E latency. It is shaped by each user's local computational capacities $(\mathrm{k}_i, \mathrm{h}_i)$ and emphasis factors α_i, ultimately impacting the optimal value of n^*_i. Meanwhile, for each user i, the choice of n^*_i directly affects their local inference latency ℓ^{L}_i and offloaded inference latency ℓ^{E}_i. For a specific emphasis factor α_i, each user's end-to-end latency and PAI jointly determine the overall performance of the whole scenario. Given the shared resources at the edge, it in turn influences the optimal decision variable $\mathbf{x}$.

Nevertheless, through an in-depth analysis of the coupling nature, the variable $\mathbf{x}$ emerges as the dominant decision variable between the two. Given a specific value of $\mathbf{x}$, the inference latency of each denoising step at the edge server becomes determined. Then, the optimization over n^*_i can be decoupled across users and solved with a well-defined closed-form expression with the consideration of their local computational capacities $(\mathrm{k}_i, \mathrm{h}_i)$ and emphasis factors α_i. Moreover, each individual n^*_i collectively contributes to the globally optimal $\mathbf{n}^*$. In other words, for a given $\mathbf{x}$, it is relatively straightforward to derive the optimal $\mathbf{n}^*$ that maximizes system performance under the given $\mathbf{x}$. However, $\mathbf{n}^*$ cannot be pre-set to readily determine $\mathbf{x}$, as it does not reduce the coupling effect among users. This leads to an asymmetric dependency between $\mathbf{n}^*$ and $\mathbf{x}$.

With this in mind, we transform the original problem into two nested sub-problems. We consider the optimization of request handling, $\mathbf{x}$, as the outer-layer optimization, while the optimization of split point, $\mathbf{n}^*$, is treated as inner-layer optimization. Given the explicit formulation and the unified expression of the problem about n^*_i across all users under a specific $\mathbf{x}$, we employ convex optimization with continuous relaxation to solve it, as detailed in Section V-B. However, for a given $\mathbf{x}$, explicitly quantifying how the optimization of $\mathbf{n}^*$ contributes to the overall performance achievable by the current $\mathbf{x}$ remains challenging. Furthermore, the high dimensionality of the decision variable $\mathbf{x}$ adds to the complexity of the outer sub-problem. In this sense, we adopt a DRL-based approach to effectively capture a satisfactory solution about $\mathbf{x}$. *Meanwhile, the implicit influence of $\mathbf{n}^*$ on on the optimality of $\mathbf{x}$ is incorporated into the environment modeling within the DRL paradigm.* This is elaborated in Section V-C. Notably, the powerful learning capability of DRL also mitigates the impact of decomposing (P1) into two sub-problems on overall optimality.

B. Convex Optimization for Split Point Selection

In this subsection, we first present the formulation of inner-layer optimization about $\mathbf{n}^*$. Then, we analyze the convexity of the optimization problem, followed by specifying the method for determining the optimal solution. Finally, based on the problem analysis, we provide insights on selecting emphasis values, α_i, for individual users with varying computational capacities.

For the users with $x^{\mathrm{d}}_i = 1$, we set $n^*_i = N$ directly. Moreover, we have that $L_i = \ell^{\mathrm{R}}_i + \ell^{\mathrm{L}}_i$, which is solely determined by the moment when the user sends request and the computing power of the local device. In this sense, given a feasible $\mathbf{x}$, the inner optimization problem on the split point only needs to be carried out for the granted users for offloading, i.e., the users with $x^{\mathrm{g}}_i = 1$. Moreover, for the granted users, ℓ^{T}_i is solely determined $\mathbf{x}$. Therefore, once $\mathbf{x}$ is given, ℓ^{T}_i can be treated as a constant. Meanwhile, the value of ℓ^{E}_i is joint determined by $\mathbf{x}$ and n^*_i. For a given $\mathbf{x}$, it depends only on n^*_i. Therefore, to ensure clarity in the subsequent analysis, we simply $\ell^{\mathrm{E}}_i(n^*_i, \mathbf{x})$ and $\ell^{\mathrm{T}}_i(\mathbf{x})$ to $\ell^{\mathrm{E}}_i(n^*_i)$

and ℓ_i^{T}, respectively. In this context, the inner sub-problem can be specifically formulated as follows:

$$\max_{n_i^* \in [\hat{N}, N]} \alpha_i \mathscr{F}(n_i^*) - \left(\ell^{\mathrm{R}} + \ell^{\mathrm{E}}(n_i^*) + \ell^{\mathrm{T}} + \ell^{\mathrm{L}}(n_i^*)\right), \tag{P2}$$

where $i \in \left\{i | x_i^{\mathrm{g}} = 1 \wedge x_i^{\mathrm{d}} = 0\right\}$.

Proposition 2: The split point n_i^* optimization is concave problem with a unique optimal solution within the domain $[\hat{N}, N]$.

Proof: For simplicity, we combine the constant terms that are independent of n^*. Recall that $\ell_i^{\mathrm{E}} = (N - n_i^*)\mathscr{L}_e(b_{\mathrm{g}}, G)$ and $\ell_i^{\mathrm{L}} = n_i^* \mathscr{L}_i(1)$, the objective function is rewritten as

$$\mathscr{O}(n_i^*) = \alpha_i \mathscr{F}(n_i^*) - (\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))\, n_i^* + \Gamma, \tag{20}$$

where $\Gamma = \ell^{\mathrm{R}} + \ell^{\mathrm{T}} - N\mathscr{L}_e(b_{\mathrm{g}}, G)$. Recall the expression of $\mathscr{F}(n_i^*) = \frac{1}{1+\exp(-a_{\mathscr{F}}(x - b_{\mathscr{F}}))}$ as well as the second derivative of the sigmoid function, we have

$$\frac{d\mathscr{O}(n_i^*)}{dn_i^*} = \alpha_i a_{\mathscr{F}} \mathscr{F}(n_i^*)(1 - \mathscr{F}(n_i^*)) - (\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G)),$$

$$\frac{d^2\mathscr{O}(n_i^*)}{d(n_i^*)^2} = \alpha_i a_{\mathscr{F}}^2 \mathscr{F}(n_i^*)(1 - \mathscr{F}(n_i^*))(1 - 2\mathscr{F}(n_i^*)).$$

Considering that both α_i and $a_{\mathscr{F}}$ are positive and $\mathscr{F}(n_i^*) \in (0, 1)$, the sign of $\frac{d^2\mathscr{O}(n_i^*)}{d(n_i^*)^2}$ depends on $(1 - 2\mathscr{F}(n_i^*))$. Since $\mathscr{F}(n^*)$ is a monotonically increasing function, we can select an appropriate value of $\hat{N}$ to ensure that $\mathscr{F}(n_i^*) > 0.5$ for all $n_i^* \in [\hat{N}, N]$. Therefore, $\frac{d^2\mathscr{O}(n_i^*)}{d(n_i^*)^2}$ is constantly less than 0 within the domain. In this sense, $\mathscr{O}(n_i^*)$ is strictly concave. Within $[\hat{N}, N]$, (P2) has a unique optimal solution.

First, in case of $\mathscr{L}_i(1) < \mathscr{L}_e(b_{\mathrm{g}}, G)$, which means that the local inference latency per denoising step is lower than the inference latency at the edge server, $\frac{d\mathscr{O}(n_i^*)}{dn_i^*}$ is strictly positive. In this context, when $n_i^* = N$, (P2) reaches its maximum value, which implies that user i completes inference locally. In contrast, if $\mathscr{L}_i(1) > \mathscr{L}_e(b_{\mathrm{g}}, G)$, the solution has three possible cases. Given that $\frac{d^2\mathscr{O}(n_i^*)}{d(n_i^*)^2} < 0$, $\mathscr{O}'(n_i^*) = \frac{d\mathscr{O}(n_i^*)}{dn_i^*}$ is monotonically decreasing on $[\hat{N}, N]$. Denote the term of $\alpha_i a_{\mathscr{F}} \mathscr{F}(n_i^*)(1 - \mathscr{F}(n_i^*))$ by $\bar{\mathscr{F}}'(n_i^*)$. If $\bar{\mathscr{F}}'(\hat{N}) > \bar{\mathscr{F}}'(N) > (\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))$, $\mathscr{O}(n_i^*)$ is monotonically increasing within the domain, and $n_i^* = N$. If $\bar{\mathscr{F}}'(N) < \bar{\mathscr{F}}'(\hat{N}) < (\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))$, $\mathscr{O}(n_i^*)$ is monotonically decreasing within the domain, and $n_i^* = \hat{N}$. In contrast, in case of $\bar{\mathscr{F}}'(N) < (\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G)) < \bar{\mathscr{F}}'(\hat{N})$, there exists a value of n_i^* that makes $\frac{d\mathscr{O}(n_i^*)}{dn_i^*} = 0$, which is the optimal solution. Due to the difficulty in obtaining a closed-form expression for n_i^*, its solution can be found using libraries for solving equations, such as SciPy.

Remark 1: In the optimal solution analysis above, for users whose local inference speed is lower than the inference speed of edge offloading, the value of α_i can be determined based on the following guidelines. For a given $\alpha_i > \frac{(\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))}{a_{\mathscr{F}} \mathscr{F}(N)(1 - \mathscr{F}(N))}$, the focus of user i on PAI is overwhelmingly dominant. For a given $\alpha_i < \frac{(\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))}{a_{\mathscr{F}} \mathscr{F}(\hat{N})(1 - \mathscr{F}(\hat{N}))}$, the optimization problem effectively reduces to optimizing solely for latency. In contrast, for $\frac{(\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))}{a_{\mathscr{F}} \mathscr{F}(\hat{N})(1 - \mathscr{F}(\hat{N}))} < \alpha_i < \frac{(\mathscr{L}_i(1) - \mathscr{L}_e(b_{\mathrm{g}}, G))}{a_{\mathscr{F}} \mathscr{F}(N)(1 - \mathscr{F}(N))}$, it can achieve a personalized trade-off between PAI and latency, with larger values of α_i placing greater emphasis on PAI. Although the exact value of b_{g} is unknown, the service provider can estimate $\hat{b}_{\mathrm{g}}$ based on the number of active users in the area, thereby offering each user a range for selecting α_i. Additionally, it should be noted that the latency optimization considered in the individual split point only accounts for the computational latency. To incorporate other types of latency integrated in Γ, the exponent of the total latency term in $\mathscr{O}(n_i^*)$ can be increased accordingly.

C. DRL-Based Request Handling Optimization

In this subsection, we aim to present a novel low-complexity DRL-based algorithm for the outer-layer problem of an extended GQAP. Specifically, to solve the problem using DRL, we first formulate it as a sequential decision problem, which is detailed in Section V-C1. Subsequently, the optimal decision variable $\mathbf{x}$ is determined by progressively making step-by-step decisions that optimize the long-term reward, corresponding to the objective of (P1). The decision-making policy for each step is introduced in Section V-C2.

1) MDP Construction: As stated in Section II-C, unlike ordering problems such as TSP, incrementally constructing a sequence of nodes is not inherent in variants of GQAP. Consequently, it cannot be straightforwardly transformed into a sequential MDP model, as is common in most existing COPs [26], [27], [28]. Moreover, manually applying masking to exclude previously selected actions from the action space is not a feasible way to characterize MDP transitions in this context. Therefore, we propose a novel MDP transformation paradigm with the following key insights:

- *Sequential Decision Modeling:* The request handling process for each user is treated as a stepwise decision-making task, where the decoupled optimization on n_i^* can integrated into the environment that the agent interacts with in each step.
- *User State Indicator:* To capture dependencies among users, we introduce an indicator in states to track past actions for each user, the current user being processed, and pending users.
- *Cyclic Shifting Mechanism:* This mechanism enhances the agent's ability to distinguish between different users' states during transitions.
- *Reward Backpropagation:* Since rewards depend on both past and future decisions, we propagate rewards backward to ensure accurate assignment at each step.
- *Local-Global State Design:* The MDP state consists of local sub-states, characterizing individual users, and global sub-states, which tracks resource constraints such as the maximum and current number of granted users.
- *Dynamic-Static State Design:* The MDP state comprises dynamic sub-states, capturing state transitions within an

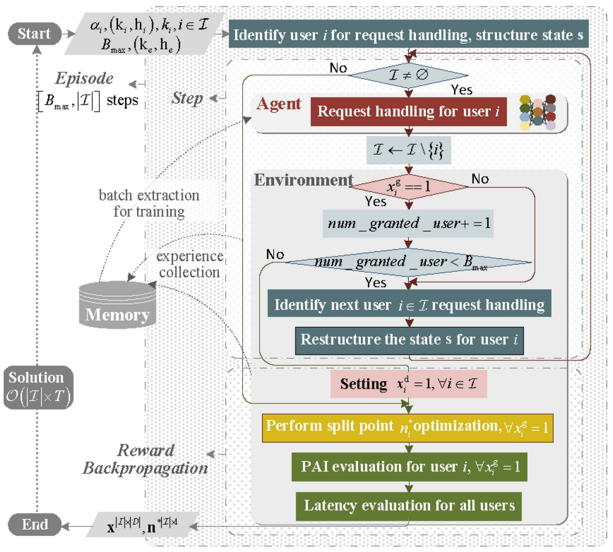

Start
Identify user i for request handling, structure state s
Episode
$[B_{\max}, |\mathcal{I}|]$ steps
Step
Agent
Request handling for user i
Environment
$num\_granted\_user+=1$
Identify next user $i \in \mathcal{I}$ request handling
Restructure the state s for user i
batch extraction for training
experience collection
Memory
Solution $\mathcal{O}(|\mathcal{I}| \times T)$
Setting $x_i^d = 1, \forall i \in \mathcal{I}$
Perform split point n_i^* optimization, $\forall x_i^g = 1$
PAI evaluation for user i, $\forall x_i^g = 1$
Latency evaluation for all users
Reward Backpropagation
End

Fig. 4. Flowchart for transitioning from extended GQAP to decision sequence.

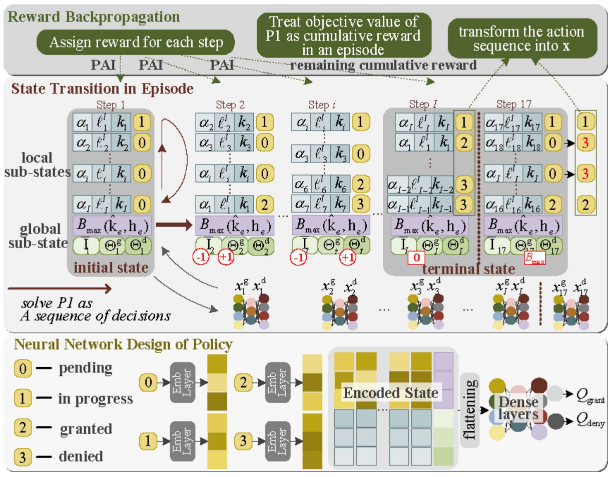

Reward Backpropagation
Assign reward for each step
Treat objective value of P1 as cumulative reward in an episode
transform the action sequence into x
remaining cumulative reward
State Transition in Episode
local sub-states
global sub-state
initial state
terminal state
solve P1 as A sequence of decisions
Neural Network Design of Policy
0 — pending
1 — in progress
2 — granted
3 — denied
Emb Layer
Encoded State
flattening
Dense layers
Q_{grant}
Q_{deny}

Fig. 5. Illustration of DQN-based request handling.

episode, and static sub-states, randomly initialized at the episode's start to help the policy adapt to varying system configurations.

- *Constraint-Aware Termination:* Instead of using penalty terms, episodes terminate automatically according to the global sub-state when resource constraints reach their limits, simplifying the learning process.

The specific MDP is shown in Fig. 4 and described as below.

State Space: To facilitate the agent to provide the optimal solution for a specific scenario, it is essential to incorporate all factors related to the objective value into the state. This includes both *static* information about all users and the edge server, such as computing power, and *dynamic* information shaped by decisions in prior steps. Specifically, as shown in Fig. 5, we define the state as comprising both local and global sub-states, each further divided into static and dynamic elements. Denote the index of steps by t. The local sub-state for user i is denoted by $\mathrm{s}^{\mathrm{l}}_{t,i} = [\alpha_i, \ell_i, k_i; o_{i,t}]$. Therein, α_i is the emphasis of user i on the PAI and E2E latency, k_i is the moments when user i sends the request to user i, and $\ell_i = \mathrm{k}_i + \mathrm{h}_i$ represents the latency per local denoising step, which is dependent on the computing power of user i's device. These three elements are static during the environment evolution. In contrast, $o_{i,t}$ is the dynamic element, which acts as the indicator of four status. For users whose requests are not handled, $o_{i,t} = 0$; for the user which is identified to be in progress of request handling in the current step, $o_{i,t} = 1$; for the user whose request has been granted, $o_{i,t} = 2$; and for the users whose request has been denied, $o_{i,t} = 3$. *For clarity, we denote the index of the user with* $o_{i,t} = 1$ *as* $i = t$. Meanwhile, the global state is denoted by $\mathrm{s}^{\mathrm{g}}_t = [B_{\max}, \hat{\mathrm{k}}_e, \mathrm{h}_e; \bar{\mathrm{I}}_t, \Theta^{\mathrm{g}}_t, \Theta^{\mathrm{d}}_t]$. Similarly, the first three elements are static, where $B_{\max}$ represents the maximum number of users allowed to offload to the edge server in a round, and $\hat{\mathrm{k}}_e$, $(\hat{\mathrm{k}}_e = \mathrm{k}_e/G)$, together with h_e jointly characterizes the computing power of the edge server. Meanwhile, $\bar{\mathrm{I}}_t$, Θ^{g}_t, and Θ^{d}_t are the dynamic elements in the global sub-state, which represent the total number of the users in pending, the total number of the granted users, and the total number of the denied users, respectively. In this context, as shown in Fig. 5, the complete representation of the state can be given by $\mathrm{s}_t = [\mathrm{s}^{\mathrm{l}}_{t,t}, \mathrm{s}^{\mathrm{l}}_{t,t+1}, \ldots, \mathrm{s}^{\mathrm{l}}_{t,I}, \mathrm{s}^{\mathrm{l}}_{t,1}, \ldots, \mathrm{s}^{\mathrm{l}}_{t,i}, \ldots, \mathrm{s}^{\mathrm{l}}_{t,t-1}, \mathrm{s}^{\mathrm{g}}_t]$.

Action Space: In each step t, the action of agent is to grant or deny the offloading request of user i from the perspective of optimizing overall system performance. In this context, we have $a_t \in \{0, 1\}$. If $a_t = 1$, $x^{\mathrm{g}}_t = 1$ and $x^{\mathrm{d}}_t = 0$, otherwise, $x^{\mathrm{d}}_t = 1$ and $x^{\mathrm{g}}_t = 0$.

Transition Rule: The state transition is determined solely by the action in a deterministic manner. Assume that in step t, the request of user t is in progress. Then, in the next state s_{t+1}, $\bar{\mathrm{I}}_{t+1} = \bar{\mathrm{I}}_t - 1$. Moreover, if $a_t = 1$, $o_{t,t+1} = 2$ and $\Theta^{\mathrm{g}}_{t+1} = \Theta^{\mathrm{g}}_t$ +1, otherwise $o_{t,t+1} = 3$ and $\Theta^{\mathrm{d}}_{t+1} = \Theta^{\mathrm{d}}_t$ +1. Meanwhile, the indicator of the user for request handling in the next step is set to 1, i.e., $o_{t+1,t+1} = 1$. Moreover, to enable the agent to more easily capture the distinct characteristics of the currently selected user, we apply cyclic shifting as shown in Fig. 5 to consistently position this sub-state at the beginning of the local sub-state. Additionally, there are two conditions for the end of an episode during the transition process: 1) if, in a state, $\bar{\mathrm{I}}_t = 0$, it signifies that all users' requests have been processed, then the episode ends 2) if, in a state, the value of $\Theta^{\mathrm{g}}_t = B_{\max}$, it indicates that the maximum limit of granted users has been reached, then the remaining users' requests are directly considered as denied and the episode ends. That is, $o_t, i = 3$ for $i > t$.

Reward: Due to the batching technique, the total number of granted users b_{g} determines the inference latency of each offloaded denoising step across all granted user. This, in turn, affects the individual trade-off between E2E latency and PAI, as discussed in Section V-B. Moreover, the reward for processing a user's request at each step depends not only on past decisions, which are embedded in the state representation, but also on future decisions, which significantly influence the final outcome. As a result, the exact QoE for each user cannot be determined until the episode ends. To address the challenge of sparse rewards in the learning process, we adopt a reward backpropagation mechanism to accurately assign rewards at each step. As illustrated in Fig. 5, we first calculate the cumulative reward function–i.e., the objective function value–after the episode terminates based on all o_i in the final state. Then, each user's achieved PAI at every step is treated as an immediate reward, while the difference between the cumulative reward and the cumulative PAI from previous steps serves as the reward for the final step of the episode. The specific formulation is shown as

$$r_t = \begin{cases} \alpha_t \mathscr{F}\left(n_t^*\right), & \text{done} = \text{FALSE}, \\ \sum_{i=t}^{I} \alpha_i \mathscr{F}\left(n_i^*\right) - \sum_{i=1}^{I} L_i, & \text{done} = \text{TRUE}, \end{cases} \quad (21)$$

where I represents the total number of users sending the request for offloading, and the flag "done" serves as an indicator of whether the episode terminates. Notably, as shown in the yellow block in Fig. 4, $\mathscr{F}(n_t^*)$ is derived based on the convex optimization discussed in Section V-B.

2) PER-DQN-Based Policy Design: Recall (21); the objective function in (P1) is transformed into the expected cumulative reward $\mathbb{E}[\sum_{t\in\mathcal{I}} \gamma^t r_t]$, with $\gamma = 1$. Considering the discrete and small action space, we choose the simple, low-complexity DQN algorithm [35] to learn a policy π to maximize the action-state value at each step, which is defined as $Q(\mathbf{s}_t, a_t) \triangleq \mathbb{E}[\sum_{t'=t}^{I_{\text{end}}} \gamma^{t'} r_{t'} | \mathbf{s}_t, a_t]$. The optimal action-value function is defined as $Q^*(\mathbf{s}_t, a) \triangleq \max_\pi Q_\pi(\mathbf{s}_t, a_t)$, from which the optimal policy can be decided as $\pi^* = \arg\max_a Q^*(\mathbf{s}_t, a)$. Then, the Bellman equation for $Q^*(\cdot)$ can be expressed by

$$Q^*\left(\mathbf{s}_t, a_t\right) = \mathbb{E}\left[r_t + \gamma \max_a Q^*\left(\mathbf{s}_{t+1}, a\right) | \mathbf{s}_t, a_t, \pi^*\right]. \quad (22)$$

To enable the agent to accurately learn the Q-values corresponding to different $(\mathbf{s}_t, a_t)$, and more easily capture changes in the indicator, we use a neural network (NN) to model the relationship. Specifically, as shown in Fig. 5, we first embed the indicator within each local sub-state and then concatenate it with the static elements. Finally, the encoded state is flattened and fed into the dense layers.

To help the agent find the optimal policy, we utilize a hybrid of *ε-greedy exploration* [35] and *Boltzmann exploration* [36]. Moreover, in order to keep the learning stability, we use two NNs with the same structure and the initial weights. One named current network is used to choose the action, the weights of which are denoted by $\boldsymbol{\theta}_c$ and update in each training step. The other named target network is used to calculate the Q-value, the weights of which are denoted by $\boldsymbol{\theta}_t$ and update periodically according to $\boldsymbol{\theta}_c$. In each step, the agent samples a mini-batch of the experiences from the *memory* to train the NN. The loss function used in training is given by

$$\mathcal{L}_{\text{loss}} = \mathbb{E}\left[\left(y_t - \left(\max_a Q^*\left(\mathbf{s}_t, a; \boldsymbol{\theta}_c\right)\right)\right)^2\right], \quad (23)$$

where $y_t = r_t + \gamma \max_a Q^*(\mathbf{s}_{t+1}, a; \boldsymbol{\theta}_t)$. Moreover, given the high variance in experiences, we assign the experience about the terminal state more attention. When sampling experiences, we set the ratio of selected terminal states to preceding states at 1:7. Meanwhile, we also employ Prioritized Experience Replay (PER) [37], where the experiences are sampled based on their significance, prioritizing transitions with higher learning potential to improve training efficiency. The setting of hyperparameters is detailed in Section VI.

VI. Simulation and Evaluation

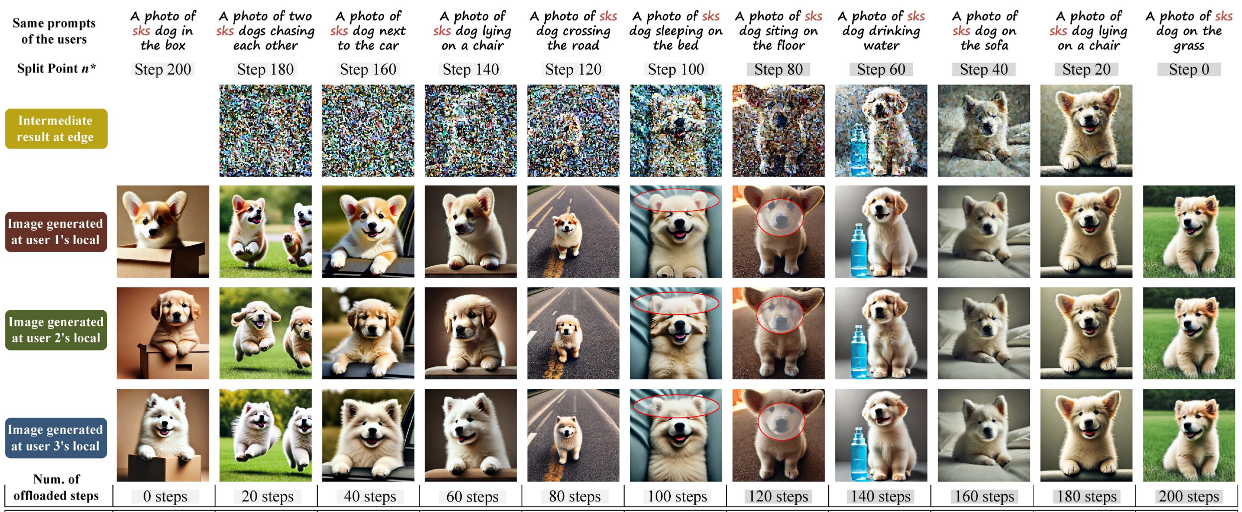
In this section, we detail the fine-tuning of the cluster-wide and local personalized models, along with PAI function fitting. We then describe the hybrid inference simulation setup and analyze PER-DQN performance. All related code, datasets, and fine-tuned models are available at https://github.com/wty2011jl/E-MOPDM.

A. Pre-Deployment Fine-Tuning & PAI Fitting

In this work, we employ the Dreambooth technique [38] to fine-tune a cluster-wide model and three local personalized models. Each local dataset includes seven photos of the corresponding user's pet. The cluster-wide model's dataset comprises nine images: six selected from local datasets and three generated using prompts with detailed personalized descriptions. We fine-tune the text encoder and UNet using an NVIDIA GeForce RTX 4090, to redefine the specific representation corresponding to "dog" for each user individually. Using the fine-tuned models, we simulate the hybrid inference process with the total denoising steps of 200 using 10 prompts, each prompt generate 20 images for each users. The virtual results have been shown in Fig. 6. To facilitate comparison, we present the personalized results generated by three users for each prompt. However, in our algorithm design, we assume that the prompts for each user are distinct. From the figure, we observe that when $n^* > 80$, shared offloading leads to uniform image layouts across three users, yet the dog portraits remain successfully personalized locally without additional detailed descriptions. Conversely, when $n^* < 80$, there is a significant decrease in personalized accuracy. Specifically, only the facial expressions retain the distinct characteristics of different dog breeds, while highly differentiated features such as the ears are not successfully altered. Therefore, we choose $\hat{N} = 80$. In addition, we can also observe that as the number of offloading steps increases, only the personalized aspects deteriorate, and people's visual evaluations align with the changes in our PAI metric defined in (1). Moreover, we present a scatterplot showing the mean PAI values for 20 images generated per prompt across 10 different prompts, with varying numbers of the split point n^* as depicted in Fig. 7. We then apply a sigmoid fitting to these scatter points, with parameters $a_{\mathscr{F}} = 0.0413$ and $b_{\mathscr{F}} = 71.44$.

B. Hybrid Inference Simulation Setup

1) Scenario Parameter Settings: We consider a scenario involving a single edge server and multiple users sending offloading requests within the past Δ. In the following simulations, the determination of inference latency and PAI is based on empirical functions derived from actual experiments, as shown in Figs. 3 and 7. Specifically, we assume that the edge server is equipped with G GPUs, all of the H100 NVL, while users have a single, varying local GPU, which could be one of the other types:


Same prompts of the users
A photo of sks dog in the box
A photo of two sks dogs chasing each other
A photo of sks dog next to the car
A photo of sks dog lying on a chair
A photo of sks dog crossing the road
A photo of sks dog sleeping on the bed
A photo of sks dog siting on the floor
A photo of sks dog drinking water
A photo of sks dog on the sofa
A photo of sks dog lying on a chair
A photo of sks dog on the grass
Split Point n*
Step 200
Step 180
Step 160
Step 140
Step 120
Step 100
Step 80
Step 60
Step 40
Step 20
Step 0
Intermediate result at edge
Image generated at user 1's local
Image generated at user 2's local
Image generated at user 3's local

Num. of offloaded steps	0 steps	20 steps	40 steps	60 steps	80 steps	100 steps	120 steps	140 steps	160 steps	180 steps	200 steps
Accuracy (CLIP)	0.3271	0.3175	0.3131	0.3063	0.3030	0.2962	0.2924	0.2933	0.2849	0.2812	0.2790
P13N (LPIPS)	0.6749	0.4754	0.3838	0.2886	0.2234	0.1831	0.1356	0.0957	0.0580	0.0208	0
PAI metric	0.9813	0.9525	0.9391	0.9157	0.8871	0.8208	0.6528	0.4116	0.1888	0.0717	0.0396

Fig. 6. The visual results of hybrid inference. Therein, "sks" in the prompts is the identifier for the personalized objects to the generated. Meanwhile, in evaluating the semantic accuracy using CLIP, we replace the original prompt term "sks" with personalized descriptive words such as "Corgi," "Golden Retriever," and "Samoyed", respectively for the three users.

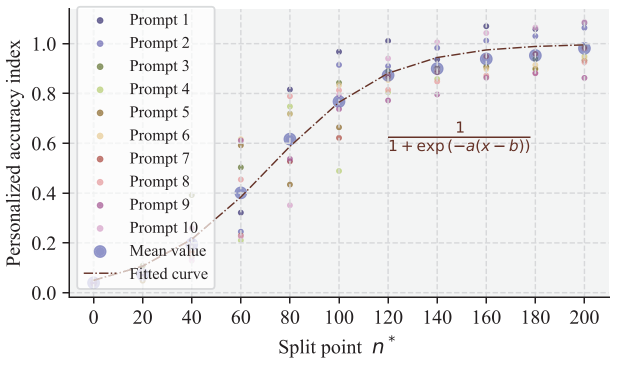


Fig. 7. The fitting curve for the relation between PAI and split point, along with the mean values of accuracy and personalization for CLIP and LPIPS metrics across 200 sets of image generated from 10 prompts at each split points.

{RTX 2060, GTX 1080, ..., RTX 4090}. Meanwhile, the emphasis α_i of user i on the PAI and the latency is randomly selected from the interval $[\frac{(\mathscr{L}_i(1)-\mathscr{L}_e(\hat{b}_g,G))}{a_{\mathscr{F}}\mathscr{F}(\hat{N})(1-\mathscr{F}(\hat{N}))}, \frac{\kappa(\mathscr{L}_i(1)-\mathscr{L}_e(\hat{b}_g,G))}{a_{\mathscr{F}}\mathscr{F}(N)(1-\mathscr{F}(N))}]$, where we set $\hat{b}_g = 20$ and $\kappa = 0.05$. The introduction of κ accounts for the fact that users sending offloading requests tend to be more sensitive to latency to varying extend. Moreover, we set the total bandwidth $W_{\max} = 1$ MHz. Assuming the base station employs power control, the spectral efficiency for each user is set as $\eta = 10$ bits/s/Hz. Considering that the data sizes of the intermediate noisy image and prompt of different users are approximately equal, we substitute the specific noisy image and prompt for each user with their respective means. Thus, we have $s_i^{\mathrm{p}} = 216$ b, and $s_i^{\mathrm{m}} = 4.4$ Mb.

2) PER-DQN Hyperparameter Settings: The Q-network consists of one embedding layer, three hidden linear layers and an output layer. The vocabulary size and embedding dimension of the embedding layer are set as 4 and 3, respectively. Each of the hidden linear layer has 256 nodes. The output layer has two nodes. The target network is updated at a frequency of once every 2000 time steps. The memory capacity is 400000, the batch size is 128, and the learning rate is 0.0001. The exploration probability ε linearly decreases from 0.5 to 0.001. The temperature parameter of Boltzmann exploration linearly decreases from 5 to 0.01. During PER, the priority exponent is set to 0.7, the importance sampling weight is set to 0.3, and the priority offset is set to 0.00002. Moreover, during the training, the reward is scaled to 0.1 of its original value. Moreover, the training is performed on a GeForce RTX 4070 Ti. To save training time, the inference latency of different GPUs is obtained from prior empirical measurements on various GPU models, as summarized in Fig. 3.

C. Hybrid Inference Performance Analysis

First, we analyze the key factors influencing the optimal offloaded denoising steps and their respective impacts. As shown in Fig. 8(a), given an edge server with 8 available GPUs, as the batch size at the edge increases, the inference latency per denoising step at the edge server lengthens, and the optimal number of offloaded steps gradually decreases. Additionally, for the same batch size, the stronger the computational capacity of the user's local device, the fewer offloaded steps are required, with more denoising steps executed locally. Furthermore, as illustrated in Fig. 8(b), with an increase in the number of GPUs on the edge server and the resulting improvement in inference speed, the optimal number of offloaded steps also increases. Both sub-figures indicate that as the value of α_i increases, reflecting

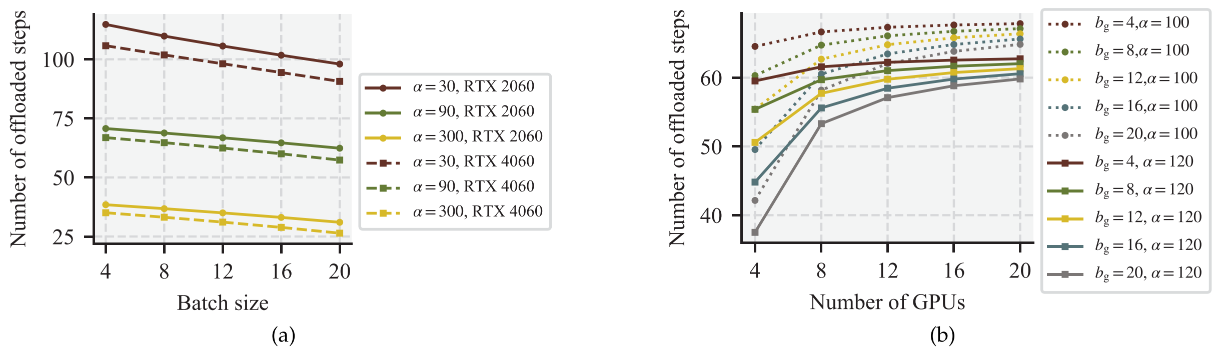


Fig. 8. The relationship between the number of denoising steps offloaded and user emphasis α_i under varying configurations. (a) Analysis with differing batch sizes at the edge server and varying computational capacities of the local terminal. (b) Impact of different numbers of available GPUs on the edge server combined with varying batch sizes.

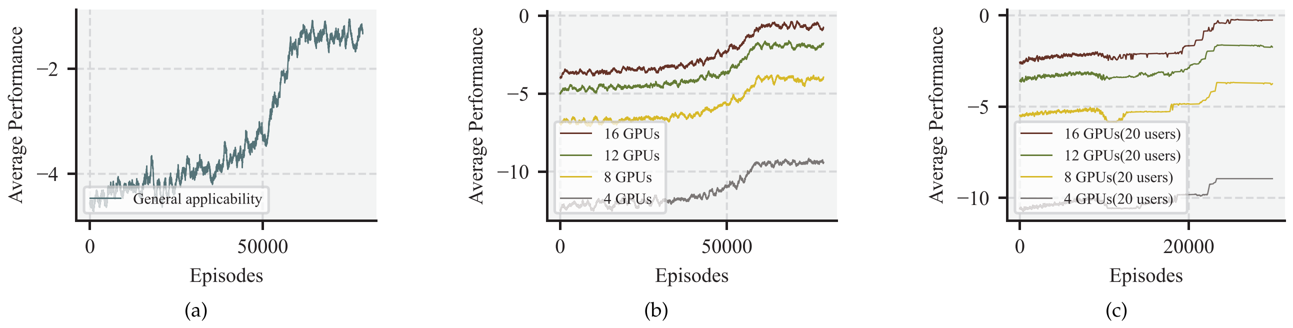


Fig. 9. Learning curves for DRL algorithms across varying scopes: (a) General application with 2000 random seeds and smoothing over 1000 points. (b) GPU-constrained application with 1000 random seeds and similar smoothing. (c) Specific application with a single seed and smoothing over 100 points.

heightened user focus on PAI, the optimal offloaded steps decrease accordingly. Moreover, when computational resources are relatively abundant, α_i exerts a dominant influence on the optimal offloading steps. Conversely, with a smaller number of available GPUs, batch size has a more significant impact on the optimal number of offloaded steps.

Next, we integrate the aforementioned split point optimization into the modeling of environmental transitions in the MDP model. Through simulations, we verify the effectiveness of the proposed DQN-convex hybrid solution. Given different applicability scopes, we train three types of Q-networks tailored for the following three scenarios:

- *General Applicability:* For scenarios with evolving computational resources at the edge, dynamic numbers of users, and various user combinations, with a greater emphasis on the algorithm's generalizability.
- *GPU-Constrained Applicability:* For cases with specific computational resources, dynamic numbers of users, and various user combinations, striking a balance between generalizability and optimality.
- *Specific Applicability:* For scenarios with fixed computational resources, a specific number of users, and a particular user combination, focusing more the with a greater focus on the algorithm's optimality.

The learning curves for the three application scopes are shown in Fig. 9. From the figure, we can observe that the agent can learn effectively in all three cases. Additionally, the more generalizable the model, the greater the fluctuation in its convergence curve, which is mainly due to the inherent diversity of the environment. Moreover, the performance of models with different applicability scopes is shown in Fig. 10. From Fig. 10(a), we observe that the model tailored to specific applications achieves generally superior performance. Specifically, excluding the impact of local computational heterogeneity and user preferences, the average performance tends to increase as the number of users decreases and the number of available GPUs on the edge server increase. As shown in Fig. 10(b) and (c), this improvement is primarily due to the inference latency reduction achieved through offloading, which outweighs the performance loss from potential PAI degradation. However, it is important to note that each performance value in this figure is derived from a specific configuration. Thus, the increase in user count from 8 to 12 not only represents a 1/3 increase in users, but the additional users' computational power and emphasis parameters also impact performance. As shown in Fig. 11(a), when 100 cases are randomly generated for different user counts and then averaged, the sudden "jump" from 12 to 8 in the curve is smoothed out. Given the slight performance difference across the three algorithms, the model for general application is preferred in highly dynamic environments, as it avoids the computational cost associated with frequent retraining and fine-tuning. For scenarios with a fixed set of requesting users, the model tailored for specific

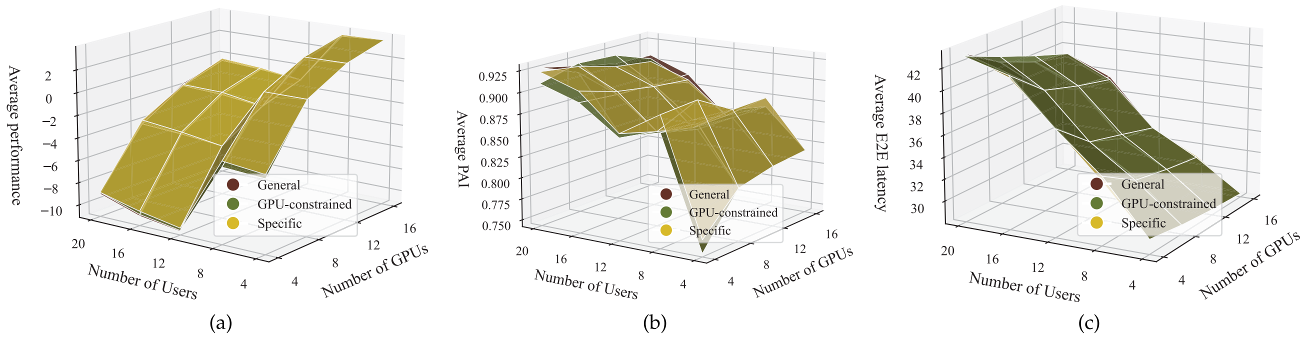


Fig. 10. Performance comparison of three DRL algorithms with different applicability scopes for the same specific case. (a) Overall all performance. (b) Average PAI. (c) Average E2E latency.

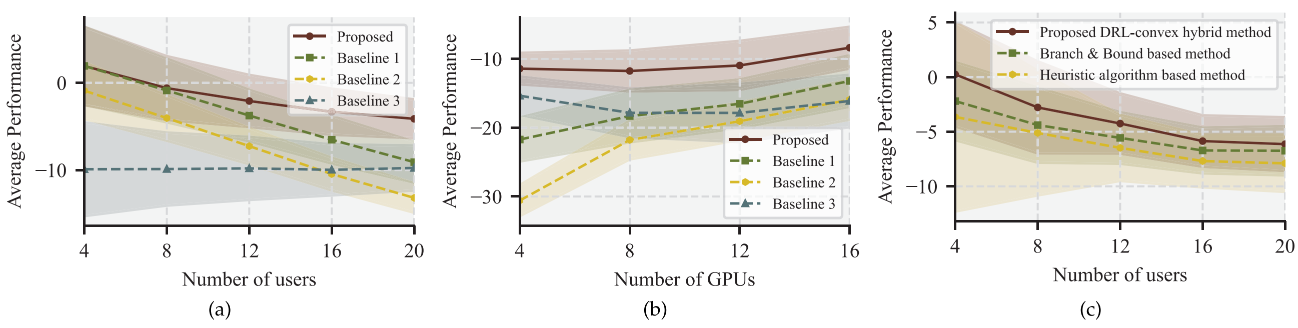


Fig. 11. Performance comparison : (a) Comparison between three baselines: Average performance versus number of users (100 cases/user count); (b) Comparison between three baselines: Average performance versus number of GPUs on edge server (100 cases/GPU configuration). (c) Comparison between DRL-convex hybrid, Branch & Bound, and Heuristic methods (100 cases/user count).

applications can be selected to achieve optimal performance. The model for GPU-constrained applications strikes a balance between generality and optimality by training separate models tailored to each GPU availability scenario, leveraging the model redundancy to accommodate varying GPU constraints.

In the subsequent algorithm comparisons, we focus on the more common dynamic scenarios and use the *general model* for performance analysis. First, we compare this algorithm with the commonly used straightforward approaches for personalized diffusion model inference. Three baseline methods considered are as follows:

- *Baseline 1:* Within the maximum batch size supported by the edge server, all users offload to the edge, with optimization of the split point.
- *Baseline 2:* Within the maximum batch size supported by the edge server, all users offload to the edge, with a fixed offloading of $\hat{N}$ steps.
- *Baseline 3:* All users perform the complete inference locally.

For baseline 1 and baseline 2, if the total number of users sending requests exceeds the maximum batch size supported by the edge server, users who send requests later perform inference locally. Fig. 11 presents a comparison of the proposed algorithm with three baseline methods, illustrating the performance trends as the number of users and the available GPUs on the edge server vary. Specifically, in Fig. 11(a), we fix the number of available GPUs on the edge server to 8, while in Fig. 11(b), we fix the number of users in each case to 20. Observing both subfigures, baseline 3, where all inference is completed locally by users, shows no noticeable trend in average performance as the number of users or available GPUs changes. In contrast, for the other three methods, average performance declines as the number of users increases or the number of GPUs decreases. Furthermore, since baseline 1 incorporates split point optimization compared to baseline 2, baseline 1 consistently maintains a stable performance gap above baseline 2. Additionally, due to its joint optimization of split point and request handling, the performance variation of the proposed method is narrower than that of baselines 1 and 2, and it consistently outperforms other baseline algorithms. This result further indicates that the algorithm can flexibly adaptively trade off between PAI and latency across different scenarios, achieving optimal overall performance.

Finally, we demonstrate the effectiveness and optimality of the DRL-convex hybrid solution for the extended GQAP by comparing it with two widely used methods for GQAP: a heuristic algorithm and a branch & bound-based Integer Linear Programming (ILP) method. The heuristic algorithm selects is a genetic algorithm with a population size of $P = 100$ and an iteration count of $M = 200$. Additionally, as the optimal split point lacks a closed-form solution, we omit the split point optimization in the Branch & Bound-based ILP method for simplicity. As illustrated in Fig. 11(c), the proposed DRL-convex hybrid solution outperforms the other methods, achieving superior performance.

TABLE II
COMPLEXITY COMPARISON REGARDING USER COUNT

Order↑	Name of Method	Complexity
1	DRL-convex hybrid	$O(I)$
2	Heuristic algorithm	$O(2 \cdot P \cdot M \cdot I)$
3	Branch & Bound	$O(2^{2\times I})$

Specifically, due to the omit of the split point optimization, the Branch & Bound based method, the optimality of the method consistently remains lower than that of our proposed method. Additionally, although the heuristic algorithm does not require the rigorous mathematical formulation demanded by the Branch & Bound based method, it can incorporate split point optimization. However, its stability in terms of optimality is noticeably lower than that of the other two algorithms. As a result, while it may occasionally achieve the same level of optimality as our proposed DRL-convex hybrid method, its average performance is the poorest. Moreover, the complexity of the three methods with with respect to user count I as shown in Table II. From the table, since in the process of constructing the MDP model, we transform the original problem with high-dimensional optimization variables into a decision sequence problem focused on a single variable, the proposed DRL-convex hybrid method shows a linear complexity of $O(I)$. This linear growth suggests that the DRL-Convex Hybrid Method is scalable and well-suited for scenarios with a large number of users. In contrast, the Heuristic Algorithm-Based Method has a complexity of $O(2 \cdot P \cdot M \cdot I)$. Although it also scales linearly with I, its overall complexity is influenced by P and M, requiring careful parameter tuning to ensure efficiency as the user count rises. In addition, the conventional Branch & Bound-Based Method presents an exponential complexity of $O(2^{2\times I})$, which grows rapidly with increasing I. This exponential growth makes it work for small-scale problems and impractical for large-scale scenarios, as the computational cost becomes prohibitive. Overall, with a well-trained Q network, the DRL-convex hybrid algorithm is more viable for larger user counts due to their manageable complexity.

VII. Conclusion

We proposed an efficient offloading framework for deploying personalized SDMs in multi-user scenarios with diverse computing capabilities. To balance latency and accuracy, we have introduced a tunable emphasis parameter and formulated offloading and split-point optimization as an extended GQAP. A DRL-convex hybrid approach has enabled real-time decision-making. Simulations have demonstrated the effectiveness of the proposed framework and solutions.

However, several limitations remain: 1) In this work, the cluster-wide model and local models share the same size and are independently trained on different datasets. This may limit the cluster-wide model's ability to capture common features effectively, and the impact of switching between inference models has not been considered. 2) Our approach assumes a given cluster as the starting point. However, key aspects such as evaluating user similarity within a cluster, defining cluster partitions, and understanding the relationship between cluster-wide model generalization and individual users' final PAI remain unexplored.

In light of above, in future work, we will pursue further optimization from the three perspectives: 1) to enhance the generalization capability of the model, we will design training optimization methods for cluster-specific models, achieving a trade-off between model size and PAI performance; 2) we will explore optimal clustering methods based on task similarity to balance the storage and computational energy consumption of multi-cluster-wide models with PAI performance. 3) We will develop a joint training algorithm for both the cluster-wide model and the personalized local models based on the federated learning with the ingenious hierarchical clustering-based aggregation method. This can also address the scenarios where users have insufficient local data.

References

[1] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10684–10695.

[2] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 6840–6851.

[3] F.-A. Croitoru, V. Hondru, R. T. Ionescu, and M. Shah, "Diffusion models in vision: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 9, pp. 10850–10869, Sep. 2023.

[4] W. Yang et al., "Rethinking generative semantic communication for multi-user systems with multi-modal LLM," 2024, *arXiv:2408.08765*.

[5] A. Kazerouni et al., "Diffusion models in medical imaging: A comprehensive survey," *Med. Image Anal.*, vol. 88, 2023, Art. no. 102846.

[6] X. Zhang et al., "A survey on personalized content synthesis with diffusion models," 2024, *arXiv:2405.05538*.

[7] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," 2020, *arXiv:2010.02502*.

[8] X. Yang, D. Zhou, J. Feng, and X. Wang, "Diffusion probabilistic model made slim," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22552–22562.

[9] G. Fang, X. Ma, and X. Wang, "Structural pruning for diffusion models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2023, Art. no. 731.

[10] X. Li et al., "Q-diffusion: Quantizing diffusion models," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 17535–17545.

[11] E. Liu, X. Ning, Z. Lin, H. Yang, and Y. Wang, "OMS-DPM: Optimizing the model schedule for diffusion probabilistic models," in *Proc. 40th Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 21915–21936.

[12] H. Du et al., "Exploring collaborative distributed diffusion-based AI-generated content (AIGC) in wireless networks," *IEEE Netw.*, vol. 38, no. 3, pp. 178–186, May 2024.

[13] C. Yan et al., "Hybrid SD: Edge-cloud collaborative inference for stable diffusion models," 2024, *arXiv:2408.06646*.

[14] F. Bao et al., "All are worth words: A ViT backbone for diffusion models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22669–22679.

[15] Y. Balaji et al., "eDiff-I: Text-to-image diffusion models with an ensemble of expert denoisers," 2022, *arXiv:2211.01324*.

[16] E. Liu, X. Ning, Z. Lin, H. Yang, and Y. Wang, "OMS-DPM: Optimizing the model schedule for diffusion probabilistic models," in *Proc. 40th Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 21915–21936. [Online]. Available: https://proceedings.mlr.press/v202/liu23ab.html

[17] S. Yang, Y. Chen, L. Wang, S. Liu, and Y.-C. Chen, "Denoising diffusion step-aware models," 2023, *arXiv:2310.03337*.

[18] G. Xie et al., "GAI-IoV: Bridging generative AI and vehicular networks for ubiquitous edge intelligence," *IEEE Trans. Wireless Commun.*, vol. 23, no. 10, pp. 12799–12814, Oct. 2024.

[19] Q. Cai, Y. Zhou, L. Liu, Y. Qi, and J. Shi, "Prioritized assignment with task dependency in collaborative mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 13505–13521, Dec. 2024.

[20] Z. Liu, Q. Lan, and K. Huang, “Resource allocation for multiuser edge inference with batching and early exiting,” *IEEE J. Sel. Areas Commun.*, vol. 41, no. 4, pp. 1186–1200, Apr. 2023.

[21] W. Ma and L. Mashayekhy, “Video offloading in mobile edge computing: Dealing with uncertainty,” *IEEE Trans. Mobile Comput.*, vol. 23, no. 11, pp. 10251–10264, Nov. 2024.

[22] A. Silva, L. C. Coelho, and M. Darvish, “Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search,” *Eur. J. Oper. Res.*, vol. 292, no. 3, pp. 1066–1084, 2021.

[23] F. Glover and E. Woolsey, “Converting the 0-1 polynomial programming problem to a 0-1 linear program,” *Operations Res.*, vol. 22, no. 1, pp. 180–182, 1974.

[24] H. D. Sherali and W. P. Adams, “Reformulation-linearization techniques for discrete optimization problems,” in *Handbook of Combinatorial Optimization*, Berlin, Germany: Springer, 1998, pp. 479–532.

[25] A. A. Pessoa, P. M. Hahn, M. Guignard, and Y.-R. Zhu, “Algorithms for the generalized quadratic assignment problem combining lagrangean decomposition and the reformulation-linearization technique,” *Eur. J. Oper. Res.*, vol. 206, no. 1, pp. 54–63, 2010.

[26] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6348–6358.

[27] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, “Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning,” 2019, *arXiv:1911.04936*.

[28] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, and A. A. Cire, “Combining reinforcement learning and constraint programming for combinatorial optimization,” in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 3677–3687.

[29] M. Pashazadeh and K. Wu, “On the difficulty of generalizing reinforcement learning framework for combinatorial optimization,” 2021, *arXiv:2108.03713*.

[30] P. S. Bagga and A. Delarue, “Solving the quadratic assignment problem using deep reinforcement learning,” 2023, *arXiv:2310.01604*.

[31] B.-K. Kim, H.-K. Song, T. Castells, and S. Choi, “BK-SDM: A lightweight, fast, and cheap version of stable diffusion,” 2023, *arXiv:2305.15798*.

[32] P. L. Hammer and A. A. Rubin, “Some remarks on quadratic programming with 0-1 variables,” *Revue française d’informatique et de recherche opérationnelle. Série verte*, vol. 4, no. V3, pp. 67–79, 1970.

[33] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, “A multiplier adjustment method for the generalized assignment problem,” *Manage. Sci.*, vol. 32, no. 9, pp. 1095–1103, 1986.

[34] S. Sahni and T. Gonzalez, “P-complete approximation problems,” *J. ACM*, vol. 23, no. 3, pp. 555–565, 1976.

[35] V. Mnih et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[36] N. Cesa-Bianchi, C. Gentile, G. Lugosi, and G. Neu, “Boltzmann exploration done right,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6284–6293.

[37] T. Schaul, “Prioritized experience replay,” 2015, *arXiv:1511.05952*.

[38] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, “DreamBooth: Fine tuning text-to-image diffusion models for subject-driven generation,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 22500–22510.

Wanting Yang received the BS and PhD degrees from the Department of Communications Engineering, Jilin University, Changchun, China, in 2018 and 2023, respectively. She is currently a research fellow and scientist with the Singapore University of Technology and Design. She was a visiting student with the Singapore University of Technology and Design. She served as Technical Programme Committee member and reviewers in flagship conferences, such as WCNC, Globecom, ICC, and VTC, and top journals. Her research interests include semantic communication, deep reinforcement learning, martingale theory, edge computing, edge intelligence, generative AI.

Zehui Xiong (Senior Member, IEEE) is currently with the Singapore University of Technology and Design, Singapore. Recognized as a highly cited researcher, he has published more than 250 research papers in leading journals, and he has won more than 10 Best Paper Awards in international conferences. He is the recipient of Forbes Asia 30u30, IEEE Asia Pacific Outstanding Young Researcher Award, IEEE Early Career Researcher Award for Excellence in Scalable Computing, IEEE Technical Committee on Blockchain and Distributed Ledger Technologies Early Career Award, IEEE Internet Technical Committee Early Achievement Award, IEEE TCSVC Rising Star Award, IEEE TCI Rising Star Award, IEEE TCCLD Rising Star Award, IEEE ComSoc Outstanding Paper Award, IEEE Best Land Transportation Paper Award, IEEE Asia Pacific Outstanding Paper Award, IEEE CSIM Technical Committee Best Journal Paper Award, IEEE SPCC Technical Committee Best Paper Award, IEEE Big Data Technical Committee Best Influential Conference Paper Award, and IEEE VTS Singapore Best Paper Award. He has served as the associate director of Future Communications R&D Programme, and deputy lead of AI Mega Centre.

Song Guo (Fellow, IEEE) is a full professor with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology. He also holds a Changjiang Chair Professorship awarded by the Ministry of Education of China. His research interests are mainly in the areas of Big Data, edge AI, mobile computing, and distributed systems. With many impactful papers published in top venues in these areas, he has been recognized as a Highly Cited Researcher (Web of Science) and received more than 12 Best Paper Awards from IEEE/ACM conferences, journals, and technical committees. He is the editor-in chief of *IEEE Open Journal of the Computer Society*. He has served on the IEEE Communications Society Board of Governors, IEEE Computer Society fellow Evaluation Committee, and editorial board of a number of prestigious international journals like *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Cloud Computing*, *IEEE Internet of Things Journal*, etc. He has also served as chair of organizing and technical committees of many international conferences. He is an ACM distinguished member.

Shiwen Mao (Fellow, IEEE) is a professor and the Earle C. Williams Eminent Scholar Chair, and the director of the Wireless Engineering Research and Education Center, Auburn University, Auburn, Alabama. His research interest includes wireless networks, multimedia communications, and smart grid. He received the IEEE ComSoc MMTC Outstanding Researcher Award, in 2023, the IEEE ComSoc TC-CSR Distinguished Technical Achievement Award, in 2019, and the NSF CAREER Award, in 2010. He is a co-recipient of the 2022 Best Journal Paper Award of IEEE ComSoc eHealth Technical Committee, the 2021 Best Paper Award of Elsevier/Digital Communications and Networks (KeAi), the 2021 IEEE Internet of Things Journal Best Paper Award, the 2021 IEEE Communications Society Outstanding Paper Award, the IEEE Vehicular Technology Society 2020 Jack Neubauer Memorial Award, the 2018 ComSoc MMTC Best Journal Paper Award and the 2017 Best Conference Paper Award, the 2004 IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems, and several ComSoc technical committee and conference best paper/demo awards. He is the editor-in-chief of *IEEE Transactions on Cognitive Communications and Networking*. He is a distinguished lecturer of *IEEE Communications Society* and the *IEEE Council of RFID*.

Dong In Kim (Fellow, IEEE) received the PhD degree in electrical engineering from the University of Southern California, Los Angeles, California, in 1990. He was a tenured professor with the School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada. He is currently a distinguished professor with the College of Information and Communication Engineering, Sungkyunkwan University, Suwon, South Korea. He is a fellow of the Korean Academy of Science and Technology and a life member of the National Academy of Engineering of Korea. He was the first recipient of the NRF of Korea Engineering Research Center (ERC) in Wireless Communications for RF Energy Harvesting from 2014 to 2021. He received several research awards, including the 2023 IEEE ComSoc Best Survey Paper Award and the 2022 IEEE Best Land Transportation Paper Award. He was selected the 2019 recipient of the IEEE ComSoc Joseph LoCicero Award for Exemplary Service to Publications. He was the general chair of the IEEE ICC 2022, Seoul. From 2001 to 2024, he served as an editor, an editor at Large, and an area editor of *Wireless Communications I* for *IEEE Transactions on Communications*. He has been listed as a 2020/2022 Highly Cited Researcher by Clarivate Analytics.

Merouane Debbah (Fellow, IEEE) is a professor with the Khalifa University of Science and Technology in Abu Dhabi and founding director of the KU 6G Research Center. He is a frequent keynote speaker at international events in the field of telecommunication and AI. His research has been lying with the interface of fundamental mathematics, algorithms, statistics, information and communication sciences with a special focus on random matrix theory and learning algorithms. In the Communication field, he has been with the heart of the development of small cells (4G), Massive MIMO (5G) and Large Intelligent Surfaces (6G) technologies. In the AI field, he is known for his work on Large Language Models, distributed AI systems for networks and semantic communications. He received multiple prestigious distinctions, prizes and best paper awards for his contributions to both fields. He is a WWRF fellow, a Eurasip fellow, an AAIA fellow, an Institut Louis Bachelier fellow and a Membre émérite SEE.