Intelligent and Adaptive Task Migration in Vehicular Edge-Cloud Computing Environments

Jiahui Zhai, Yaxi Yang, Ziqi Wang, and Junqi Zhang

***Abstract*—The growing prevalence of computationally intensive applications such as autonomous driving and in-vehicle infotainment places a substantial energy burden on modern vehicles. To mitigate this challenge, computational offloading in vehicular edge computing (VEC) has attracted increasing attention. However, existing offloading solutions for VEC often face limitations in practicality, slow convergence, or unsatisfactory optimization quality. To overcome these challenges, this work designs Variational Autoencoder-Enhanced Lévy Differential Evolution Offloader (VELO), an optimization framework for task offloading in VEC environments. VELO dynamically selects between roadside units and cloud servers as offloading targets, aiming to reduce system energy consumption. The framework incorporates a variational autoencoder for dimensionality reduction to accelerate inference and integrates a differential evolution algorithm augmented with a Lévy flight strategy to improve optimization quality. Experimental results show that VELO achieves competitive results, effectively lowering system-level energy consumption while preserving rapid convergence. VELO offers a promising solution to reduce the computational load on next-generation vehicle applications and supports the development of energy-efficient, low-carbon intelligent transportation systems.**

***Key Words*—Vehicular edge computing, task offloading, variational autoencoder, and energy-efficient optimization.**

I. Introduction

THE evolution of intelligent transportation systems, driven by computationally intensive applications like autonomous driving and advanced infotainment, is placing unprecedented demands on vehicle onboard units (OBUs). However, these capabilities are fueled by a dramatic increase in computational workload, imposing a significant energy consumption burden. To manage this workload and provide low-latency and energy-saving services, computational offloading in vehicular edge computing (VEC) has emerged as a critical paradigm.

Research in edge computing computational offloading can be broadly divided into two main categories: traditional optimization and machine learning approaches. Traditional optimization methods mathematically model the computation offloading problem to minimize metrics such as latency or energy consumption. For instance, Bi *et al.* [1] employ an improved hybrid metaheuristic algorithm to address the energy minimization problem. However, their design still relies on a relatively low-dimensional decision space and simplified energy coefficients, which makes it difficult to capture highly dynamic vehicular conditions and heterogeneous hardware characteristics in large-scale VEC. Yuan *et al.* [2] propose a novel optimization algorithm, combined with simulation experiments, to solve the problem of minimizing the overall system cost, but their framework assumes a static offloading structure and does not explicitly exploit learned low-dimensional representations to accelerate search in large task populations. Liu *et al.* [3] formulate a mixed-integer optimization problem for mobility-aware multi-hop task offloading in autonomous driving scenarios and solve it via a semidefinite relaxation approach with adaptive adjustment. However, their optimization still operates directly in the original high-dimensional decision space with handcrafted objective terms, which leads to scalability issues and potential local optima when the number of tasks and candidate offloading nodes grows. Cheng *et al.* [4] employ a matching game and dynamic programming to reduce VEC task delay and cost in a collaborative edge and vehicle scenario, but the game-theoretic model is built upon strong assumptions on network states and rational behaviors, which may not hold under rapidly fluctuating energy and channel conditions. Tan *et al.* [5] utilize convex optimization to find the optimal offloading mode for minimizing the weighted sum of overall task response times and communication energy consumed at vehicles. However, such convex formulations require carefully crafted problem structures and cannot easily accommodate nonconvex energy models or high-dimensional, combinatorial offloading patterns. Overall, these traditional methods often face practical challenges in high-dimensional and dynamic VEC scenarios, where strong modeling assumptions and direct search in the original space limit convergence speed.

On the other hand, machine learning approaches are gaining traction as they can better adapt to the complex and dynamic VEC environment. Deep learning can automatically extract features to make predictions and better adapt to the complex environment of VEC [6, 7, 8], but these methods typically require large labeled datasets and fixed data distributions, which is difficult to guarantee in time-varying vehicular networks with evolving energy profiles. To mitigate the need for manual modeling, deep reinforcement learning (DRL), which learns through direct environmental interaction, provides an alternative. There is currently extensive research on the application of DRL in VEC [9, 10, 11, 12, 13], leveraging its advantages in automatically extracting features and adapting to complex, high-dimensional environments for optimizing offloading strategies. Nevertheless, the "trial-and-

Manuscript received December 8, 2025; revised December 10 and December 15, 2025; accepted January 6, 2026. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

J. Zhai, Y. Yang, Z. Wang, and J. Zhang are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China.

Corresponding author: Ziqi Wang

error" exploration process often leads to long training time, high online learning cost, and unstable performance when task distributions or network conditions shift, which hinders deployment under strict energy and latency constraints. In contrast, this work proposes an energy-aware computational offloading framework that leverages a variational autoencoder (VAE) to build a smooth latent representation of high-dimensional offloading decisions and an enhanced differential evolution (DE) algorithm with an exponentially decaying Lévy flight to efficiently search in this latent space based solely on energy feedback, thereby avoiding heavy labeling or costly DRL training.

Given these problems, this work proposes an energy-aware computational offloading framework called Variational Autoencoder-Enhanced Lévy Differential Evolution Offloader (VELO) for VEC tasks, which leverages a VAE and an enhanced DE algorithm to boost the performance in optimization. We employ a VAE to robustly handle data uncertainty and generate effective low-dimensional representations. The core optimization process is driven by a DE algorithm, enhanced with an exponentially decaying Lévy flight strategy that dynamically balances exploration and exploitation to escape local optima and fine-tune solutions effectively. This framework effectively utilizes the powerful processing capabilities of machine learning models for complex inputs, while the integration with a heuristic algorithm avoids the need for labeled datasets or the long "trial-and-error" issues of DRL. Furthermore, the introduction of Lévy flights effectively mitigates the local optima problem in heuristic searches. VELO also has significant practical utility, as it requires fewer prior assumptions, relying solely on energy consumption for feedback. Experimental results demonstrate that VELO significantly reduces overall system energy consumption, achieving competitive performance and offering a viable path toward low-carbon, energy-efficient intelligent transportation. The main contributions of this work are as follows.

- We propose VELO, a VAE-enhanced differential evolution framework for energy-aware task offloading in a three-tier vehicular edge computing architecture, enabling label-free optimization based solely on energy feedback.
- We formulate an energy minimization model that jointly captures heterogeneous computation efficiencies and transmission costs on vehicles, roadside units (RSUs), and cloud servers under dynamic hardware and network conditions.
- We develop an exponentially decaying Lévy-flight-based differential evolution strategy that performs search in the VAE latent space, and demonstrate through extensive simulations that VELO achieves lower energy consumption and faster convergence than state-of-the-art metaheuristics and autoencoder-assisted evolutionary baselines.

The remainder of this work is organized as follows. Section II reviews the related work. Section III formulates the energy minimization problem. Section IV presents the proposed VELO framework. Section V introduces the experimental setup, evaluation metrics, baseline algorithms, and performance analysis. Section VI concludes this work.

II. Related Work

This section reviews three lines of research closely related to VEC task offloading: (i) optimization-based offloading strategies, (ii) deep and reinforcement-learning-based approaches, and (iii) autoencoder-assisted evolutionary optimization.

A. Optimization-based Offloading Strategies

Optimization-based methods constitute a core line of research for task offloading in mobile-edge and VEC systems. Chai *et al.* [14] investigate joint multi-task offloading and resource allocation in satellite-assisted IoT edge networks. They formulate a coupled optimization problem that captures both communication and computation constraints and solve it via an iterative convex-relaxation algorithm to minimize system delay. However, this formulation assumes relatively stable channels and a favorable convex structure, which limits its applicability to highly dynamic VEC scenarios with nonconvex energy–delay characteristics. Focusing on energy-constrained devices, Jiang *et al.* [15] propose a joint offloading and resource-allocation framework under strict energy budgets. The resulting mixed-integer optimization problem is addressed through a decomposition-based strategy that balances latency and energy consumption. Despite its effectiveness, the approach operates in a high-dimensional decision space and relies on handcrafted decomposition, which may incur combinatorial complexity and slow convergence as task scale increases. Li *et al.* [16] further study computation-rate maximization in wireless-powered edge computing with multi-user cooperation. Their model jointly optimizes wireless power transfer, cooperative scheduling, and binary offloading decisions using successive convex approximation. Nevertheless, its convergence and solution quality depend heavily on problem-specific convexification and lack adaptability to rapidly varying energy conditions. From the perspective of green computing, Yang *et al.* [17] introduce carbon-aware task offloading by explicitly incorporating carbon emission intensity into the optimization objective. It formulates a multi-objective problem that jointly minimizes latency and carbon emissions and derives an offloading policy using convex optimization techniques. This approach, however, assumes accurate knowledge of time-varying carbon intensity and adopts centralized optimization, which limits scalability in dynamic vehicular environments. Similarly, Song *et al.* [18] propose an environmentally conscious resource-management framework for edge–cloud systems by integrating energy consumption and carbon footprint into a unified cost model. It dynamically shifts workloads toward greener computing nodes but depends on precise system modeling and does not explicitly address the dimensionality challenges posed by large-scale task populations.

Distinct from the aforementioned strategies, VELO adopts a novel paradigm by directly addressing high-dimensional offloading decisions. This is achieved by mapping these decisions into a continuous latent space utilizing a VAE, effectively reducing the dimensionality of the search space. Furthermore,

VELO executes a global-local evolutionary search guided exclusively by energy feedback, thereby circumventing the restrictive requirements of model convexity or the necessity for a closed-form analytical solution.

B. Deep Reinforcement Learning-based Task Offloading

Deep learning has been applied to infer optimal offloading strategies in dynamic vehicular environments. Deep neural networks extract contextual features from VEC observations for improved decision making [19]. However, such models typically rely on large-scale labeled datasets and may overfit to specific traffic and network patterns, limiting their robustness when the underlying environment or energy profile changes. To mitigate the need for labeled samples, DRL approaches have been widely investigated [20]. These methods exhibit strong adaptability under dynamic network conditions by learning policies through interaction. However, DRL-based offloading schemes generally require extensive "trial-and-error" exploration and incur high training overhead, especially in high-dimensional state-action spaces with numerous tasks and heterogeneous servers. Moreover, the learned policies can be sensitive to distribution shifts and need to be retrained or fine-tuned when the task distribution or network conditions change, which further increases deployment cost. These characteristics make DRL solutions difficult to deploy in scenarios where rapid adaptation, limited training time, or stable energy-aware performance is required.

In contrast, VELO avoids expensive DRL exploration by conducting evolutionary search directly in a VAE-learned latent space of offloading decisions. Specifically, VAE captures the intrinsic structure of feasible offloading patterns and enables a compact, low-dimensional search space, which significantly improves optimization efficiency compared with conventional evolutionary methods operating in the original decision space. In addition, compared with traditional heuristic or metaheuristic optimization, the latent-space-guided evolution provides better scalability and faster convergence when task scale or system configuration changes. VELO only relies on scalar energy feedback, thereby eliminating the need for labeled datasets or long online training phases, and its population-based optimization mechanism naturally adapts to varying task scales and fluctuating energy coefficients.

C. Autoencoder-Assisted Evolutionary Optimization

Autoencoder-assisted evolutionary optimization has gained increasing traction as a technique for addressing high-dimensional or computationally expensive decision-making problems. Specifically, bi-population cooperative evolutionary strategies with embedded autoencoders demonstrate effectiveness in compressing search spaces to achieve more efficient optimization [21]. However, standard autoencoders within these frameworks often generate latent spaces lacking explicit probabilistic regularization, which can result in irregular or partially discontinuous structures that impede stable population-based search. Furthermore, Cui *et al.* [22] propose an autoencoder-based surrogate-assisted evolutionary algorithm, achieving improved performance for complex optimization tasks by combining dimensionality reduction and surrogate modeling. Nevertheless, their method is tailored for generic expensive optimization rather than for energy-aware VEC offloading, and the surrogate construction inherently introduces additional modeling and tuning overhead. In addition, non-variational autoencoders may still furnish latent representations that lack sufficient smoothness for fine-grained evolutionary refinement.

To overcome these limitations, VELO incorporates a VAE to construct a continuous and structured latent space, in which offloading decisions can be represented as smooth, low-dimensional vectors. This latent space is coupled with an enhanced DE mechanism equipped with an exponentially decaying Lévy flight strategy to balance global exploration and local refinement over different optimization phases. In this way, VELO inherits the advantages of autoencoder-assisted optimization while explicitly addressing the latent-space quality and convergence issues that appear in standard autoencoder (AE)-based evolutionary methods.

III. Problem Presentation

TABLE I Nomenclature for the Problem Formulation

Symbol	Description
N	Total number of tasks to be processed.
$a_{i,0}, a_{i,1}, a_{i,2}$	Binary decision for task i, indicating execution on the vehicle, edge, or cloud, respectively.
C_i	Computational workload of task i.
D_i	Data size of task i for transmission.
$\eta_{\mathrm{v}}, \eta_{\mathrm{e}}, \eta_{\mathrm{c}}$	Energy consumption per CPU cycle on the vehicle, edge, and cloud, respectively.
ϵ_{e}	Energy consumption for edge transmission.
ϵ_{c}	Energy consumption for cloud transmission.

This work considers a three-tier VEC architecture, as illustrated in Fig. 1. The system comprises three main components: the vehicle's OBUs, RSUs, and a remote cloud server. It is assumed that vehicles generate a series of computational tasks that are not required to be executed locally, thereby making them suitable candidates for offloading. Each task is treated as an indivisible processing unit and must be executed entirely on one of the three tiers, *i.e.*, the OBU, an RSU, or the cloud server [23, 24]. This non-partial task offloading assumption is adopted to balance modeling realism and computational tractability in highly dynamic vehicular environments, where task partitioning may introduce additional overhead due to synchronization, intermediate data exchange, and coordination under time-varying network conditions. The key premise is that RSUs and cloud servers are equipped with more advanced and energy-efficient computing architectures compared to the OBUs. As a result, executing tasks on these remote servers can lead to significantly lower computational energy consumption. This gives rise to an optimization problem: the energy cost incurred by transmitting a task to a remote server must be balanced against the energy savings achieved through more efficient remote computation. The objective is to determine the optimal execution location for each task to minimize the total energy expenditure. The main system parameters are

summarized in Table I. We note that extending the proposed framework to support partial or split task offloading by decomposing tasks into finer-grained components is a promising direction for future work.

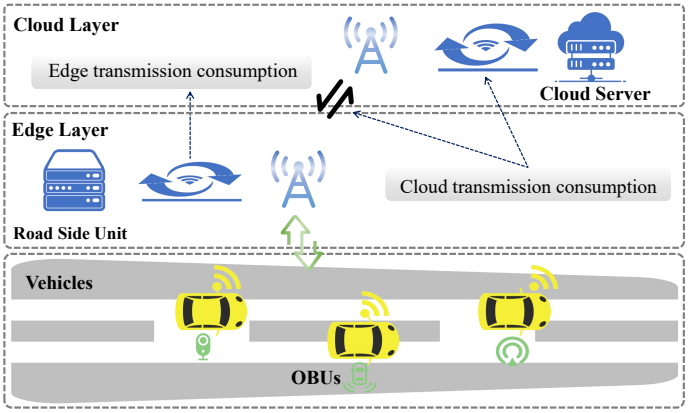

Fig. 1. Architecture of Vehicular Edge Computing.

To formulate this objective, we represent the location for tasks using binary decision vectors. For each task i, the location is $\mathbf{a}_i = \{a_{i,0}, a_{i,1}, a_{i,2}\}$, where each element corresponds to a possible execution location. We analyze a full offloading scenario, *i.e.*, a task can only be executed at a single location. Thus, the constraint $\sum_{j=0}^{2} a_{i,j}=1$ must be satisfied, with $a_{i,j}\in\{0,1\}$. If the task is executed locally ($a_{i,0}=1$), the energy consumed is solely for computation on the OBU. Conversely, if the task is offloaded ($a_{i,1}=1$ or $a_{i,2}=1$), the energy is the sum of the transmission energy from the vehicle and the computational energy at the designated remote server. The total energy consumption E_t is the energy consumptions of all tasks, *i.e.*,

$$E_t = \sum_{i=1}^{N} E_i, \tag{1}$$

where E_i is the energy consumption for executing task i.

The energy consumption for an individual task i is composed of computational energy, determined by the C_i and η, and additional transmission energy, determined by the D_i and ϵ. In this work, we assume that the RSU and cloud servers have sufficient computational capacity to execute offloaded tasks without explicit resource contention, and thus server-side capacity constraints are not modeled in the energy formulation. This assumption is adopted to maintain analytical clarity and to focus on the energy-efficiency trade-off in highly dynamic vehicular environments. Considering the dynamic nature of the real-world environment, the energy efficiency coefficients η and ϵ are not treated as fixed constants. In our implementation, their values are modeled as random numbers drawn from a uniform distribution within a predefined range, reflecting fluctuations in hardware performance and network conditions.

The energy consumed when the task is executed locally on the vehicle is defined as $E_{i,\mathrm{v}}$, *i.e.*,

$$E_{i,\mathrm{v}} = C_i \cdot \eta_{\mathrm{v}}. \tag{2}$$

$E_{i,\mathrm{e}}$ denotes the energy consumption for offloading to and executing on the RSU, including both transmission and computation costs, *i.e.*,

$$E_{i,\mathrm{e}} = D_i \cdot \epsilon_{\mathrm{e}} + C_i \cdot \eta_{\mathrm{e}}. \tag{3}$$

Similarly, the energy for offloading to and executing on the cloud server is calculated as:

$$E_{i,\mathrm{c}} = D_i \cdot \epsilon_{\mathrm{c}} + C_i \cdot \eta_{\mathrm{c}}. \tag{4}$$

Based on the binary decision vector $\mathbf{a}_i = \{a_{i,0}, a_{i,1}, a_{i,2}\}$, the total energy consumption for task i is formulated as a piecewise function:

$$E_i = \begin{cases} E_{i,\mathrm{v}}, & \text{if } a_{i,0} = 1, \\ E_{i,\mathrm{e}}, & \text{if } a_{i,1} = 1, \\ E_{i,\mathrm{c}}, & \text{if } a_{i,2} = 1. \end{cases} \tag{5}$$

Due to hardware differences, the different energy efficiencies of the vehicle, edge, and cloud tiers are captured by the distinct range of η and ϵ. The objective function is to minimize the sum of energy for all tasks:

$$\min E_t = \min \sum_{i=1}^{N} E_i. \tag{6}$$

IV. Proposed Methodology

A. VAE for Dimensionality Reduction

The decision variables for each tasks including $a_{i,0}$, $a_{i,1}$, and $a_{i,2}$. This creates a high-dimensional problem space when the number of task grows. Directly optimizing within this high-dimensional space hinders the ability to find high-quality solutions. Therefore, dimensionality reduction is a critical prerequisite for efficient optimization. While existing research has utilized standard AE for this purpose [22], it often produce a latent space that is irregular or discontinuous. This poor latent space quality can trap optimization algorithms.

To overcome this limitation, we employ a VAE to achieve a more robust and higher-quality dimensionality reduction. The VAE is a generative model that learns a structured, continuous latent representation of the input data. It consists of two main components: a probabilistic encoder and a decoder. The encoder, $q(z|x)$, maps an input data point x to a probability distribution, typically a multivariate Gaussian with a diagonal covariance matrix. It outputs this distribution's mean μ and the log-variance $\log(\sigma^2)$. Then, the latent vector z is obtained by sampling from this distribution:

$$z \sim q(z|x) = \mathcal{N}(z; \mu, \sigma^2 I), \tag{7}$$

where $\mathcal{N}$ denotes the multivariate Gaussian distribution and I is the identity matrix.

To enable gradient-based optimization through backpropagation, the VAE utilizes the reparameterization trick. Instead of directly sampling z from the distribution, the model samples a random noise vector ϵ from a standard normal distribution $\mathcal{N}(0, I)$ and then compute the latent vector z as:

$$z = \mu + \sigma \odot \epsilon, \tag{8}$$

where $\odot$ denotes element-wise multiplication. The decoder, $p(x|z)$, then takes this latent vector z and attempts to reconstruct the original input data, producing $\hat{x}$.

In this paper's VAE network implementation, both the encoder and decoder employ two-layer feedforward networks. Specifically, the encoder first encodes the input to the D_1 dimension, and then further encodes it into the D_0 latent space. Conversely, the decoder operates in the reverse order, first decoding from the D_0 latent space to the D_1 dimension before reconstructing the original input. This hierarchical structure facilitates the network's ability to learn more complex feature representations. Notably, only in the inference stage, each row in the decoder's output matrix is processed as a one-hot encoded vector to align with the problem's requirements.

The VAE is trained by optimizing a loss function composed of two terms: a reconstruction loss L_R and the kullback-leibler (KL) divergence L_{KL}. The total loss corresponds to the negative Evidence Lower Bound, *i.e.*,

$$L_{VAE} = L_R + L_{KL}, \tag{9}$$

where L_R measures the difference between the original input x and the decoder's output $\hat{x}$, and we use the mean squared error for this purpose. L_{KL} is the KL divergence between the encoder's learned distribution $q(z|x)$ and a prior distribution over the latent variables $p(z)$. It is typically a standard normal distribution $\mathcal{N}(0, I)$.

The KL divergence acts as a regularizer, forcing the learned latent distributions to be close to the prior. In that case, it ensures that the latent space is well-structured and continuous. This property is the primary advantage of the VAE for our task, which is more suitable for subsequent optimization algorithms. The smooth structure of the latent space enables the Differential Evolution algorithm to navigate more effectively, allowing it to make small yet meaningful steps toward a global optimum while avoiding the fragmented regions that often hinder standard AE-based representations.

B. Exponentially Decaying Lévy Flight for Enhanced DE

The standard DE algorithm is a widely adopted population-based optimization method that iteratively improves candidate solutions through mutation, crossover, and selection. Its mutation mechanism, typically based on a random strategy, generates a mutant vector V_i by perturbing a base vector with the scaled difference between two other randomly selected vectors from the population:

$$V_i = X_{r1} + F \cdot (X_{r2} - X_{r3}), \tag{10}$$

where X_{r1}, X_{r2}, and X_{r3} are three distinct vectors randomly selected from the current population and $r1 \neq r2 \neq r3$. F is the scaling factor that controls the amplification of the differential variation between vectors.

After the mutation step, crossover and selection are applied to produce the next generation of candidate solutions. However, standard DE is prone to premature convergence, particularly in complex search spaces. To enhance its global search capability, we incorporate the Lévy flight mechanism, which is a random walk strategy that employs occasional large jumps to help the algorithm escape local optima. The step size s is generated as:

$$s = \frac{u}{|v|^{1/\beta}}, \tag{11}$$

where $u \sim \mathcal{N}(0, \sigma_u^2)$ and $v \sim \mathcal{N}(0, \sigma_v^2)$ are drawn from normal distributions, and β is a control parameter. We integrate this into an enhanced mutation strategy where the mutant vector V_i is generated by applying a Lévy flight to the current position:

$$V_i = X_i + \alpha \cdot s \cdot \mathrm{sgn}(r_1 - 0.5), \tag{12}$$

where α is the step scaling factor, r_1 is a random number drawn uniformly from the interval $[0, 1]$, and $\mathrm{sgn}(\cdot)$ is the signum function, which returns $+1$ for positive inputs and -1 for negative ones, determining the direction of the perturbation. This new mutant then undergoes the standard crossover and selection steps.

While Lévy flight is effective for early-stage exploration, its tendency to produce large steps can hinder fine-tuning in the later stages of optimization. To overcome this limitation, we propose a hybrid strategy that combines standard DE search with a Lévy flight mechanism whose influence decays exponentially over time. Specifically, we introduce an iteration-dependent probability $p_l(n)$ that controls whether Lévy-based mutation is applied at iteration n. This probability gradually decreases as the algorithm progresses, allowing the search to shift from global exploration to local exploitation:

$$p_l(n) = \alpha_l \cdot e^{-\beta_l k n}, \tag{13}$$

where $k = n/n_{\max}$, and the parameters α_l and β_l control the decay rate. It enables a seamless transition from global exploration to local exploitation. Initially, a high probability of invoking Lévy flights allows the algorithm to broadly explore the search space. As the search progresses, this probability rapidly decays, causing the algorithm to shift to standard DE operations that excel at local fine-tuning. This creates a dynamic balance that fosters both rapid exploration and efficient convergence.

C. VELO Optimization Framework

By integrating the VAE with this enhanced DE algorithm, an efficient optimization framework VELO is formed in Algorithm 1, which operates in two phases. First, the VAE is trained offline on a sufficient dataset of randomly generated offloading schemes, allowing it to learn a robust, low-dimensional latent representation of the high-dimensional decision space.

Once the VAE is trained, the main optimization task begins with a randomly initialized population of offloading solutions for the algorithm. The core optimization loop proceeds as follows: for any given solution vector x from the DE population, it is first encoded into its latent representation z using the VAE's encoder. Then the algorithm performs its search operations directly within this continuous and structured latent space. A new trial individual in the latent space z' is generated via the hybrid mutation process:

$$z' = \begin{cases} z + \alpha \cdot s \cdot \mathrm{sgn}(r_1 - 0.5) & \text{if } r_2 < p_l(n), \\ z + F \cdot (z_{r1} - z_{r2}) & \text{otherwise.} \end{cases} \tag{14}$$

Algorithm 1 VAE-Enhanced DE with Exponentially Decaying Lévy Flight (VELO)

Offline Training: Train VAE Encoder $E(\cdot)$ and Decoder $D(\cdot)$ on a sufficient set of samples. **Initialize:** Population P_x=$\{x_1, x_2, \ldots, x_p\}$, max iterations $n_{\max}$, parameters $\alpha_l, \beta_l, F, \alpha$. Create latent population P_z by encoding each solution: $z_i \leftarrow E(x_i)$ for all $x_i \in P_x$. $n = 1$ **to** $n_{\max}$ $k \leftarrow n/n_{\max}$. $p_l \leftarrow \alpha_l \cdot \exp(-\beta_l \cdot k \cdot n)$. each individual (z_i, x_i) in populations (P_z, P_x) Generate random number $r_1 \in [0, 1]$. $r_1 < p_l$ Generate Lévy step s. Generate random number $r_2 \in [0, 1]$. $v_z \leftarrow z_i + \alpha \cdot s \cdot \mathrm{sgn}(r_2 - 0.5)$. Select distinct z_{r1}, z_{r2} from P_z. $v_z \leftarrow z_i + F \cdot (z_{r1} - z_{r2})$. Perform crossover between z_i and v_z to create u_z. $x_{\text{trial}} \leftarrow D(u_z)$. $f(x_{\text{trial}})$ is better than $f(x_i)$ $z_i \leftarrow u_z$. $x_i \leftarrow x_{\text{trial}}$. **Return:** The best individual found in P_x.

The choice between these mutation strategies is governed by comparing a random number $r_2 \in [0, 1]$ with the probability $p_l(n)$. z_{r1} and z_{r2} are two other randomly selected different individuals from the encoded population. This mutant vector z' then undergoes a crossover operation with the original latent vector z to create the final trial vector. This trial vector is then decoded back into the original high-dimensional solution space using the VAE's decoder, yielding a new candidate solution x'=Decoder(z'). Finally, this new solution x' is evaluated using E_t to determine its fitness. This entire process enables a rapid and effective search for optimal offloading strategies.

D. Convergence Analysis of VELO

VELO performs stochastic search in the latent space and accepts improvements through a greedy selection rule. Let $\mathcal{X} \subset \mathbb{R}^{d_x}$ denote the feasible solution set in the original decision space and assume $\mathcal{X}$ is nonempty and compact. Let $f : \mathcal{X} \to \mathbb{R}$ be the objective function (fitness) and assume f is bounded below on $\mathcal{X}$, *i.e.*, $\inf_{x \in \mathcal{X}} f(x) > -\infty$. Denote by $x_{\text{best}}^{(n)}$ the best individual in the population at iteration n. Since VELO employs elitist selection, $f(x_{\text{best}}^{(n+1)}) \le f(x_{\text{best}}^{(n)})$ holds for all n, implying that $\{f(x_{\text{best}}^{(n)})\}_{n \ge 0}$ is a non-increasing sequence. Together with the lower-boundedness of f, it follows that $\{f(x_{\text{best}}^{(n)})\}$ converges to a finite limit value. To characterize global convergence in probability, consider the latent space $\mathcal{Z} \subset \mathbb{R}^{d_z}$ induced by the VAE encoder–decoder pair. Assume the decoder $D(\cdot)$ is continuous on $\mathcal{Z}$ and maps any latent vector to a feasible solution, *i.e.,* $D(z) \in \mathcal{X}$ for all $z \in \mathcal{Z}$. Moreover, assume the hybrid mutation in VELO satisfies an irreducibility condition: for any $z \in \mathcal{Z}$ and any open ball $\mathcal{B}_\epsilon(\tilde{z}) \subset \mathcal{Z}$, there exists a constant $\delta(\epsilon) > 0$ such that the probability of generating a trial vector $u_z \in \mathcal{B}_\epsilon(\tilde{z})$ from z in one iteration is at least $\delta(\epsilon)$. This condition is standard for stochastic evolutionary search and is ensured when (i) the crossover rate is strictly between 0 and 1, and (ii) the Lévy flight perturbation is applied with a strictly positive probability $p_l(n)$ for infinitely many iterations.

Let $\mathcal{X}^\star = \arg\min_{x \in \mathcal{X}} f(x)$ be the set of global optima and let $\mathcal{N}_\epsilon(\mathcal{X}^\star)$ denote its ϵ-neighborhood. Under the above assumptions, VELO induces a Markov process over populations with a nonzero probability of visiting any neighborhood of $\mathcal{X}^\star$. By the irreducibility of the latent-space sampling and the greedy acceptance of improvements, the probability that the best-so-far solution enters $\mathcal{N}_\epsilon(\mathcal{X}^\star)$ approaches one as $n_{\max} \to \infty$, *i.e.*,

$$\lim_{n_{\max} \to \infty} \Pr\left(x_{\text{best}}^{(n_{\max})} \in \mathcal{N}_\epsilon(\mathcal{X}^\star)\right) = 1, \quad \forall \epsilon > 0. \tag{15}$$

Therefore, VELO is globally convergent in probability in the sense that, given unlimited iterations, it can reach an arbitrarily small neighborhood of the global optimum with probability one, while the elitist selection guarantees monotonic improvement of the best objective value across iterations.

E. Computational Complexity of VELO

From a computational perspective, VELO framework consists of an offline representation learning stage and an online evolutionary optimization stage. The offline training of the VAE incurs a one-time computational cost of $O(E_{\text{vae}} N C_{\text{vae}})$, where N and E_{vae} denote the size of the training dataset and the number of training epochs, respectively. During the online optimization stage, VELO evolves a population of size p over $n_{\max}$ iterations. For each individual, the dominant operations include latent-space mutation and crossover with complexity $O(d_z)$, VAE decoding with cost $O(C_D)$, and fitness evaluation with cost $O(C_f)$. As a result, the overall online computational complexity of VELO is $O\big(n_{\max}\, p\, (d_z + C_D + C_f)\big)$. In practical computation offloading scenarios, the fitness evaluation typically dominates the computational burden ($C_f \gg d_z, C_D$), implying that the effective complexity is primarily determined by the number of fitness evaluations, *i.e.*, $O(n_{\max}\, p\, C_f)$, while the latent-space optimization significantly reduces the algorithmic overhead compared with direct search in the original high-dimensional decision space.

V. Experimental Evaluation

A. Simulation Setup and Task Generation

The performance of the proposed VELO framework is evaluated in a simulated three-tier VEC environment. Vehicles generate computational tasks that can be executed locally on the OBU, offloaded to a RSU, or offloaded to a remote cloud server. The objective of all compared algorithms is to determine the optimal execution location of each task to minimize the total energy consumption defined in Section III. Table II summarizes the simulation settings. The number of tasks N varies in $\{50, 100, 150, 200\}$ to evaluate scalability. For each task i, the computational workload C_i and data size D_i are randomly sampled from predefined intervals to emulate heterogeneous VEC services. The energy coefficients η_{v}, η_{e}, η_{c}, ϵ_{e}, and ϵ_{c} follow uniform sampling to represent fluctuating hardware efficiency and dynamic network conditions. To partially reflect real-world uncertainties, the randomized sampling of task parameters and energy coefficients implicitly captures the effects of network fluctuations and heterogeneous resource conditions across different execution layers. Nevertheless, the current simulation focuses on static task instances and does not explicitly model time-varying network bandwidth, server-side

resource contention, or inter-task dependency relationships. For each task scale, all algorithms operate on the same set of generated task instances to ensure fair comparison. Results are reported as averaged values over multiple independent runs with distinct random initializations.

TABLE II Simulation Parameter Settings

Category	Parameter	Value
Problem Scale	N	[50, 100, 150, 200]
Task Attributes	C_i	rand(128, 1024) K-cycles
	D_i	rand(1024, 65536) K-bits
Energy Coefficients	η_{v}	rand(10^{-7}, 5×10^{-5}) J/cycle
	η_{e}	rand(10^{-8}, 5×10^{-6}) J/cycle
	η_{c}	rand(10^{-9}, 5×10^{-7}) J/cycle
	ϵ_{e}	rand(10^{-11}, 10^{-10}) J/bit
	ϵ_{c}	rand(10^{-10}, 10^{-9}) J/bit
Heuristic Algorithm	α_l	0.3
	β_l	7
	$n_{\max}$	500
	Population size	50
	Crossover rate	0.9
VAE Network	Dim_0	25
	Dim_1	64
	Activation	ReLU
	Optimizer	Adam (lr=10^{-3})
	Batch size	128
	Epochs	200

* rand(a,b) denotes uniform sampling from interval $[a, b]$.

B. Evaluation Metrics

Three metrics are used to comprehensively evaluate the quality and practicality of task offloading strategies:

- **Total energy consumption:**

$$E_t = \sum_{i=1}^{N} E_i, \tag{16}$$

  which serves as the primary optimization objective.
- **Convergence performance:** the evolution of E_t over optimization iterations, reflecting convergence speed and stability of the algorithm.
- **Runtime overhead:** the wall-clock computational time under different task scales, indicating algorithmic efficiency and deployment feasibility.

C. Baselines and VELO Variants

VELO is compared against four widely adopted metaheuristic and evolutionary optimization methods:

- **STORA** [25, 26]: A self-adaptive teaching-learning-based optimizer enhanced with RBF surrogates and sparse autoencoders, which achieves strong performance on generic high-dimensional benchmarks.
- **GWO** [27, 28]: Grey Wolf Optimizer, modeling hierarchical hunting mechanisms. It features simple parameterization and low computational overhead. Its direct search in the original high-dimensional decision space tends to yield slower convergence and weaker robustness for large task populations.
- **IVY** [29]: An ivy-growth-inspired optimizer balancing exploration and exploitation, whose plant-growth-inspired dynamics improve global search but lack mechanisms for dimensionality reduction or latent-space guidance.
- **SAEO** [22]: A surrogate-assisted autoencoder-embedded evolutionary algorithm for high-dimensional expensive optimization, leveraging autoencoders and surrogates.

All baseline algorithms are adapted to the same discrete, ternary task offloading decision space as VELO, where each task is assigned to one of three execution locations: the vehicle OBU, an RSU, or the remote cloud server. For baselines originally designed for continuous or high-dimensional search spaces, a unified discretization and action-mapping strategy is employed to map their candidate solutions to valid ternary offloading decisions, ensuring a level playing field for comparison.

To verify the contributions of VELO components, three ablation variants are evaluated:

- **VELO(AE)**: replaces VAE with a standard autoencoder.
- **VELO(DE)**: removes Lévy flight and performs standard DE search in the latent space.
- **VELO(LF)**: applies step distribution without an exponential decay component.

All methods use comparable population sizes and identical maximum iteration budgets to ensure fairness. In addition, all methods are evaluated on identical task instances and repeated over multiple independent runs with different random seeds to account for stochastic variability.

D. Results and Discussion

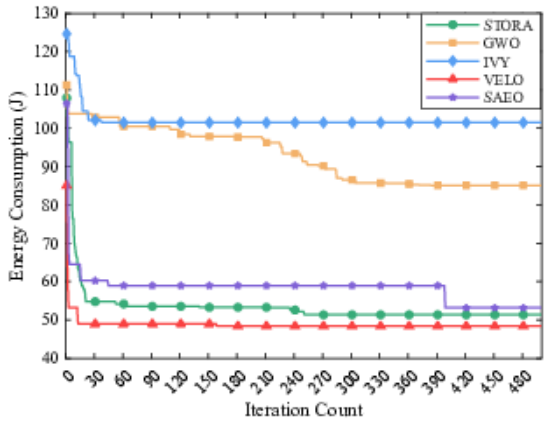

Fig. 2. Convergence performance comparison (N=100).

Fig. 2 shows that algorithms using latent-space dimensionality reduction (SAEO, STORA, and VELO) converge significantly faster and achieve lower final energy than methods without dimensionality reduction. VELO reaches approximately 48 J within the first 30 iterations and maintains stable convergence afterward.

Fig. 3 evaluates scalability. VELO consistently yields the lowest energy consumption across all task sizes, and its slower

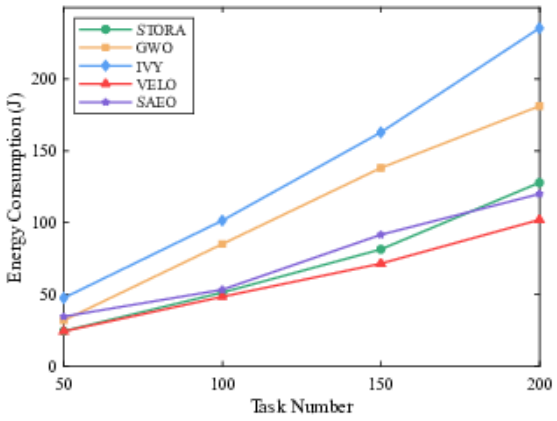

Fig. 3. Energy consumption vs. number of tasks.

growth slope indicates stronger robustness under increasing computational demand, making it suitable for large-scale VEC environments.

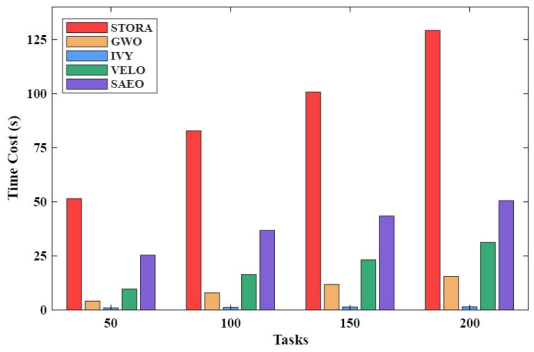

Fig. 4. Time cost analysis.

Fig. 4 presents runtime overhead. Although GWO and IVY incur lower computational cost, their inferior energy optimization limits practical applicability. VELO achieves a better balance between efficiency and solution quality and outperforms both SAEO and VELO(AE), confirming the benefit of using VAE to obtain smoother latent representations.

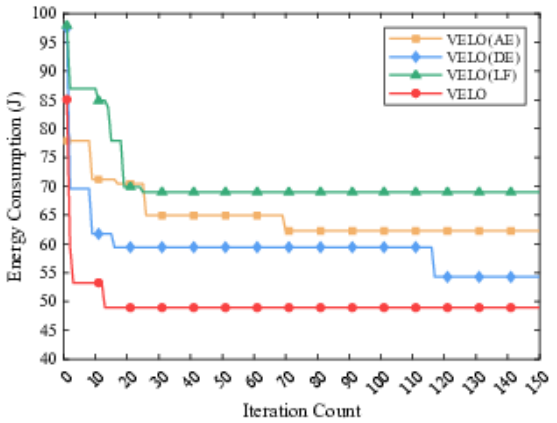

Fig. 5. Ablation experiment results.

The ablation comparison in Fig. 5 shows that VELO(AE) converges to a higher energy level (around 54 J), illustrating the benefit of the structured latent space obtained via VAE. VELO(DE) and VELO(LF) exhibit degraded performance, indicating that the exponentially decaying Lévy flight mechanism is essential for balancing global search and local refinement during different optimization phases. These findings confirm that both VAE-based dimensionality reduction and the enhanced DE strategy are critical to VELO's optimization capability.

VI. Conclusions

This work proposes a novel, energy-aware computational offloading framework named Variational Autoencoder-Enhanced Lévy Differential Evolution Offloader (VELO). VELO integrates a variational autoencoder (VAE) for robust dimensionality reduction and an enhanced differential evolution algorithm. VAE effectively transforms the high-dimensional offloading problem into a structured, continuous latent space. DE algorithm enhanced with an exponentially decaying Lévy flight strategy intelligently balances global exploration and local fine-tuning to rapidly converge on high-quality solutions. Experimental results confirm that VELO significantly reduces the overall energy footprint of vehicular edge computing operations. This work presents a promising solution for alleviating the operational burden of vehicle devices and contributes to the broader goal of developing low-carbon, energy-efficient intelligent transportation systems. For future work, we plan to enhance the model's realism by explicitly incorporating finite computational capacities of edge and cloud servers, so that offloading decisions are optimized under practical resource constraints. This can be achieved by introducing server-side capacity limits and load-aware feasibility constraints into the optimization formulation. Moreover, dynamic network factors such as time-varying bandwidth and stochastic traffic patterns will be integrated to capture non-stationary vehicular environments. We also plan to extend the current binary offloading scheme to support partial and split task offloading through fine-grained task decomposition, enabling better support for latency-sensitive and computation-intensive applications.

References

[1] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16672–16683, May 2024.

[2] H. Yuan, J. Bi, Z. Wang, J. Yang, and J. Zhang, "Partial and Cost-Minimized Computation Offloading in Hybrid Edge and Cloud Systems," *Expert Systems with Applications*, vol. 250, pp. 1–13, Sep. 2024.

[3] L. Liu, M. Zhao, M. Yu, M. A. Jan, D. Lan, and A. Taherkordi, "Mobility-Aware Multi-Hop Task Offloading for Autonomous Driving in Vehicular Edge Computing and Networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 2, pp. 2169–2182, Feb. 2023.

[4] C. Cheng, L. Zhai, X. Zhu, Y. Jia, and Y. Li, "Dynamic Task Offloading and Service Caching Based on Game Theory in Vehicular Edge Computing Networks," *Computer Communications*, vol. 224, pp. 29–41, Aug. 2024.

[5] K. Tan, L. Feng, G. Dán, and M. Törngren, "Decentralized Convex Optimization for Joint Task Offloading and Resource Allocation of Vehicular Edge Computing Systems," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 12, pp. 13226–13241, Dec. 2022.

[6] B. Salehi, G. Reus-Muns, D. Roy, Z. Wang, T. Jian, J. Dy, S. Ioannidis, and K. Chowdhury, "Deep Learning on Multimodal Sensor Data at the Wireless Edge for Vehicular Network," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 7, pp. 7639–7655, Jul. 2022.
[7] D. Zhang, W. Shi, M. St-Hilaire, and R. Yang, "Multiaccess Edge Integrated Networking for Internet of Vehicles: A Blockchain-Based Deep Compressed Cooperative Learning Approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 11, pp. 21593–21607, Nov. 2022.
[8] L. Zhao, T. Li, E. Zhang, Y. Lin, S. Wan, A. Hawbani, and M. Guizani, "Adaptive Swarm Intelligent Offloading Based on Digital Twin-Assisted Prediction in VEC," *IEEE Transactions on Mobile Computing*, vol. 23, no. 8, pp. 8158–8174, Aug. 2024.
[9] J. Lin, S. Huang, H. Zhang, X. Yang, and P. Zhao, "A Deep-Reinforcement-Learning-Based Computation Offloading with Mobile Vehicles in Vehicular Edge Computing," *IEEE Internet of Things Journal*, vol. 10, no. 17, pp. 15501–15514, Sep. 2023.
[10] J. Wu, M. Tang, C. Jiang, L. Gao, and B. Cao, "Cloud-Edge–End Collaborative Task Offloading in Vehicular Edge Networks: A Multilayer Deep Reinforcement Learning Approach," *IEEE Internet of Things Journal*, vol. 11, no. 22, pp. 36272–36290, Nov. 2024.
[11] H. Tang, H. Wu, G. Qu, and R. Li, "Double Deep Q-Network Based Dynamic Framing Offloading in Vehicular Edge Computing," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1297–1310, May 2023.
[12] C. Li, F. Liu, B. Wang, C. L. P. Chen, X. Tang, J. Jiang, and J. Liu, "Dependency-Aware Vehicular Task Scheduling Policy for Tracking Service in VEC Networks," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 3, pp. 2400–2414, Mar. 2023.
[13] J. Shi, J. Du, Y. Shen, J. Wang, J. Yuan, and Z. Han, "DRL-Based V2V Computation Offloading for Blockchain-Enabled Vehicular Networks," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 3882–3897, Jul. 2023.
[14] F. Chai, Q. Zhang, H. Yao, X. Xin, R. Gao, and M. Guizani, "Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 7783–7795, Jun. 2023.
[15] H. Jiang, X. Dai, Z. Xiao, and A. Iyengar, "Joint Task Offloading and Resource Allocation for Energy-Constrained Mobile Edge Computing," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4000–4015, Jul. 2023.
[16] Y. Li, X. Zhang, B. Lei, Q. Zhao, M. Wei, Z. Qu, and W. Wang, "Computation Rate Maximization for Wireless-Powered Edge Computing with Multi-User Cooperation," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 965–981, 2024.
[17] Y. Yang, Y. Chen, K. Li and J. Huang, "Carbon-Aware Dynamic Task Offloading in NOMA-Enabled Mobile Edge Computing for IoT," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 15723–15734, May. 2024.
[18] Z. Song, M. Xie, J. Luo, T. Gong and W. Chen, "A Carbon-Aware Framework for Energy-Efficient Data Acquisition and Task Offloading in Sustainable AIoT Ecosystems," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 39103–39113, Dec. 2024.
[19] X. Hu and Y. Huang, "Deep Reinforcement Learning Based Offloading Decision Algorithm for Vehicular Edge Computing," *PeerJ Computer Science*, vol. 8, Art. no. e1126, 2022.
[20] W. Shi, L. Chen, and X. Zhu, "Task Offloading Decision-Making Algorithm for Vehicular Edge Computing: A Deep-Reinforcement-Learning-Based Approach," *Sensors*, vol. 23, no. 17, Art. no. 7595, 2023.
[21] M. Cui, L. Li, M. Zhou, J. Li, A. Abusorrah, and K. Sedraoui, "A Bi-Population Cooperative Optimization Algorithm Assisted by an Autoencoder for Medium-Scale Expensive Problems," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 11, pp. 1952–1966, Nov. 2022.
[22] M. Cui, L. Li, M. Zhou, and A. Abusorrah, "Surrogate-Assisted Autoencoder-Embedded Evolutionary Optimization Algorithm to Solve High-Dimensional Expensive Problems," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 676–689, Aug. 2022.
[23] J. Zhai, J. Bi, H. Yuan, M. Wang, J. Zhang, Y. Wang, and M. Zhou, "Cost-Minimized Microservice Migration With Autoencoder-Assisted Evolution in Hybrid Cloud and Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 40951—40967, Dec. 2024.
[24] J. Zhai, J. Bi, H. Yuan, J. Zhang, and R. Buyya, "Energy-Efficient and Latency-Aware Task Offloading for Industrial Cloud-Edge Systems With Heterogeneous CPUs and GPUs," *IEEE Internet of Things Journal*, vol. 12, no. 13, pp. 25757–25772, Jul. 2025.
[25] J. Bi, Z. Wang, H. Yuan, J. Qiao, J. Zhang, and M. Zhou, "Self-Adaptive Teaching-Learning-Based Optimizer with Improved RBF and Sparse Autoencoder for Complex Optimization Problems," *Proc. IEEE Int. Conf. Robotics and Automation*, May 2023, pp. 7966–7972.
[26] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Self-Adaptive Teaching-Learning-Based Optimizer with Improved RBF and Sparse Autoencoder for High-Dimensional Problems," *Information Sciences*, vol. 630, pp. 463–481, Jun. 2023.
[27] W. Liao, L. Zhou, C. Zhang, D. Wang, J. Zhang, and L. Guo, "A Method for Discriminating the Moisture Status of OIP Bushing Based on Dissado-Hill and GWO-HMM Model," *IEEE Transactions on Industry Applications*, vol. 58, no. 2, pp. 1512–1520, Mar./Apr. 2022.
[28] J. Bi, J. Zhai, H. Yuan, Z. Wang, J. Qiao, J. Zhang, and M. Zhou, "Multi-swarm Genetic Gray Wolf Optimizer with Embedded Autoencoders for High-dimensional Expensive Problems," *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2023, pp. 7265–7271.
[29] M. Ghasemi, M. Zare, P. Trojovský, R. V. Rao, E. Trojovská, and V. Kandasamy, "Optimization Based on the Smart Behavior of Plants with Its Engineering Applications: Ivy Algorithm," *Knowledge-Based Systems*, vol. 295, 111850, Jul. 2024.

Jiahui Zhai (Graduate Student Member, IEEE) received the B.S. degree in Software Engineering from Zhengzhou University, Henan, China, in 2019, and the M.S. degree in Software Engineering from Beijing University of Technology, Beijing, China, in 2022. He is currently working toward the Ph.D. degree with the College of Computer Science, Beijing University of Technology, Beijing, China. From 2024 to 2025, he visited University College Dublin, Ireland, as a visiting scholar. His current research interests include cloud/edge computing, edge intelligence, and distributed machine learning. He was the recipient of the Best Paper Award-Finalist in the 18th IEEE International Conference on Networking, Sensing and Control (ICNSC).

Yaxi Yang is currently pursuing the M.S. degree with the School of Computer Science, Beijing University of Technology, China. Before that, she received the B.S. degree in Software Engineering in 2025 from Shanxi University, China. Her research interests include time series analysis, cloud computing, intelligent optimization algorithms, and deep learning.

Ziqi Wang is currently a Master student in the College of Computer Science, Beijing University of Technology, Beijing, China. Before that, he received his B.E. degree in Internet of Things from Beijing University of Technology in 2022. His research interests include cloud computing, task scheduling, intelligent optimization algorithms and machine learning.

Junqi Zhang received his B.S. and Ph.D. degrees from the Department of Automation at Tsinghua University in 2015 and 2021, respectively. From 2021 to 2022, he worked as an algorithm engineer at JD.com, focusing on ads retrieval. From 2022 to 2023, he transitioned to Qiyuan Lab, where he assumed a research role with a focus on vision-language models. His research interests include information retrieval, recommender systems, multimodal models, self-supervised learning, and explainable AI.