Energy-Efficient Task Allocation for Smart Traffic Signal Control in Edge-Cloud Hierarchies

Ziqi Wang, Jiayi Li, Haitao Yuan, and Jing Bi

***Abstract*—In smart traffic signal control, large-scale urban road networks increasingly rely on distributed sensing, communication, and computation resources deployed at intersections, roadside facilities, and remote cloud platforms. These heterogeneous resources must collaboratively process massive numbers of delay-sensitive and computation-intensive tasks under stringent energy and security requirements. This work investigates energy-minimized task allocation in a three-tier cloud-edge-device architecture that includes field traffic signal controllers, roadside computing units, and a centralized cloud data center. A dual-mode optimization model is formulated that integrates a general unconstrained offloading mode with a security-aware offloading mode, where highly sensitive tasks are forced to remain local. The objective is to minimize total energy consumption by jointly modeling computation latency, communication latency, dynamic processing energy, and transmission energy, while respecting resource capacity and delay bounds. To efficiently search this non-convex, constrained solution space, an adaptive constraint-enhanced metaheuristic algorithm is developed, incorporating dynamic penalty adjustment, hybrid early termination, adaptive population regulation, and pruning strategies for infeasible solutions. Experimental results on large-scale synthetic traffic workloads demonstrate that the proposed algorithm achieves lower energy consumption and faster convergence than several representative metaheuristic baselines, and can be effectively applied to resource scheduling in smart traffic control systems.**

***Key Words*—Cloud–edge–device collaboration, dual-mode offloading, energy optimization, intelligent traffic control.**

I. Introduction

THE deep integration of information technology with urban governance has established the Internet of Things (IoT) as a cornerstone for smart cities [1]. By densely interconnecting networks of sensing nodes, intelligent terminals, and computational resources distributed throughout urban environments, IoT constructs an intelligent neural network capable of dynamically sensing, transmitting, and processing information [2], transforming traditional paradigms towards data-driven fine-grained governance and real-time response [3].

With explosive growth in sensing devices and evolving computing paradigms, data processing complexity in smart city applications is increasing exponentially [4]. Particularly in critical domains like traffic guidance, environmental monitoring, and emergency response. Terminal devices face significant computational pressure when handling massive data volumes. However, large-scale terminal deployments are constrained by limited computational resources and unstable energy consumption, hindering efficient execution of complex tasks [5] and causing substantial energy wastage [6]. Computation offloading technology [7–9] provides a core solution: By deploying edge computing infrastructure at key urban nodes integrated with cloud data centers, computation-intensive tasks can be intelligently distributed across terminal-edge-cloud hierarchies [10]. However, offloading involves wide-area data transmission, introducing non-negligible communication overhead in delay and bandwidth [11]. The central challenge is thus dynamically determining optimal offloading locations for computational tasks based on their characteristics and real-time system states under resource constraints [12].

Motivated by this, we design a computation offloading strategy within a three-tier cloud-edge-device architecture to minimize energy consumption while guaranteeing delay requirements and security constraints [13, 14]. We first construct an intelligent traffic-light-controlled architecture with multiple edge nodes and terminals [15]. Next, we formulate an energy minimization problem incorporating device processing capabilities and transmission overheads [16]. Finally, we propose an enhanced plant-inspired metaheuristic, referred to as the IVY algorithm [17], and further extend it into the Adaptive Constraint Enhanced IVY (ACE-IVY) algorithm, featuring key innovations: Levy flight-inspired dynamic penalty optimization, adaptive population size scaling for varying task volumes, a hybrid early stopping mechanism for computational economy, and an alpha-beta pruning-like strategy for efficiency [18]. Unlike conventional metaheuristics that employ static penalties and fixed population sizes, ACE-IVY integrates dynamic penalty adjustment and adaptive population regulation to enhance feasibility handling and search efficiency under constrained cloud–edge optimization scenarios. Experimental results demonstrate ACE-IVY's efficacy in enhancing system performance within our architecture. The main contributions of this work are as follows:

- We establish an integrated traffic signal system architecture that supports both general offloading and security-aware constrained offloading.
- We formulate a constrained optimization model that jointly considers computation latency, communication overhead, energy consumption, resource limits, and task-

Manuscript received December 7, 2025; revised December 14 and December 21, 2025; accepted December 24, 2025. This article was recommended for publication by Associate Editor Shujin Qin upon evaluation of the reviewers' comments.

Z. Wang, J. Li and J. Bi are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: bijing@bjut.edu.cn).

H. Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. (e-mail: yuan@buaa.edu.cn).

Corresponding author: Jing Bi.

level security requirements.
- We develop an adaptive constraint-enhanced IVY algorithm with dynamic penalty adjustment, population scaling, early stopping, and pruning strategies, achieving improved convergence and energy efficiency.

Sections are organized as follows: Section II reviews the related research on cloud–edge task offloading, constrained metaheuristic optimization, and adaptive constraint-handling techniques. Section III introduces the intelligent traffic signal architecture and establishes the latency, energy, and constraint models that form the basis of the optimization problem. Section IV describes the proposed ACE-IVY algorithm in detail, including its adaptive mechanisms, pruning strategies, and experimental evaluation under large-scale cloud-edge-terminal workloads. Section V concludes the paper and outlines potential extensions toward learning-driven offloading strategies and broader cyber-physical applications.

II. Related Work

This section reviews the related work from two aspects, including cloud-edge task offloading with energy-aware scheduling, and constrained optimization methods.

A. Cloud–Edge Task Offloading and Energy-Aware Scheduling

Cloud–edge collaborative computing has become essential for delay-sensitive and computation-intensive applications across Internet of Things (IoT), vehicular systems, and industrial automation. As multiple edge nodes and cloud servers jointly provide computing resources, an efficient scheduling and offloading strategy must decide how tasks are distributed across edge devices and cloud datacenters. The goal is to satisfy heterogeneous Quality of Service (QoS) requirements while minimizing system overheads such as latency and energy consumption. Several recent studies have focused on improving task offloading efficiency under different network, mobility, and resource conditions.

Zhang *et al.* [19] conduct a comprehensive survey on computation offloading categorized by task types, demonstrating that workload heterogeneity substantially affects scheduling complexity and system performance. However, their study mainly provides taxonomic insights without offering unified optimization mechanisms suitable for highly dynamic cloud-edge environments. Deep reinforcement learning is introduced to support adaptive task offloading under uncertainties. Chen *et al.* [20] propose a multi-agent DRL-based allocation framework that jointly determines offloading and computational resource decisions, showing significant gains in delay and QoE. However, the approach relies on centralized training and accurate environment observability, which limits its deployment in real-world systems with partial information and resource constraints.

Energy-aware task offloading under mobility and wireless dynamics has also attracted attention. Li *et al.* [21] study energy-efficient computation offloading in vehicular edge computing with speed-adjustment mechanisms, revealing that mobility-induced variations greatly complicate energy minimization. However, their mobility modeling remains simplified and does not represent heterogeneous industrial workloads or multi-tenant edge environments. In industrial MEC systems, Chi *et al.* [22] design ATOM, a two-stage hybrid matching framework that improves delay performance and load balancing. However, ATOM primarily focuses on latency and matching efficiency, offering limited treatment of joint latency-energy optimization under bursty workloads. Existing work demonstrates the importance of coordinated resource scheduling across cloud and edge layers. However, many approaches either rely on learning models with high computational overhead or simplify system constraints excessively, making it difficult to balance energy, latency, and feasibility in practical deployments. These limitations motivate the development of more flexible constrained optimization frameworks for cloud-edge task scheduling.

B. Constrained Optimization for Cloud-Edge Systems

Cloud-edge task scheduling often leads to nonconvex, high-dimensional constrained optimization problems involving latency limits, energy budgets, and resource coupling. Metaheuristic optimization methods have therefore been widely adopted due to their scalability and flexibility. Several studies analyze constraint-handling mechanisms and propose improved optimization strategies suitable for complex constrained environments.

Liang *et al.* [23] provide a comprehensive survey on evolutionary constrained multiobjective optimization (ECMO), summarizing mainstream constraint-handling techniques such as feasibility rules, stochastic ranking, penalty functions, and multiobjective reformulation. They point out that constraint handling frequently becomes the dominant bottleneck, and many techniques depend heavily on manual parameter tuning, reducing robustness across diverse constraint structures. To enhance optimization efficiency, Ming *et al.* [24] propose a multitasking constrained multiobjective optimization framework that leverages knowledge transfer among related tasks. However, the method requires carefully designed auxiliary tasks, which may be difficult to construct in cloud-edge systems with irregular or partially known constraints.

Li *et al.* [25] develop a competitive–cooperative evolutionary framework that integrates multiple constraint-handling operators and adaptively selects effective mechanisms based on historical performance. Although such ensembles improve robustness, they also introduce significant computational overhead, making them unsuitable for scenarios with costly objective evaluations such as full-stack cloud-edge simulations. Adaptive constraint-handling mechanisms have also been investigated to dynamically balance feasibility pressure and objective improvement. Ming *et al.* [26] propose a decomposition-based constraint-handling technique that adaptively tunes reference points and feasibility pressure to enhance convergence and feasible-solution exploration. Nevertheless, its tight coupling with decomposition-based MOEAs reduces applicability to single-objective or mixed-integer cloud-edge scheduling problems.

Furthermore, Li *et al.* [27] introduce a task-clone-based multitasking optimization framework that decomposes complex constrained multiobjective problems into multiple cloned

tasks to enhance parallel constraint exploration. However, cloning increases search dimensionality and computational cost, potentially limiting scalability under strict time and energy budgets in cloud-edge systems. In summary, existing metaheuristic and adaptive constraint-handling methods provide valuable foundations for solving constrained cloud-edge optimization problems. However, they often suffer from high computational complexity, limited adaptability, or weak scalability under strong constraints. These observations motivate the development of the ACE-IVY framework, which integrates adaptive constraint enhancement and dynamic population management tailored specifically for energy-aware task offloading.

III. Problem Presentation

To optimize the cost-minimization problem, Fig. 1 illustrates an intelligent traffic control system with cloud–edge–device collaboration. This architecture connects Traffic Signal Units (TSUs) to Roadside Units (RSUs), where each RSU may serve multiple TSUs. The Cloud Data Center (CDC) coordinates all RSUs, establishing bidirectional communication between TSUs, RSUs, and the CDC. The system contains N RSUs and M TSUs, all with limited computational capacity. The model supports two distinct task allocation strategies: Unconstrained Offloading for general tasks, and Security-constrained Offloading for operations requiring enhanced security measures.

In the Unconstrained Offloading mode, the computational tasks of a single TSU can be allocated in arbitrary proportions. To represent task offloading to the TSU, RSU, or CDC, three variables α_i,β_{ij},γ_i are introduced, α_i denotes the proportion of tasks computed locally on the TSU i. β_{ij} denotes the proportion of TSU i's tasks offloaded to the RSU j for computation. γ_i denotes the proportion of TSU i's tasks offloaded to the CDC for computation. For any TSU, its total computational task allocation must satisfy,the sum of local and offloaded portions cannot exceed one, *i.e.*,

$$\alpha_i+\sum_{j=1}^{N}\beta_{ij}+\gamma i\leqslant 1 \tag{1}$$

In the Security-aware Offloading mode, to ensure the security of computational tasks, a safety threshold $\mathfrak{s}$ is introduced. When the safety requirement coefficient of TSU i's task exceeds this safety threshold $\mathfrak{s}$, then $\alpha_i = 1$, meaning the task must be entirely executed locally on the TSU i.

A. Latency Model

The latency of this cloud-edge-device collaborative architecture consists of three components related to the TSU, RSU, and CDC, respectively.

1) Latency of TSUs: Let $\mathcal{L}_i$ represent the local execution latency of computational tasks on the TSU i. It is given by:

$$\mathcal{L}_i=\alpha_i\left(\frac{T_i}{V_i}\right) \tag{2}$$

where T_i denotes the required computational workload for TSU i's tasks, for example the number of task operations; and V_i denotes the computational speed of its processing unit, such as tasks completed per unit time by the CPU.

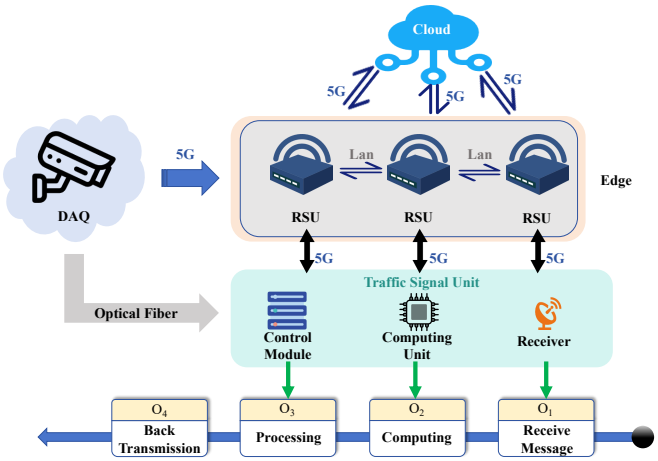

Fig. 1. Architecture of the cloud-edge-device collaborative traffic signal control system, illustrating task generation, communication links, and computation modules.

2) Latency of RSUs: Let $\mathcal{E}'_{ij}$ represent the execution time of tasks offloaded to the RSU j, and let v_j denote the computational speed of the processor in the RSU j. *i.e.*,

$$\mathcal{E}'_{ij}=\beta_{ij}\left(\frac{T_i}{v_j}\right) \tag{3}$$

Let $\mathcal{E}''_{ij}$ represent the transmission latency between the TSU i and RSU j. it is obtained as:

$$\mathcal{E}''_{ij}=\beta_{ij}\left(\frac{\hat{D_{ij}}}{\hat{S_{ij}}}\right)+\beta_{ij}\left(\frac{\check{D_{ij}}}{\check{S_{ij}}}\right) \tag{4}$$

where the first term corresponds to the downlink transmission latency from RSU to TSU, and the second term corresponds to the uplink transmission latency from TSU to RSU. $\hat{D_{ij}}$ and $\check{D_{ij}}$ represents the downlink and uplink data size transmitted from RSU j to TSU i and from TSU i to RSU j. Let $\hat{S_{ij}}$ and $\check{S_{ij}}$ represent the downlink and uplink transmission rates respectively. The calculation formula for $\hat{S_{ij}}$ is as follows:

$$\hat{S_{ij}}=\log_2\left\{1+\frac{\hat{SNR}_{ij}(\hat{\eta_{ij}})}{\hat{\mu_{ij}}}\right\} \tag{5}$$

$$\check{S_{ij}}=\log_2\left\{1+\frac{\check{SNR}_{ij}(\check{\eta_{ij}})}{\check{\mu_{ij}}}\right\} \tag{6}$$

where $\hat{SNR}_{ij}$ represents the uplink channel signal-to-noise ratio, and $\check{SNR}_{ij}$ represents the downlink channel signal-to-noise ratio. $\hat{\eta_{ij}}$ denotes the uplink data loss coefficient, while $\check{\eta_{ij}}$ denotes the downlink data loss coefficient. $\hat{\mu_{ij}}$ indicates the uplink channel occupancy rate, and $\check{\mu_{ij}}$ indicates the downlink channel occupancy rate.

The latency related to RSUs is calculated as follows:

$$\mathcal{E}_{ij}=\mathcal{E}'_{ij}+\mathcal{E}''_{ij} \tag{7}$$

3) Latency of CDC: Let C'_i represent the execution time of tasks offloaded to the CDC, and υ denote the computational speed of the CDC's CPU. Then:

$$C'_i=\gamma_i\left(\frac{T_i}{\upsilon}\right) \tag{8}$$

Therefore, The total time C_i of processing associated with CDC can be expressed as: $C_i = C_i' + C_i''$, where C_i'' represents the transmission latency between the TSU i and the CDC.

Finally, the total time L_i to complete a task on the TSU i is the maximum value between the computation time of the offloaded portion and the local computation time, *i.e.*,

$$L_i = \max\left\{\mathcal{L}_i, \sum_{j=1}^{N} \mathcal{E}_{ij}, C_i\right\} \tag{9}$$

Additionally, L_i must be less than the task's time constraint ϑ_i, *i.e.*, $L_i < \vartheta_i$

B. Energy Consumption Modeling

1) System-Level Energy Decomposition: The total energy consumption in a cloud-edge computing system is decomposed into: $E_s = E_c + E_t$, where E_c denotes the dynamic computational energy from processing units, and E_t represents the energy consumed in data transmission across networks.

2) Dynamic Computational Energy Model: Dynamic computational energy originates from hardware switching activities: $E_c = C_e \cdot V_d^2 \cdot f \cdot N \cdot t$, where C_e is the effective switching capacitance by pF, V_d is the supply voltage by V, f is the processor frequency by Hz, N is the number of switching transistors, and t is the computation time by s.

From the perspective of task allocation ratios (x, y, z), the dynamic energy of task i is:

$$E_{c,i} = \sum_{\text{node} \in S_u} \alpha_n \cdot \omega_i \cdot r_{n,i} \tag{10}$$

where $S_u = \{TSU, RSU, CDC\}$, α_n is the energy coefficient for node type, ω_i is the computational demand of task i, and $r_{n,i}$ is the allocation ratio for node and task i.

3) Transmission Energy Model: The total transmission energy is given by: $E_t = E_{tx} + E_{rx}$, where E_{tx} is the energy consumption at the sender, and E_{rx} is the energy consumption at the receiver.

The sender energy model is: $E_{tx} = (P_0 + k \cdot d^{\eta} \cdot B) \cdot B/R$, with P_0 as the static power consumption of the transmitter circuit measured by Watt, k as the amplifier coefficient related to transmitter device characteristics, d as the distance between sender and receiver measured by meter, η as the path loss exponent, B as the amount of data transmitted (bit), and R as the data transmission rate (bit/second).The receiver energy model is: $E_{rx} = (P_a + P_p) \cdot B/R$, with P_a and P_p measured by Watt, as the power consumption of the receiver amplifier and processing circuit measured by Watt.

C. Integrated Model

1) Objective Function: Minimizing Total Energy: The goal is to minimize energy via optimal task allocation using the parameters defined in the latency model:

$$f(x) = \min\left[\sum_{i=1}^{N} A_i + \sum_{i=1}^{N} B_i\right] \tag{11}$$

where

$$A_i = \left(\alpha_i \cdot \epsilon_{TSU} + \sum_{j=1}^{M} \beta_{ij} \cdot \epsilon_{RSU} + \gamma_i \cdot \epsilon_{CDC}\right) \cdot \omega_i$$

$$B_i = \left(\sum_{j=1}^{M} \beta_{ij} \cdot \tau_{RSU} + \gamma_i \cdot \tau_{CDC}\right) \cdot S_i,$$

and α_i denotes the proportion of tasks computed locally on TSU i, $0 \le \alpha_i \le 1$, β_{ij} denotes the proportion of TSU i's tasks offloaded to RSU j, $0 \le \beta_{ij} \le 1$, and γ_i denotes the proportion of TSU i's tasks offloaded to CDC, $0 \le \gamma_i \le 1$. Here, $\epsilon_{TSU/RSU/CDC}$ denotes the computational energy per unit task for each node type, $\tau_{RSU/CDC}$ is the transmission energy per unit data, and S_i is the data size of task i.

2) Optimization Constraints: The optimization problem is subject to the following constraints: Edge/Cloud nodes have finite computation resources, expressed as:

$$\sum_{i=1}^{N}\left(\sum_{j=1}^{M} \beta_{ij} \cdot \omega_i\right) \le C_{RSU}, \quad \sum_{i=1}^{N} (\gamma_i \cdot \omega_i) \le C_{CDC} \tag{12}$$

where C_{RSU} and C_{CDC} denote the maximum computational loads for RSUs and CDCs, respectively. For tasks with high security requirements [28], if Sec_i exceeds the threshold $\mathfrak{s}$, then $\beta_{ij} = 0$ and $\gamma_i = 0$.

IV. Adaptive Constraint Enhanced IVY Algorithm (ACE-IVY)

ACE-IVY is proposed to solve the optimization problem in edge computing environments. Four key improvements, adaptive constraint handling, early termination, parameter tuning, and efficient constraint pruning, was promoted to enhance the performance of the original IVY algorithm.

A. Initialization

The population $\vec{I}$ and growth rates Δ are initialized as:

$$I_i = lb + (ub - lb) \times \text{rand}(1, d) \tag{13}$$

$$\Delta_i = (0.1 + 0.2 \times \text{rand}(1, d)) \times (ub - lb) \tag{14}$$

where i denotes individual in one iteration, lb and ub is lower and upper bounds of each dimension, and d denotes the problem dimension.

B. Optimization Process

For each individual i at iteration k:

1) Penalized fitness evaluation: :

$$f_{raw}(I) = \text{obj\_func}(I) \tag{15}$$

$$\text{violation}(I) = \sum_{k} \max(0, g_k(I)) \quad \text{(positive deviation)} \tag{16}$$

$$f_p(I) = f_{raw}(I) + p(k) \times \text{violation}(I) \tag{17}$$

where f_{raw} defines the original output of the object function, g_k is the k-th constraint function, and $p(t)$ is the adaptive penalty coefficient at iteration k. We explicitly formalize the dynamic penalty adjustment mechanism to balance feasibility enforcement and objective optimization.

2) Adaptive penalty coefficient: :

$$p(t)=p_{\min}+(p_{\max}-p_{\min})\times\left(1-\frac{k}{K_{max}}\right)^{\beta} \tag{18}$$

where β controls the decay rate (higher values accelerate penalty reduction), K_{max} is the maximum iteration count, $p_{\min}$ is the minimum penalty, and $p_{\max}$ is the maximum penalty. Initial penalty $p(0) = p_{\max}$ will converge to $p_{\min}$ by the increment of k, until $k = K_{max}$. This formulation enforces strong constraint pressure in early iterations to rapidly guide the population toward the feasible region, while gradually relaxing the penalty to allow fine-grained energy optimization once feasibility is largely satisfied.

3) Growth rate update with momentum: :

$$\Delta_i=0.9\times\Delta_i+(1-0.9)\times\left(\text{rand}^2\odot\mathcal{N}(\mathbf{0},\mathbf{I}_d)\right) \tag{19}$$

where $\mathcal{N}(\mathbf{0},\mathbf{I}_d)$ denotes D-dimensional standard normal random vector, simulating stochastic growth variations, and $\odot$ denotes element-wise multiplication. The momentum term simulates plant growth inertia.

4) Dynamic behavior threshold: :

$$\beta_t=0.5+0.5\times(k/K_{max}) \tag{20}$$

where β_t is the time-dependent threshold coefficient, increasing from 0.5 to 1.0 during optimization.

$$\theta_i=\beta_t\times f_{\text{p}}(I_B) \tag{21}$$

where θ_i is the adaptive behavior selection threshold for individual i, and I_B is the best solution.

5) Neighbor selection: :

$$I_n=\begin{cases}\text{random better individual} & \text{if available}\\ I_B & \text{otherwise}\end{cases} \tag{22}$$

"Better" defined by penalized fitness f_{p}, where I_B is the global best solution, and I_n is the relative neighborly individual.

6) Bio-inspired behavior selection: : $I_i^{'}$ is updated as:

$$\begin{cases}I_i+\alpha\times(|\mathcal{N}(\mathbf{0},\mathbf{I}_d)|\odot(I_n-I_i)+\mathcal{N}(\mathbf{0},\mathbf{I}_d)\odot\Delta_i) & f_{\text{p}}(I_i)<\theta_i\\ I_B\odot(\text{rand}(1,d)+\beta_e\times\mathcal{N}(\mathbf{0},\mathbf{I}_d)\odot\Delta_i) & \text{otherwise}\end{cases} \tag{23}$$

where $\alpha=0.1\times(1-k/K_{max})$ is the climbing coefficient that decreases over time, $\beta_e=0.2\times(1+k/K_{max})$ is the expansion coefficient that increases over time, and $\odot$ denotes element-wise multiplication. Local search enhancement is adopted with 30% probability:

$$I_i^{'}=I_B+0.05\times(ub-lb)\times\mathcal{N}(\mathbf{0},\mathbf{I}_d) \tag{24}$$

7) Intelligent boundary handling: : $I_i^{'}$ is updated as:

$$\begin{cases}lb_d+0.8\times(lb_d-I_i^{'}) & I_i^{'}<lb_d\\ ub_d-0.8\times(I_i^{'}-ub_d) & I_i^{'}>ub_d\\ lb_d+0.02\times(ub_d-lb_d) & \|I_i^{'}-lb_d\|<0.05(ub_d-lb_d)\\ ub_d-0.02\times(ub_d-lb_d) & \|I_i^{'}-ub_d\|<0.05(ub_d-lb_d)\\ I_i^{'} & \text{otherwise}\end{cases} \tag{25}$$

where lb_d and ub_d are dimension-specific bounds.

8) Dynamic population management: :

$$\delta=\text{mean}(\text{std}(I,\text{axis}=0)) \tag{26}$$

where δ is the population diversity metric.

$$\text{Injection condition: } \delta<0.1\times\text{mean}(ub-lb) \tag{27}$$

$$I_{\text{inj}}=I_B+0.1\times(ub-lb)\times\mathcal{N}(\mathbf{0},\mathbf{I}_d) \tag{28}$$

where I_{inj} is the injected individual.

$$n_{\text{inj}}=\max\left(1,\lfloor n_{\text{pop}}\times0.1\rfloor\right) \tag{29}$$

where n_{inj} is the number of individuals to inject.
Population resizing (applied every 100 iterations):

$$n_{\text{pop}}(t)=n_{\text{pop}}(0)\times\left(0.7+0.3\times\left(1-\frac{k}{K_{max}}\right)\right) \tag{30}$$

where $n_{\text{pop}}(t)$ is the current population size.

$$n_{\text{pop}}(t)\geq n_{\min}=20 \quad \text{for } k>0.5K_{max} \tag{31}$$

where $n_{\min}$ is the minimum population size.
When downsizing, elite individuals are preserved based on fitness ranking.

9) Hybrid early stopping: (k=100):

$$\Delta f=\frac{|f_k-f_{k-x}|}{\max(|f_k|,10^{-8})}<10^{-4} \tag{32}$$

where Δf is the relative fitness change over x iterations.

$$\sigma_{dim}=\sqrt{\frac{1}{x}\sum_{k'=k-x+1}^{k}(x_d(k')-\bar{x}_d)^2}<0.01 \quad \forall dim\in d \tag{33}$$

where σ_{dim} is the parameter fluctuation for dimension d.

$$R_s=\frac{\sigma(\{f_{k-x+1},\ldots,f_k\})}{\mu(\{f_{k-x+1},\ldots,f_k\})}<10^{-6}\times(1+k/K_{max}) \tag{34}$$

where R_s is the relative standard deviation of recent best fitness values. The process will be terminated if any condition satisfied when $t > 0.7T$.

10) Pruning strategies: Position update pruning:

$$I_i^{k+1}=\begin{cases}I_i^{'} & f_{\text{p}}(I_i^{'})<f_{\text{p}}(I_i^{t})\\ I_i^{k} & \text{otherwise}\end{cases} \tag{35}$$

where I_i^k denotes the current position of individual i at iteration k, $I_i^{'}$ represents the new candidate position generated by behavior selection, f_{p} is the penalized fitness function defined in equation (17), and I_i^{k+1} is the accepted position for the next iteration. This strategy accepts position updates only when they improve the penalized fitness value. Diversity injection pruning is performed as:

$$I_B^{k+1}=\begin{cases}I_{\text{inj}} & f_{\text{p}}(I_{\text{inj}})<f_{\text{p}}(I_B^k)\\ I_B^k & \text{otherwise}\end{cases} \tag{36}$$

where I_B^k is the current global best solution at iteration k, I_{inj} represents the injected individual generated through diversity enhancement as defined in equation (28), and I_B^{k+1} is the updated global best solution. The global best solution is only updated by this strategy when injected individuals demonstrate

Algorithm 1 ACE-IVY Optimization Algorithm

Input: f, g_k, d, lb, ub, K_{max}, $n_{\text{pop}(0)}$, $n_{\min}$=20
Output: I_B, f_{best}

Initialize I_i and Δ_i with (13)-(14)
Evaluate f_{p} with (15)-(17)
$I_B \leftarrow f_{\text{p}}$, $f_{\text{best}} \leftarrow \min f_{\text{p}}$
for k=1 **to** K_{max} **do**
  Update $p(k)$ with (18)
  Update Δ_i with (19)
  for each i **do**
    Calculate θ_i with (20)-(21)
    Select I_{neighbor} with (22)
    Generate $I_i^{'}$ with 30% probability local search with (23)-(24)
    Apply boundary handling with (25)
    Evaluate $f_{\text{p}}(I_i^{'})$
    Apply position update pruning with (35)
  end for
  Update I_B and f_{best} if improved
  if $k \mod 100$=0 **then**
    Resize population with (30)-(31)
    Compute δ with (26)
    if (27) **then**
      Inject n_{inj} with (28)-(29)
      Update I_B with (36)
    end if
  end if
  Record f_{best} history and parameter history
  if k>0.7K_{max} **and** $k \mod 50$=0 **then**
    if (32) OR (33) OR (34) **then**
      break {Early stopping}
    end if
  end if
end for
return I_B, f_{best}

superior fitness. The complete ACE-IVY process is formalized in Algorithm 1.

Let K_{max} denote the maximum number of iterations, n_{pop} the population size, d the problem dimension, and g_k the set of constraint functions as defined in Algorithm 1. At each iteration k, the algorithm performs population-wise operations including growth update, behavior selection, boundary handling, and pruning, each requiring $O(d)$ arithmetic operations per individual. In addition, the evaluation of the penalized fitness function f_p involves computing the objective function f and constraint violations g_k, resulting in a per-individual cost of $O(d + |g_k|)$. Therefore, the dominant computational cost per iteration is

$$O\big(n_{\text{pop}} \cdot (d + |g_k|)\big), \tag{37}$$

and the overall time complexity of ACE-IVY is given by

$$O\big(K_{max} \cdot n_{\text{pop}} \cdot (d + |g_k|)\big). \tag{38}$$

The proposed adaptive mechanisms, including dynamic penalty adjustment, population resizing, diversity injection, and early stopping, are triggered periodically or conditionally and involve only lightweight scalar operations or partial population updates. As a result, their computational overhead is negligible compared with the fitness evaluation step and does not affect the overall asymptotic complexity. Consequently, ACE-IVY exhibits the same time complexity order as the original IVY algorithm.

C. Experimental Evaluation

This section presents the experimental setup and results for evaluating ACE-IVY. The objective is to establish a reproducible and transparent assessment framework. The evaluation consists of seven components: (1) datasets and task generation, (2) implementation details, (3) evaluation metrics, (4) baseline algorithm comparison, (5) convergence analysis, (6) runtime analysis, and (7) energy efficiency analysis under cloud-edge task offloading scenarios.

1) Datasets and Task Generation: Since large-scale real-world cloud-edge task offloading traces are not publicly available, synthetic task streams are generated following workload patterns inspired by traffic-oriented simulators such as CityFlow. Task sizes S_i, computational demand T_i, and security sensitivity Sec_i are sampled within predefined parameter ranges. Tasks with Sec_i exceeding a security threshold remain executed locally. The complete configuration is released with the open-source implementation for full reproducibility.

2) Implementation Details: All algorithms are implemented in Python 3.10 and executed on an Intel Xeon Silver 4314 server-class CPU with 128GB RAM. A unified execution protocol is adopted: Maximum iteration count: $K_{max} = 1000$, Initial population size: $n_{pop}(0) = 80$, lower bound $n_{min} = 20$, Dynamic penalty coefficient range: $p_{min} = 0.1$, $p_{max} = 20$, Early-stop check interval: 50 iterations. Each experiment is repeated for 10 runs, and the mean ± standard deviation is reported.

3) Evaluation Metrics: The following quantitative metrics are used:

- **Energy consumption:** $E_s = E_c + E_t$
- **Average task latency:** $L_{avg} = \frac{1}{N} \sum_i L_i$
- **Feasible task ratio:**

$$P_{feasible} = \frac{\#\{i \mid L_i < \vartheta_i\}}{N}$$

- **Convergence speed:** iterations required to enter the 5% deviation range of the best observed fitness

4) Baseline Comparison: To assess performance improvements, ACE-IVY is compared against four metaheuristic optimization approaches: the Original IVY, DE [29], APOA [30–32], and GA [33], using the Rastrigin global optimization benchmark. All experiments are repeated for 10 independent runs to ensure statistical reliability.

5) Convergence Performance: Fig. 2 illustrates the convergence behavior observed in the experiments. ACE-IVY shows a steeper initial decrease, consistent with its adaptive penalty and diversity preservation components, and demonstrates effective convergence across 10 independent runs.

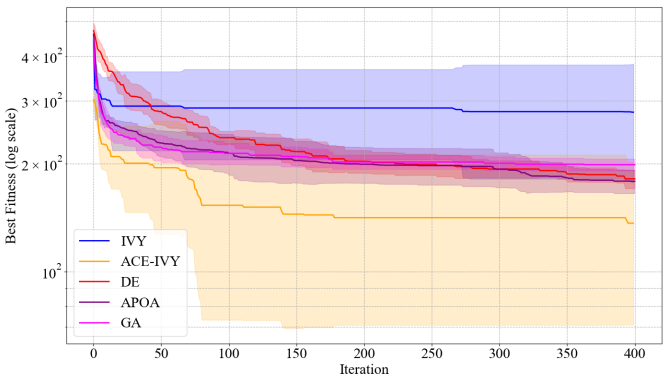

Fig. 2. Convergence comparison of different algorithms on the Rastrigin benchmark. The horizontal axis denotes the iteration number, and the vertical axis denotes the objective value. Results are averaged over 10 independent runs.

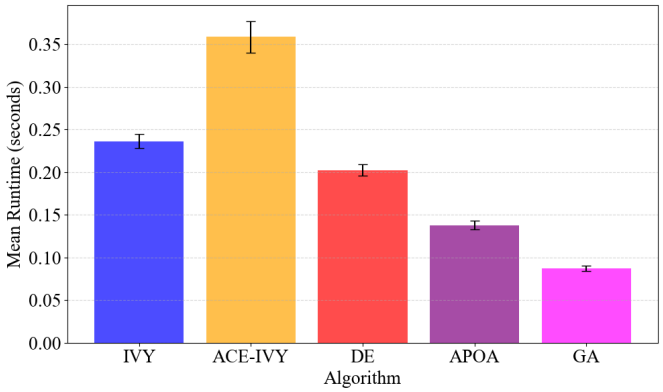

Fig. 3. Runtime comparison of different algorithms. Bars represent the mean runtime (seconds) over 10 independent runs, and error bars indicate the standard deviation.

6) Runtime Efficiency: Fig. 3 presents the runtime comparison across evaluated algorithms (results aggregated over 10 runs). Despite incorporating adaptive mechanisms, ACE-IVY maintains competitive runtime relative to the original IVY.

7) Energy Efficiency in Cloud-Edge Offloading: To examine performance under cloud-edge offloading, three task scales (0.5, 1.0, 2.0) and varying resource configurations with edge nodes $n \in [1, 20]$ and local devices $m \in [1, 50]$ are evaluated. Fig. 4 presents the resulting 3D energy distribution patterns.

It is worth noting that the experimental evaluation is conducted using synthetic workloads due to the lack of publicly available real-world cloud-edge task offloading traces for traffic signal control systems with detailed energy and security annotations. The synthetic tasks are generated following workload patterns inspired by traffic-oriented simulators such as CityFlow, which are commonly calibrated using real traffic data and widely adopted in intelligent transportation research. While this setup enables controlled and reproducible evaluation, validating the proposed framework on real-world deployments or simulator-based datasets calibrated with real traffic traces remains an important direction for future work.

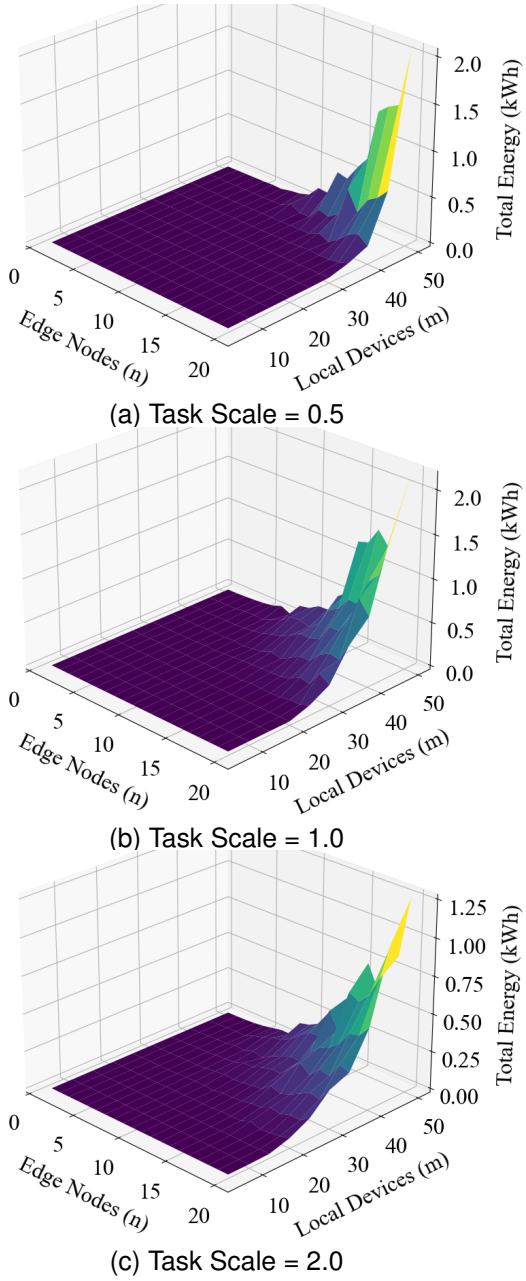

Fig. 4. 3D energy consumption of the cloud–edge system using ACE-IVY under different task scales. The horizontal axes denote the number of edge nodes and local devices, and the vertical axis denotes total energy consumption (kWh). Results are averaged over 10 independent runs.

V. Conclusions

The rapid evolution of IoT-enabled smart cities has intensified computational demands for real-time traffic control, where resource-constrained field controllers must process massive numbers of delay-sensitive tasks under strict energy and latency requirements. This work constructs a dual-mode offloading model and formulates a constrained optimization problem that minimizes total energy consumption while satisfying resource capacity, delay bounds, and security constraints. Building on this formulation, the adaptive constraint-enhanced

ivy algorithm integrates dynamic penalty coefficients, hybrid early stopping, adaptive population regulation, and pruning of inferior or infeasible solutions. Experimental results on large-scale synthetic traffic workloads demonstrate that the proposed algorithm consistently achieves lower energy consumption and competitive runtime compared with several representative metaheuristic baselines. These findings indicate that the approach is well suited for practical deployment in smart traffic signal systems and more general cloud–edge–device environments. In future work, we intend to couple the proposed optimization framework with deep reinforcement learning for adaptive edge selection, and to extend the modeling and algorithmic ideas to other cyber–physical infrastructures such as smart manufacturing and intelligent energy management.

REFERENCES

[1] R. Zadeh, A. Elmi *et al.*, "A conceptual high level multiagent system for wildfire management," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 63, pp. 1–15, Apr. 2025.

[2] R. Min, Y. Chen *et al.*, "Das vehicle signal extraction using machine learning in urban traffic monitoring," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–10, Feb. 2024.

[3] H. Huang, W. Li *et al.*, "A long-term vehicle tracking algorithm of correlation filter optimized by swarm intelligence tracking framework," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 62, pp. 1–16, Aug. 2024.

[4] X. Wang, M. Zhou *et al.*, "A branch and price algorithm for crane assignment and scheduling in slab yard," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 3, pp. 1122–1133, Jul. 2021.

[5] H. Zhao, Z. Wang *et al.*, "Online workload scheduling for social welfare maximization in the computing continuum," *IEEE Transactions on Services Computing*, 2025, early access.

[6] V. Rekha, S. Prasad *et al.*, "Hybrid edge-cloud approach for renewable energy management using deep learning with predictive analytics," in *Proc. 2024 Int. Conf. Recent Advances in Science and Engineering Technology (ICRASET)*, B G Nagara, Mandya, India, Feb. 2025, pp. 1–7.

[7] J. Bi, Z. Wang *et al.*, "Cost-minimized computation offloading and user association in hybrid cloud and edge computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16 672–16 683, May 2024.

[8] J. Bi, Z. Wang *et al.*, "Cost-minimized partial computation offloading in cloud-assisted mobile edge computing systems," in *Proc. 2023 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, Honolulu, HI, USA, Jan. 2024, pp. 5052–5057.

[9] J. Bi, Z. Wang *et al.*, "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.

[10] Z. Liu, Y. Chen *et al.*, "Topology-aware dynamic computation offloading in vehicular networks," in *Proc. 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, Helsinki, Finland, Jun. 2021, pp. 1–5.

[11] K. Kalyani and S. Thangaraj, "Reducing the communication overhead in novel data sharing algorithm in cloud data storage over triple data encryption standard algorithm," in *Proc. 2024 Second Int. Conf. Advances in Information Technology (ICAIT)*, Chikkamagaluru, Karnataka, India, Oct. 2024, pp. 1–4.

[12] J. Zhou, D. Tian *et al.*, "Reliability-oriented optimization of computation offloading for cooperative vehicle-infrastructure systems," *IEEE Signal Processing Letters*, vol. 26, no. 1, pp. 104–108, Jan. 2019.

[13] H. Gu, L. Zhao *et al.*, "Ai-enhanced cloud-edge-terminal collaborative network: Survey, applications, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 26, no. 2, pp. 1322–1385, Dec. 2023.

[14] S. Qin, S. Zhang *et al.*, "Multiobjective multiverse optimizer for multi-robotic u-shaped disassembly line balancing problems," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 2, pp. 882–894, Feb. 2024.

[15] J. Wang, M. Zhou *et al.*, "Multiperiod asset allocation considering dynamic loss aversion behavior of investors," *IEEE Transactions on Computational Social Systems*, vol. 6, no. 1, pp. 73–81, Feb. 2019.

[16] K. Zhang, R. Zhou *et al.*, "Transmission line component defect detection based on uav patrol images: A self-supervised hc-vit method," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 11, pp. 6510–6521, Nov. 2024.

[17] G. Mojtaba, Z. Mohsen *et al.*, "Optimization based on the smart behavior of plants with its engineering applications: Ivy algorithm," *Knowledge-Based Systems*, vol. 295, Jun. 2024.

[18] Z. Zhang, X. Guo *et al.*, "Multi-objective discrete grey wolf optimizer for solving stochastic multi-objective disassembly sequencing and line balancing problem," in *Proc. 2020 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC)*, Toronto, ON, Canada, Oct. 2020, pp. 682–687.

[19] S. Zhang, N. Yi *et al.*, "A survey of computation offloading with task types," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 8, pp. 8313–8333, Aug. 2024.

[20] Y. Chen, Y. Sun *et al.*, "Joint task and computing resource allocation in distributed edge computing systems via multi-agent deep reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 4, pp. 3479–3494, Jul.–Aug. 2024.

[21] H. Li, R. Duan *et al.*, "System-wide energy-efficient computation offloading in vehicular edge computing with speed adjustment," *IEEE Transactions on Green Communications and Networking*, vol. 8, no. 2, pp. 701–715, 2024.

[22] T. Chi, W. Zhang *et al.*, "Atom: Adaptive task offloading with two-stage hybrid matching in mec-enabled industrial iot," *IEEE Transactions on Mobile Computing*, vol. 23, no. 5, pp. 4861–4877, May 2024.

[23] J. Liang, K. Deb *et al.*, "A survey on evolutionary constrained multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 2, pp. 201–221, Apr. 2023.

[24] F. Ming, J. Zhang *et al.*, "Constrained multiobjective optimization via multitasking and knowledge transfer," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 77–89, Feb. 2024.

[25] X. Li, M. Li *et al.*, "A competitive and cooperative evolutionary framework for ensemble of constraint handling techniques," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2023, early access.

[26] F. Ming, J. Zhang *et al.*, "A constraint-handling technique for decomposition-based constrained many-objective evolutionary algorithms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 12, pp. 7783–7793, Dec. 2023.

[27] X. Li, S. Chen *et al.*, "Decoupling constraint: Task clone-based multitasking optimization for constrained multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 29, no. 2, pp. 404–417, Apr. 2025.

[28] A. Correia, J. Matias *et al.*, "Resolution of constrained non-linear optimization problems using direct search methods combined with new measures to admissibility in filters method," in *Proc. 2017 Int. Conf. Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)*, Prague, Czech Republic, May 2017, pp. 242–247.

[29] Y. Hou, D. Qiao *et al.*, "Knowledge graph-based dynamic differential evolution algorithm for e-waste recycling vehicle routing problem," in *Proc. 2025 IEEE 6th Int. Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, Shenzhen, China, Apr. 2025, pp. 1–6.

[30] T. Grinshpoun and A. Meisels, "Compapo: A complete version of the apo algorithm," in *Proc. 2007 IEEE/WIC/ACM Int. Conf. Intelligent Agent Technology (IAT'07)*, Fremont, CA, USA, Nov. 2007, pp. 370–376.

[31] J. Zhang, "Performance optimization in communication systems using deep reinforcement learning with elite reverse learning strategy – arctic puffin optimization," in *Proc. 2025 3rd Int. Conf. Integrated Circuits and Communication Systems (ICICACS)*, Raichur, India, Apr. 2025, pp. 1–5.

[32] M. Zhong, H. Zhang *et al.*, "Apo-based parallel algorithm of channel allocation for cognitive networks," *China Communications*, vol. 13, no. 6, pp. 100–109, Jun. 2016.

[33] K. Deb, A. Pratap *et al.*, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

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.

Jiayi Li is currently pursuing a master's degree at the College of Computer Science, Beijing University of Technology, China. She previously earned a bachelor's degree in Computer Science and Technology from Southwest Petroleum University in 2024. Her research interests span time series analysis, cloud computing, intelligent optimization algorithms, and deep learning.

Haitao Yuan (S'15–M'17–SM'21) received the Ph.D. degree in Computer Engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA in 2020. He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China. His research interests include cloud computing, edge computing, data centers, big data, machine learning, deep learning and optimization algorithms. He received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, and the Best Paper Award in the 17th ICNSC.

Jing Bi (M'13–SM'16) received her B.S., and Ph.D. degrees in Computer Science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively. She is currently a Professor with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology, Beijing, China. She has over 150 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud computing, large-scale data analytics, machine learning and performance optimization. She is now an Associate Editor of IEEE Transactions on Systems Man and Cybernetics: Systems and IEEE Transactions on Industrial Informatics. She is a senior member of the IEEE.