Energy-Minimized Task Offloading in Smart Traffic Signal Edge-Cloud Systems

Zeyuan Gao
College of Computer Science
Beijing University of Technology
Beijing, China
2477117465@qq.com

Haolin Ma
College of Computer Science
Beijing University of Technology
Beijing, China
2632553006@qq.com

Xiangxi Wu
College of Computer Science
Beijing University of Technology
Beijing, China
xiangxi.wu@emalis.bjut.edu.cn

Cheng Bao
School of Computer Science and Technology
East China Normal University
Shanghai, China
2686786811@qq.com

Ziqi Wang
School of Software Technology
Zhejiang University
Zhejiang, China
wangziqi0312@163.com

Jing Bi
College of Computer Science
Beijing University of Technology
Beijing, China
bijing@bjut.edu.cn

2025 China Automation Congress (CAC) | 979-8-3315-8967-7/25/$31.00 ©2025 IEEE | DOI: 10.1109/CAC67268.2025.11487136

***Abstract*—In the context of cloud-edge-device collaboration for intelligent traffic signal control, the three-tier architecture comprising Traffic Signal Units (TSUs), Roadside Units (RSUs), and Cloud Data Centers (CDCs) faces challenges in task allocation for energy optimization and real-time constraints. This work constructs a dual-mode model integrating unconstrained offloading and reliability-constrained offloading, establishing an optimization problem that minimizes total energy consumption while satisfying resource capacity and security compliance. The model fuses computational and transmission latency components with dynamic computation and transmission energy consumption models. The IVY algorithm, a meta-heuristic optimization method inspired by the growth patterns of ivy plants, efficiently navigates complex search spaces to identify optimal solutions. Building on this foundation, the Adaptive Constraint Enhanced IVY Algorithm (ACE-IVY) introduces dynamic penalty coefficients, early stopping mechanisms, adaptive population regulation, and constraint pruning. Experiments in large scale tasks intelligent traffic scenario show that ACE-IVY achieves faster convergence and higher efficiency compared to traditional algorithms, effectively solving task allocation under security constraints and verifying its practicality in real-time traffic control resource scheduling.**

***Index Terms*—Cloud-edge-device collaboration, intelligent traffic control, dual-mode offloading model, and energy optimization.**

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].

This work was supported by the Beijing Natural Science Foundation under Grants L233005 and 4232049, National Natural Science Foundation of China under Grants 62473014 and 62173013. (Corresponding author: Ziqi Wang.)

With explosive growth in sensing devices and evolving computing paradigms, data processing complexity in smart city applications is increasing exponentially. 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 [4] and causing substantial energy wastage [5]. Computation offloading technology [6]–[8] 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 [9]. However, offloading involves wide-area data transmission, introducing non-negligible communication overhead in delay and bandwidth [10]. 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 [11].

Motivated by this, we design a computation offloading strategy within a cloud-edge-terminal architecture to minimize energy consumption while guaranteeing delay requirements and security constraints [12]. We first construct an intelligent traffic-light-controlled architecture with multiple edge nodes and terminals. Next, we formulate an energy minimization problem incorporating device processing capabilities and transmission overheads. Finally, we propose an enhanced plant-inspired algorithm called Adaptive Constraint Enhanced IVY Algorithm (ACE-IVY), featuring key innovations: Levy flight-inspired dynamic penalty optimization, adaptive population size scaling for varying task volumes, hybrid early stopping mechanism for computational economy, and an alpha-beta pruning-like strategy for efficiency. Experimental results demonstrate ACE-IVY's efficacy in enhancing system performance within our architecture.

Sections are organized as follows: Section II details the architecture and problem formulation. Section III presents the ACE-IVY algorithm implementation and simulation results, and Section IV summarizes findings and future directions.

II. Problem Presentation

To optimize the cost-minimization problem, Fig. 1 illustrates an intelligent traffic control system with cloud-edge-device collaboration. This architecture connects TSUs to RSUs, where each RSU may serve multiple TSUs. The 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 Reliability-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 Reliability-constrained 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.

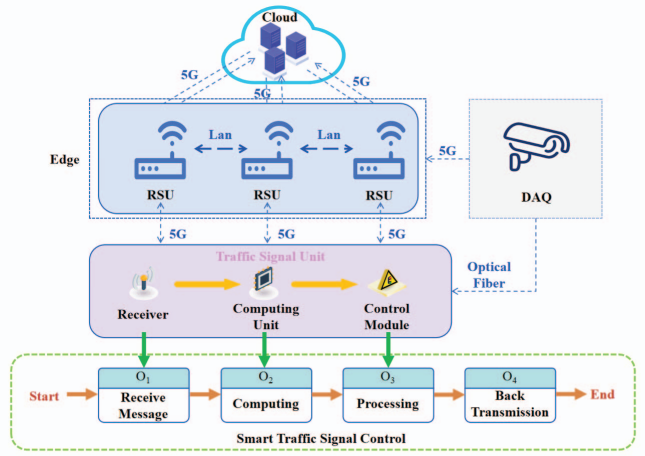


Fig. 1. Architecture of the smart traffic signal system

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.

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

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

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:

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

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

where $S\hat{N}R_{ij}$ represents the uplink channel signal-to-noise ratio, and $S\check{N}R_{ij}$ 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 $\mathcal{C}'_i$ represent the execution time of tasks offloaded to the CDC, and υ denote the computational speed of the CDC's CPU. Then:

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

Therefore, The total time $\mathcal{C}_i$ of processing associated with CDC can be expressed as: $\mathcal{C}_i=\mathcal{C}'_i+\mathcal{C}''_i$, where $\mathcal{C}''_i$ 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.*,

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

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: $E_{\mathrm{s}}{=}E_{\mathrm{c}} + E_{\mathrm{t}}$, 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: $E_{\mathrm{c}}{=}C_{\mathrm{e}}{\cdot}V_{\mathrm{d}}^2{\cdot}f{\cdot}N{\cdot}t$, 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:

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

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 $r_{\mathrm{n},i}$ is the allocation ratio for node and task i.

3) Transmission Energy Model: The total transmission energy is given by: $E_{\mathrm{t}}{=}E_{\mathrm{tx}}{+}E_{\mathrm{rx}}$, 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: $E_{\mathrm{tx}}{=}\left(P_0{+}k{\cdot}d^{\eta}{\cdot}B\right){\cdot}B/R$, 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: $E_{\mathrm{rx}}{=}(P_{\mathrm{a}}{+}P_{\mathrm{p}}){\cdot}B/R$, 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

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

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

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, $\epsilon_{\mathrm{TSU/RSU/CDC}}$ denotes the computational energy per unit task for each node type, $\tau_{\mathrm{RSU/CDC}}$ 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:

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

where C_{RSU} and C_{CDC} denote the maximum computational loads for RSUs and CDCs, respectively. For tasks with high security requirements [13], if Sec_i exceeds the threshold $\mathfrak{s}$, then $\beta_{ij}{=}0$ and $\gamma_i{=}0$.

III. 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:

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

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

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: :

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

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

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

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.

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 $k = K_{max}$.

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:

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

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.

$$n_{\text{inj}}{=}\max\left(1,\lfloor n_{\text{pop}}{\times}0.1\rfloor\right) \tag{29}$$

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

C. Experimental Evaluation

To validate the improvement of the ACE-IVY it was compared with four baselines, including the Original IVY, DE [14], APOA [15]–[17] and GA [18], on the Rastrigin function, a benchmark for global optimization. All experiments run 10 times to ensure statistical significance.

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 ← f_p, f_best ← min f_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_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
```

1) Convergence Performance: Fig. 2 plots the convergence curves, in which mean ± is the standard deviation of competing algorithms. The ACE-IVY exhibits a steeper initial decline because of its dynamic penalty and diversity injection mechanisms, enabling stronger global exploration. It consistently outperforms baselines and converges to a lower fitness value, validating the proposed adaptive strategies.

2) Runtime Efficiency: Fig. 3 compares the mean runtime across algorithms. Despite integrating adaptive mechanisms, the ACE-IVY maintains competitive efficiency with the original IVY. This balance between optimization accuracy and speed is critical for real-time edge computing.

3) Energy Efficiency in Cloud-Edge Offloading: To assess the improved IVY algorithm's performance in cloud-edge task allocation, we analyze system energy consumption across three task scales 0.5, 1.0, 2.0, and varying resource configurations, edge nodes $n \in [1, 20]$ and local devices $m \in [1, 50]$.

Fig. 4 illustrates the 3D energy distribution. The x-axis and y-axis represent the number of edge nodes and local devices, respectively, while the z-axis and color gradient quantify energy consumption: warmer colors = higher energy. In addition, for the edge node scaling, energy decreases with more edge nodes, but gains saturate beyond $n = 15$. Larger tasks, *e.g.* scale 2.0, increase baseline energy, but the algorithm maintains smooth, low-energy regions via dynamic offloading. Moreover, the algorithm avoids energy spikes for robustness, demonstrating effective constraint handling under scaled workloads.

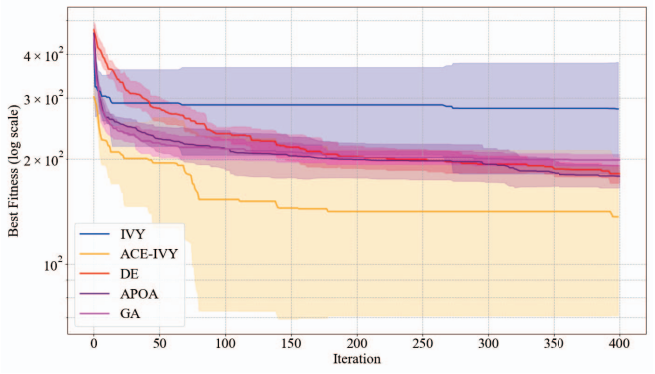


Fig. 2. Algorithm Convergence Comparison on Rastrigin (10 Runs)

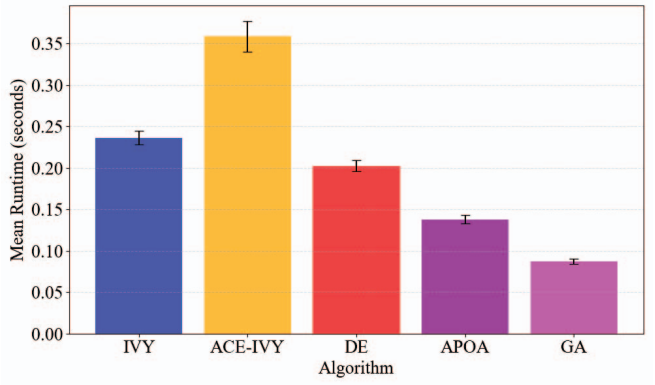
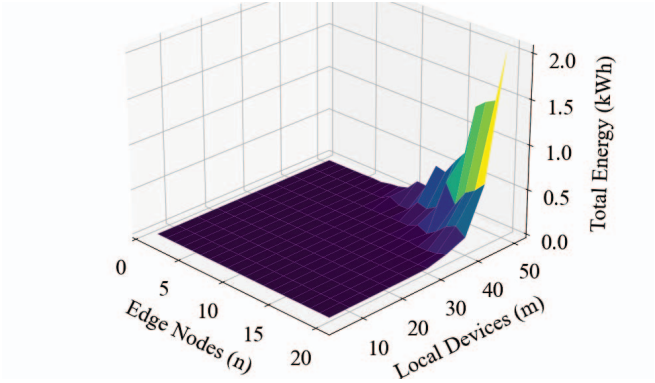


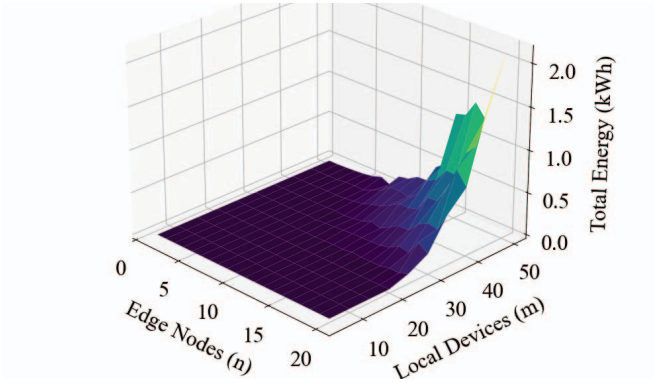
Fig. 3. Algorithm Runtime Comparison (Mean ± Std)

IV. Conclusion

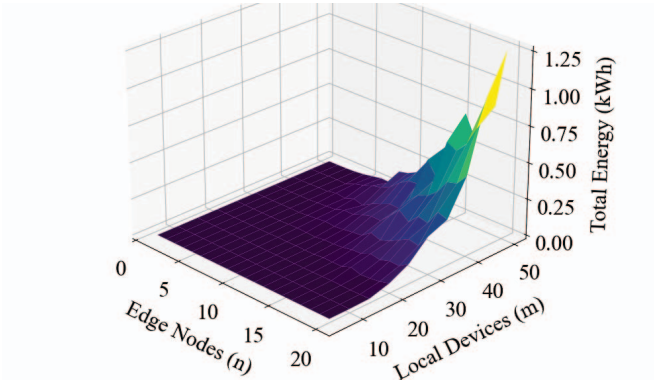
The rapid evolution of IoT-enabled smart cities has intensified computational demands for real-time traffic control, where resource-constrained TSUs struggle with energy limitations and latency requirements. To address this, our work constructs a dual-mode offloading model and formulates a constrained optimization problem minimizing total energy consumption $E_s = E_c + E_t$ while satisfying resource capacity and security compliance. Our solution, the ACE-IVY algorithm, integrates four key innovations: (1) dynamic penalty coefficients, (2) hybrid early stopping, (3) adaptive population regulation, (4) constraint pruning. Experimental validation with the model above demonstrates that ACE-IVY attains the best solution with lower cost. For future work, we intends to incorporate



(a) Task Scale = 0.5



(b) Task Scale = 1.0



(c) Task Scale = 2.0

Fig. 4. 3D Energy Consumption of Cloud-Edge Systems Using ACE-IVY

deep reinforcement learning for adaptive edge selection, and extend the usage of ACE-IVY and the modeling method above in related fields [17].

REFERENCES

[1] R. B. Zadeh, A. Elmi, V. Moghaddam and S. MahmoudZadeh, "A Conceptual High Level Multiagent System for Wildfire Management," in IEEE Transactions on Geoscience and Remote Sensing, vol. 63, pp. 1-15, Apr. 2025.

[2] R. Min, Y. Chen, H. Wang and Y. Chen, "DAS Vehicle Signal Extraction Using Machine Learning in Urban Traffic Monitoring," in IEEE Transactions on Geoscience and Remote Sensing, vol. 62, pp. 1-10, Feb. 2024.

[3] H. Huang, W. Li, M. Niu, M. Sipon Miah, T. Gao and H. Wang, "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] H. Zhao Z. Wang, G. Cheng, W. Qian, P. Chen, J. Yin, S. Dustdar, and S. Deng, "Online Workload Scheduling for Social Welfare Maximization in the Computing Continuum," *IEEE Transactions on Services Computing*, doi: 10.1109/TSC.2025.3570845.

[5] V. Rekha et al., "Hybrid Edge-Cloud Approach for Renewable Energy Management Using Deep Learning with Predictive Analytics," 2024 International Conference on Recent Advances in Science and Engineering Technology (ICRASET), B G Nagara,Mandya, India, 2024, pp. 1-7, Feb. 2025.

[6] 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.

[7] J. Bi, Z. Wang, H. Yuan and J. Zhang, "Cost-Minimized Partial Computation Offloading in Cloud-Assisted Mobile Edge Computing Systems," *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Honolulu, Oahu, HI, USA, 2023, pp. 5052–5057, Jan. 2024.

[8] J. Bi, Z. Wang, H. Yuan, J. Zhang and M. Zhou, "Self-adaptive Teaching-learningbased Optimizer with Improved RBF and Sparse Autoencoder for Highdimensional Problems," *Information Sciences*, vol. 630, pp. 463–481, Jun. 2023.

[9] Z. Liu et al., "Topology-Aware Dynamic Computation Offloading in Vehicular Networks," 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 2021, pp. 1-5, Jun. 2021.

[10] K. Kalyani and S. J. J. Thangaraj, "Reducing the Communication Overhead in Novel Data Sharing Algorithm in Cloud Data Storage Over Triple Data Encryption Standard Algorithm," 2024 Second International Conference on Advances in Information Technology (ICAIT), Chikkamagaluru, Karnataka, India, 2024, pp. 1-4, Oct. 2024.

[11] J. Zhou, D. Tian, Y. Wang, Z. Sheng, X. Duan and V. C. M. Leung, "Reliability-Oriented Optimization of Computation Offloading for Cooperative Vehicle-Infrastructure Systems," in IEEE Signal Processing Letters, vol. 26, no. 1, pp. 104-108, Jan. 2019.

[12] H. Gu, L. Zhao, Z. Han, G. Zheng and S. Song, "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.

[13] A. Correia, J. Matias, P. Mestre and C. Serôdio, "Resolution of Constrained Non-linear Optimization Problems Using Direct Search Methods Combined with New Measures to Admissibility in Filters Method," 2017 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO), Prague, Czech Republic, 2017, pp. 242-247, May. 2017.

[14] Y. Hou, D. Qiao, H. Han and J. Huang, "Knowledge Graph-Based Dynamic Differential Evolution Algorithm for E-Waste Recycling Vehicle Routing Problem," 2025 IEEE 6th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT), Shenzhen, China, 2025, pp. 1-6, Apr. 2025.

[15] T. Grinshpoun and A. Meisels, "CompAPO: A Complete Version of the APO Algorithm," 2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07), Fremont, CA, USA, 2007, pp. 370-376, Nov. 2007.

[16] J. Zhang, "Performance Optimization in Communication Systems Using Deep Reinforcement Learning with Elite Reverse Learning Strategy - Arctic Puffin Optimization," 2025 3rd International Conference on Integrated Circuits and Communication Systems (ICICACS), Raichur, India, 2025, pp. 01-05, Apr. 2025.

[17] M. Zhong, H. Zhang and B. Ma, "APO-based parallel algorithm of channel allocation for cognitive networks," in China Communications, vol. 13, no. 6, pp. 100-109, Jun. 2016.

[18] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, Apr. 2002.

[19] Y. Cao, "Design of Computer Wireless Network Teaching System based on Collaborative Filtering Optimization Algorithm," 2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT), Bengaluru, India, 2024, pp. 1-5, May. 2024.