ARTICLE | Article in Press

Reliability-oriented edge computation offloading strategy for Internet of Vehicles: based on improved hybrid fox optimization algorithm

Yubao Liu · Quanchao Sun · Bocheng Yan · Fengru Li · Chenhao Li

Received: 24 March 2026 / Accepted: 11 September 2026

Abstract
In the Internet of Vehicles (IoV), edge computing supports computation-intensive tasks through roadside unit servers. However, guaranteeing reliable task execution remains challenging in dynamic environments with limited resources. We propose a reliability-oriented computation offloading strategy based on the Improved Hybrid Fox Optimization (IHFOX) algorithm. The algorithm integrates sine chaotic mapping and Lévy flight to balance exploration and exploitation. It further employs a standard normal distribution for position updates, thereby accelerating convergence and avoiding local optima. For model construction, we establish a single-replica reliability model from dual dimensions of transmission reliability and computational reliability, meeting system-level reliability constraints through a multi-replica parallel redundancy mechanism. Meanwhile, vehicle mobility, resource/energy constraints, and joint weight factors are integrated to form a comprehensive optimization model under reliability constraints. Experiments demonstrate that the IHFOX algorithm guarantees high reliability with over 95% task completion rate, while significantly reducing system cost, delay, and energy consumption, exhibiting superior practical value in reliability-sensitive IoV applications.

Keywords Internet of Vehicles · Computation offloading · Edge computing · Reliability · Heuristic algorithm · Resource optimization

Introduction

In recent years, driven by the concurrent advances in artificial intelligence and wireless communication technologies, traditional transportation systems have gradually evolved into intelligent transportation systems[1]. As the number of connected autonomous vehicles grows rapidly and the Internet of Vehicles (IoV) matures, a myriad of computationally intensive and delay-sensitive applications have emerged at vehicle terminals. These applications, such as augmented reality (AR) navigation, vehicle intrusion detection, and autonomous driving systems[2,3], all rely on powerful computing power and low-delay communication to achieve real-time data processing, rapid decision-making, and precise information interaction[4]. However, it is difficult to meet the needs of these services only by relying on the computing and storage resources of the vehicle itself[5]. Therefore, mobile Edge Computing

Published online: 17 September 2026

(MEC), a technology with substantial potential, has attracted widespread attention for enhancing the computing power at the vehicle edge layer[6]. Specifically, vehicles offload computationally intensive tasks from their applications and network operations to servers deployed at the edge of the vehicular network for execution[7,8], enabling faster and more intelligent services and functions. This paradigm is referred to as Vehicle Edge Computing (VEC)[9].

Computation offloading has been extensively investigated under single-user and multi-user IoV frameworks, yet existing works concentrate on delay and energy optimization while neglecting reliability guarantees. For instance, Mao et al.[10] leveraged Lyapunov optimization with energy harvesting to prolong device battery life, yet their approach becomes ineffective when harvesting opportunities are scarce. Similarly, Liu et al.[11] employed Markov Decision Processes (MDP) to minimize delay, but the inherent complexity of MDP severely limits its scalability in multi-vehicle networks. As IoV applications evolve toward realistic multi-user and multi-task deployments, researchers have incorporated dynamic pricing[12], highest-response-ratio-next scheduling[13], and distributed offloading frameworks[14]. Some studies even expanded the resource pool to include parked vehicles and pedestrians[15,16], while others explored dependency-aware task scheduling[17] or cost-driven resource allocation[18]. However, a critical oversight across these works is the absence of a systematic reliability guarantee—they assume perfect wireless channels and fault-free servers, or at best consider bit-error-rate in isolation, without jointly accounting for computation failures, server crashes, and mobility-induced interruptions. This omission is particularly detrimental in safety-critical IoV applications, where task success is paramount. Since these model-based optimization frameworks fail to jointly model mobility dynamics, resource constraints and multi-source task failures simultaneously, numerous scholars resort to metaheuristic algorithms to obtain feasible suboptimal solutions for NP-hard offloading problems.

Meta-heuristic algorithms are widely adopted for NP-hard offloading problems, yet conventional variants fail to simultaneously address reliability constraints and high vehicular mobility. The Genetic Algorithm (GA)[19] and its improved versions[20] have been applied to edge offloading, achieving performance gains but often at the cost of slow convergence. Particle Swarm Optimization (PSO)[21] combined with max-min fairness has been used to optimize execution times for clustered vehicles, yet it suffers from premature stagnation in rapidly changing environments. Other variants, such as the Quantum Ant Colony algorithm[22], Ant Colony Optimization for job caching[23], linear programming combined with binary PSO for dynamic task allocation[24], and the Bat Algorithm for vehicular edge services[25], have also been explored. Nevertheless, these traditional metaheuristics share two fundamental weaknesses when deployed in dynamic IoV scenarios. First, they exhibit slow convergence and poor adaptability under resource-saturated or high-mobility conditions, as their exploration-exploitation balance is static and cannot respond to rapidly changing channel states and vehicle topologies. Second, and more importantly, reliability constraints are entirely absent from their optimization frameworks—they minimize delay or energy as objectives, but never enforce a hard reliability threshold, meaning the lowest-cost solution may be practically infeasible when transmission or computation failures occur. This limitation motivates the recent introduction of learning-based paradigms for dynamic offloading scenarios.

Deep reinforcement learning (DRL) has become a mainstream paradigm for dynamic IoV offloading, yet its inherent drawbacks limit application in reliability-critical scenarios. DDPG[26] is designed for continuous action spaces and enables joint optimization of offloading decisions and resource allocation in highly dynamic IoV scenarios. SAC[27] introduces a maximum entropy framework to improve global exploration while ensuring convergence, making it suitable for complex offloading decisions involving multiple vehicles and edge nodes. TD3[28] further improves upon DDPG by mitigating overestimation bias and enhancing stability in volatile vehicular networks. These DRL-based methods achieve excellent performance in optimizing delay and energy consumption and represent the state-of-the-art in current computation offloading research. However, despite their popularity, DRL-based approaches face inherent limitations that prevent them from directly addressing our reliability-oriented problem. First, they rely on sparse and delayed reward signals, making it extremely difficult to train stable policies when reliability violations (e.g., transmission timeouts or server failures) occur infrequently. Second,

the learned policies are brittle to environmental shifts—sudden changes in vehicle density or channel quality can cause severe performance degradation without retraining. Third, and most critically, DRL optimizes reliability only through soft penalty terms in the reward function, which cannot provide deterministic guarantees.

By synthesizing the limitations of the three research branches above, we systematically identify the unaddressed core gaps and clarify the positioning of our IHFOX scheme. Existing works fail to simultaneously (i) model transmission reliability and computational reliability within a unified mathematical framework; (ii) guarantee system-wide task success via multi-replica parallel redundancy with explicit hard reliability constraints; (iii) holistically integrate vehicle mobility, computing/energy resource limits and their coupling relationships; (iv) adopt an optimizer with fast convergence and strong local optimum avoidance for high-dimensional discrete offloading decisions. To fill this gap, we propose a reliability-oriented offloading strategy based on the Improved Hybrid Fox Optimization (IHFOX) algorithm. Our model jointly considers transmission and computational reliability, introduces a replica-count decision rule to meet a predefined threshold, and integrates mobility and resource constraints into a comprehensive cost function. The IHFOX algorithm is enhanced with sine chaotic initialization, Lévy flight exploration, and normal-distribution-based updates. It is specifically tailored to overcome the limitations of traditional heuristics in highly dynamic and reliability-critical IoV scenarios. Specifically, we identify that existing studies fall short in three critical dimensions—reliability modeling (oversimplified or absent), dynamic adaptability (static search cannot track high-speed vehicle mobility), and algorithmic performance (slow convergence or frequent constraint violations). A concise dimension-wise comparison against existing studies is summarized in Table 1, highlighting the novelty and necessity of our approach.

With the above issues in mind, the contribution of this paper can be summarized as follows:

- Model: We construct a multi-vehicle multi-service computing task offloading model based on a vehicle-road-cloud collaborative environment. This model is suitable for highway driving scenarios. The model integrates the infrastructure such as roadside units, vehicles and MEC servers to form a complete vehicle edge computing network communication platform. We construct a single-replica reliability model from two dimensions: transmission reliability and computational reliability. System-level reliability constraints are met through a multi-replica parallel mechanism. In addition, the model takes vehicle mobility into account and uses joint weighting factors to balance delay and energy costs.
- Algorithm: We propose an Improved Hybrid Fox Optimization (IHFOX) algorithm. Inspired by fox hunting behavior, the algorithm integrates three synergistic mechanisms. First, sine chaotic initialization ensures uniform coverage of the high-dimensional discrete decision space and avoids initial population clustering. Second, Lévy flight exploration enables large-jump search for escaping local optima and tracking time-varying feasible regions caused by vehicle mobility. Third, an adaptive position update strategy based on the standard normal distribution stabilizes fine-grained search near strict resource constraint boundaries. These improvements respectively address three core challenges faced by the traditional FOX algorithm in IoV scenarios:

Table 1 Comparison of existing studies and our work.

Dimension	Existing studies	Our work
Reliability guarantee	Ignore reliability requirements, or only consider a single dimension (transmission or computation)	Joint modeling of transmission + computation dual dimensions, multi-replica parallel redundancy to meet system-level reliability threshold
Dynamic environment adaptation	Exploration-exploitation imbalance, difficulty in tracking time-varying feasible regions caused by high-speed vehicle mobility	Lévy flight enables large-jump search, rapidly adapting to vehicle mobility and channel variations
Convergence performance	Traditional heuristics (PSO/GA/FOX) converge slowly in dynamic environments	Sine chaotic initialization +normal distribution update, accelerating convergence and avoiding local optima
Constraint coupling handling	Delay, energy, and reliability optimized independently without considering resource constraint coupling	Joint weight factors balance multiple objectives, hard constraints (mobility/resources/reliability) coupled and satisfied

first, insufficient coverage of high-dimensional decision space and population clustering caused by random initialization; second, difficulty in tracking dynamic feasible regions triggered by high-speed vehicle mobility; and third, unstable boundary search and frequent invalid iterations under strict resource constraints. The three mechanisms operate synergistically: chaotic initialization ensures diverse starting points for global exploration, Lévy flight enables rapid relocation capability in dynamic environments, and standard normal distribution update guarantees fine-grained stable search at constraint boundaries. Collectively, these enhancements improve the global search ability, convergence speed, and dynamic adaptability of the traditional FOX algorithm, making it more suitable for edge computation offloading scenarios in the Internet of Vehicles.

- Validation: Compared to existing benchmark solutions, our approach significantly reduces the average system overhead, time delay, and energy consumption. Simulation results also demonstrate how the number of CPU cycles required for tasks, the amount of task data, and the number of vehicles affect system performance.

The structure of this paper is arranged as follows: The second part introduces models, covering task model, vehicle mobility model, communication model, local computation model and computation offloading model. The third part describes the computation offloading strategy based on the improved algorithm in detail. The fourth part shows the experimental results and analysis. Finally, The fifth part concludes the paper and discusses future research directions.

System model

Figure 1 illustrates the multi-vehicle, multi-service model for vehicular computation offloading. The road is divided into multiple segments, each covered by roadside units (RSUs) with co-located Mobile Edge Computing (MEC) servers. Vehicles communicate with RSUs via Vehicle-to-Infrastructure (V2I) links. All RSUs are evenly spaced along the road. They are interconnected through optical fiber links, which enable high-speed and stable data transmission. The high bandwidth of optical fiber meets the demand for large-scale data transmission in vehicular networks. Meanwhile, its low delay and error rate ensure real-time and reliable data delivery. The set of RSUs (each integrated with an MEC server) is defined as $M = \{1, 2, 3, \ldots, m\}$ and the coverage radius of each RSU is denoted as R.

Figure 2 illustrates the overall system architecture of the proposed reliability-oriented offloading framework.

Task model

Vehicles generate tasks according to a Poisson arrival process during travel. Each vehicle carries exactly one task. The set of tasks is denoted as $N = \{N_1, N_2, \ldots, N_n\}$, and the set of vehicles as $V = \{1,2,\ldots,n\}$. At time t, the task generated by vehicle $j \in V$ is represented by a quadruple $T = (D_j, C_j, T_{max}, \chi)$ Table 2.

Vehicle mobility model

Vehicles travel at a speed v. To enhance the practicality of the model, a random fluctuation model is adopted for vehicle speed. Vehicles travel at a reference speed v_0, while the actual speed $v_j(t)$ fluctuates randomly within each time slot due to the impacts of road conditions, traffic density and driving behaviors:

$$v_j(t) = v_0 + \Delta v_j(t) \tag{1}$$

where $\Delta v_j(t)$ denotes the velocity disturbance term and follows a truncated normal distribution:

$$\Delta v_j(t) \sim \mathrm{TN}(\mu_v, \sigma_v^2, -v_{max}, v_{max}) \tag{2}$$

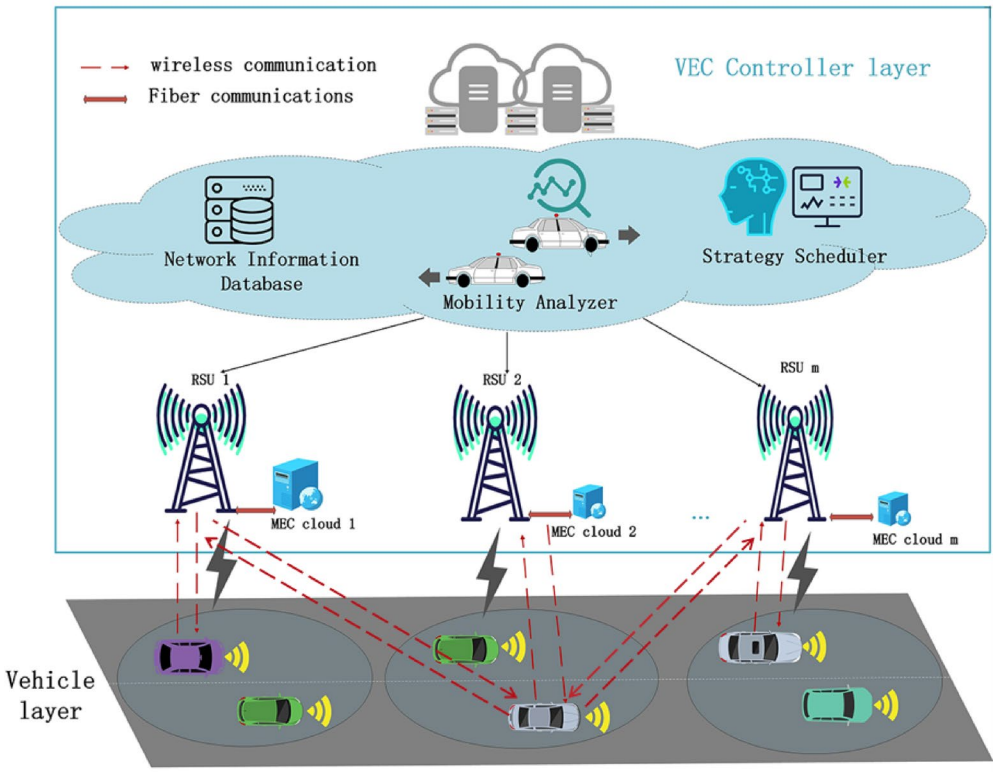


Fig. 1 Internet of Vehicles scenario.

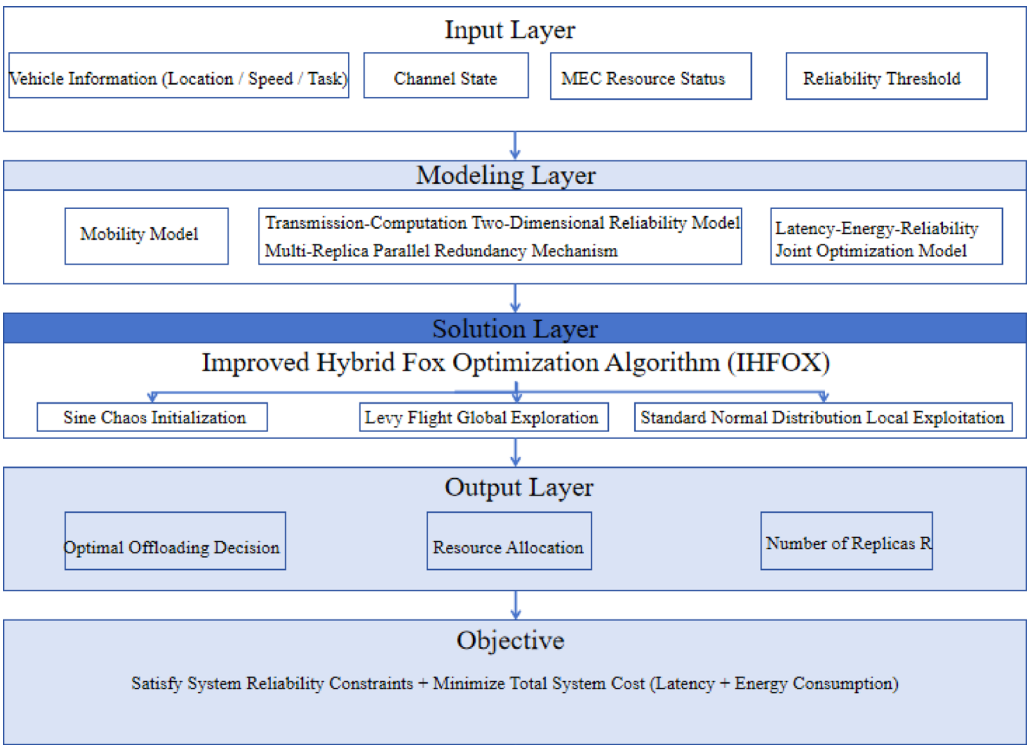


Fig. 2 Overall architecture of the proposed reliability-oriented edge computation offloading system.

Table 2 System model notation.

Symbol	Description
N	Total number of vehicles (tasks)
M	Set of RSUs (each with an MEC server)
R_m	Radius of communication coverage of RSUs
v_0	Reference speed of vehicles
σ_v	Standard deviation of speed fluctuation
$d_j(t)$	Horizontal distance between vehicle j and its associated RSU at time t
$E\left[T_{res}^{j}(t)\right]$	Expected residual dwell time of vehicle j in RSU coverage
D_j	Data size of the task j
C_j	CPU cycles required for the task j
T_{max}	Maximum tolerable delay
p_j	Transmit power per vehicle j
B	channel bandwidth
N_0	Gaussian channel white noise power
V_m	the set of vehicles simultaneously transmitted to RSU m
$H_{j,m}^2$	Channel gain between vehicle j and RSU m
r_j	Uplink transmission rate of vehicle j
f_{loc}	Calculation frequency per vehicle
F_i	computing resources of the MEC server i
S_m	The set of tasks offloaded to the RSU m
ζ	Power coefficient of CPU
$T(j)^{loc}$	Local computation delay for task j
$T(j)^{rsu}$	Offloading delay for task j
$E(j)^{loc}$	Local computing energy consumption for task j
$E(j)^{rsu}$	Offloading Calculated Energy Consumption for task j
ϵ_b	Bit Error Rate
R_t	Transmission reliability
R_c	Computational reliability
R_{single}	Single-replica reliability
R_{multi}	Multi-replica reliability
r_{th}	System reliability threshold
$R(a_i)$	Actual number of replicas deployed for vehicle i
η	Setting the delay factor
λ	Energy consumption factor
χ	Decision variable for task offloading
μ_v	Mean of speed disturbance
κ	Confidence coefficient
α	Path loss exponent
u	Channel fading constant
$T(j)^{R}$	Average delay under reliability constraints
$E(j)^{R}$	Average energy consumption under reliability constraints

Herein, $\mu_v = 0$ indicates no systematic velocity deviation, and the truncation interval $[-v_{max}, v_{max}]$ ensures that the vehicle velocity is non-negative and bounded (generally, $v_{max} = 0.3v_0$ is adopted).

To minimize transmission interruptions caused by handovers between RSUs, tasks initiated within the coverage of one RSU should ideally be completed before the vehicle leaves that RSU's coverage[29]. Assume vehicle j is currently within the coverage of an edge server. Because the velocity disturbance term has zero mean (i.e., $E[\Delta v_j(t)] = 0$), the expected velocity equals the reference velocity v_0. Consequently, the expected residual dwell time is determined by the reference speed rather than the instantaneous fluctuating speed, the expected residual dwell time of vehicle j within the coverage of RSU m at time t is defined as:

Algorithm 1 Fox optimization algorithm.

1: Initialize fox population $XP\ [i]\ (i\ =\ 1, 2, \ldots)$
2: **while** $it\ <\ It\_Max$ do
3: Initialize D_SouT , D_FoxP , T_SouT , $V\_Sou\ S$, H_Jump, T_Min , a, $BestX\ \ P$, $BestF$
4: Calculate the fitness value for each search variable using Eq.(30)
5: The population selects $BestXP$, $BestF$ for each iteration.
6: **if** $fit_i > fit_{i+1}$ then
7: $BestF = fit_{i+1}$
8: **end if**
9: **if** $r\ \geq\ 0.5$ then
10: **if** $p\ >\ 0.18$ then
11: Use Eq. (36) to find $XP\ (i\ +\ 1)$
12: **else** $p\ \leq\ 0.18$
13: Use Eq. (37) to find $XP\ (i\ +\ 1)$
14: **end if**
15: **else** $r\ <\ 0.5$
16: Calculate $T_Min$and a using Eq. (38) and (39).
17: Use Eq. (40) to find $XP(it + 1)$
18: **end if**
19: Updated $BestXP$
20: $it\ =\ it\ +\ 1$
21: **end while**
22: return $BestXP$, $BestF$

$$E\left[T_{res}^{j}(t)\right] = \frac{2\sqrt{R_m^2 - d_j(t)^2}}{v_0} \tag{3}$$

where $d_j(t) = |R_m - v_0 \cdot t|$ denotes the horizontal distance between vehicle j and RSU m at time t, assuming the vehicle enters the RSU coverage at $t = 0$.

Communication model

Tasks are offloaded from vehicles to edge servers at the corresponding RSUs. After computation, the results are transmitted back to the vehicles. In this process, vehicles and roadside units communicate wireless using Dedicated Short Range Communications (DSRC)[30], with the uplink and downlink channels modeled using frequency-flat fast fading Rayleigh channels[31]. In dense IoV scenarios, multiple vehicles within the same RSU's coverage may transmit simultaneously on the same frequency band. This results in co-channel inter-vehicle interference. The transmission rate is calculated by the Shannon formula based on the signal-to-interference-plus-noise ratio (SINR), which is expressed as:

$$r_j = B\log_2\left(1 + \frac{p_j H_{j,m}^2}{\sum_{k \in V_m, k \neq j} P_k H_{k,m}^2 + N_0}\right), \quad \forall\, j \in V \tag{4}$$

Where the interference term $\sum_{k \in V_m, k \neq j} P_k H_{k,m}^2$ represents the total inter-vehicle interference received by vehicle j. The channel gain is expressed as:

$$H_{j,m}^2 = u \cdot d_{j,m}^{-\alpha} \tag{5}$$

Where u is the channel fading model of the vehicle user denoted as $\mathrm{u} = 127 + 30\log\mathrm{d}$[32], $d_{j,m}$ is the distance between vehicle j and RSU m, and α denotes the path loss exponent.

Local computing model

Task j is processed directly on the vehicle, taking into account only the computation delay and the energy consumption that occurs locally. The local computation delay is:

$$T(j)^{loc} = \frac{C_j}{f(j)^{loc}}, \quad \forall\, j \in V \tag{6}$$

The energy consumption of the local onboard unit's CPU when executing task j is given by:

$$E(j)^{loc} = \zeta\left[f(j)^{loc}\right]^2 C_j, \quad \forall\, j \in V \tag{7}$$

$\zeta\left[f(j)^{loc}\right]^2$ denotes the power consumption of each vehicle core computing task[33].

Offloading model

When a vehicle offloads a computationally demanding and delay-sensitive task to an edge server at an RSU, the total offloading delay comprises three components: server computation delay, task transmission delay, and result return delay. In the simulations, the computation result size is much smaller than the uploaded data size—typically only 1% to 5%. Combined with DSRC's high-speed downlink, the return delay is less than 0.1 ms. This accounts for less than 0.5% of the upload and computation delays, which are both in the order of tens of milliseconds. Therefore, the return delay is negligible[34]. This assumption conforms to the common modeling and

experimental standards in the field of vehicular edge computing and will not affect the accuracy of experimental results or the fairness of comparisons. Thus, the task offloading delay is represented as:

$$T(j)^{rsu} = T(j)^{tran} + T(j)^{comp}, \quad \forall j \in V \tag{8}$$

The transmission delay is represented as:

$$T(j)^{tran} = \frac{D_j}{R_j} \tag{9}$$

The computation delay is expressed as:

$$T(j)^{comp} = \frac{C_j}{F_i} \tag{10}$$

The primary source of energy consumption when offloading tasks from vehicles stems from the energy expended in transferring data to the edge server. Thus, the task offloading energy consumption is represented as:

$$E(j)^{rsu} = \frac{p_j D_j}{r_j}, a \quad \forall j \in V \tag{11}$$

Reliability model

To quantify task completion reliability in IoV MEC offloading, we construct a single-replica reliability model from two dimensions: transmission reliability and computation reliability. System-level reliability constraints are met through a multi-replica parallel mechanism.

Single-replica reliability

In the reliability model of this paper, transmission failures, computation failures, and mobility interruptions are assumed to be mutually independent events. The rationale is as follows: transmission failures are caused by millisecond-level wireless channel fading and interference, while computation failures are determined by the hardware failure rate λ_m of edge servers. These two types of failures belong to the physical layer and the system layer, respectively, and are thus decoupled from each other. Mobility interruptions are characterized by the second-level vehicle dwell time (see Eq. (3)), whose time scale is much larger than that of the transmission and computation processes, and can therefore be approximated as a quasi-static process. Based on the above independence assumption, the single-replica reliability can be decomposed and solved in the product form via Eq. (12). The reliability of a single-replica task is jointly determined by the reliability of the wireless transmission phase, R_t, and the reliability of the MEC server computation phase, R_c. Specifically, the single-replica reliability, R_{single}, satisfies:

$$R_{single} = R_t \cdot R_c \tag{12}$$

If the task is chosen for local computation (without wireless transmission), thus, the transmission reliability R_t does not need to be considered, and the single-copy reliability is solely determined by the local computation reliability.

Assumptions and justification

The proposed reliability model adopts the product form $R_{single} = R_t \cdot R_c$ and the independent multi-replica mechanism $R_{multi} = 1 - (1 - R_{single})^R$. This choice follows standard practice in edge computing literature and is justified

for the following reasons: (i) transmission and computation failures originate from physically distinct layers (wireless channel vs. server hardware), making the independence approximation reasonable; (ii) replicas are dispatched to geographically separated RSUs with independent power and network backhauls, thus correlated failures are rare in urban IoV deployments; (iii) the exponential failure model for computational reliability is a first-order approximation that captures the dominant trend.

However, we also acknowledge that the above assumptions have certain limitations. Correlated failures may occur under extreme conditions, such as large-scale regional power outages that simultaneously affect the power supply of multiple RSUs, or sudden strong electromagnetic interference that disrupts all V2I wireless channels at the same time. Should such correlated failures actually take place, the actual system reliability R_{multi} would fall below our estimated values, because the independence assumption would overestimate the probability that at least one replica executes successfully. Furthermore, the constant failure rates (λ_m) in the exponential model may not fully hold in long-term operation scenarios, where factors such as hardware aging or thermal throttling could cause the failure rate to increase over time. Nevertheless, given the short execution time of IoV tasks (on the order of tens of milliseconds) and the typical operating conditions in urban IoV environments, the model provides sufficiently accurate and tractable approximations for optimizing offloading decisions.

Transmission reliability R_t

The reliability of the wireless transmission stage is determined by the Bit Error Rate (BER). In this paper, the QPSK modulation method is adopted. Firstly, the signal-to-interference-plus-noise ratio (SINR, dB) is converted to the linear signal-to-interference-plus-noise ratio, denoted as $SINR_{linear}$. Then, the Bit Error Rate ε_b is calculated through the complementary error function:

$$SINR_{linear} = 10^{SINR(dB)/10} \epsilon_b = \frac{1}{2} \cdot erfc\left(\sqrt{SINR_{linear}}\right) \tag{13}$$

Where $erfc()$ is the complementary error function, defined as:

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{+\infty} e^{-t^2} dt \tag{14}$$

The input data volume for the task is D_j(MB), which is converted to $8 D_j \times 10^6$ bits. Assuming each bit is transmitted independently, the probability of all bits being transmitted correctly (transmission reliability) is:

$$R_t = (1 - \epsilon_b)^{8D_j \times 10^6} \tag{15}$$

Computational reliability R_c

The calculation process of the MEC server incorporates an instantaneous failure rate and employs an exponential failure rate model to characterize computational reliability:

$$R_c = e^{-\lambda_m \cdot T(j)^{comp}} \tag{16}$$

Where λ_m represents the instantaneous failure rate of the MEC server, and $T(j)^{comp}$ denotes the computation delay of a single-replica task on the MEC server.A constant failure rate is adopted for two reasons: first, task computation time (tens of milliseconds) is far shorter than the mean time between failures (hours to days), so load-induced failure rate variation is negligible within the task execution window; second, this aligns with the standard constant-failure-rate assumption of the bathtub curve's flat region in reliability engineering. Load-dependent time-varying failure rates will be investigated in future extensions.

Consistent with the MEC server, the failure of the vehicle local CPU is also modeled by an exponential failure rate model:

$$R_c = e^{-\lambda_{loc} \cdot T(j)^{loc}} \tag{17}$$

Where λ_{loc} represents the instantaneous failure rate of the local CPU, and $T(j)^{loc}$ denotes the computation delay of a single-replica task on the vehicular local.

Reliability of parallel operation with multiple replicas

To meet the system reliability threshold r_{th}, a multi-replica parallel mechanism is introduced: if R replicas of a task are deployed, the task is deemed successful as long as at least one replica is executed successfully. In this work, the R replicas of the same task are distributed across different RSUs and their corresponding MEC servers. The primary replica is placed at the currently serving RSU, while the remaining replicas are deployed at neighboring RSUs. All replicas are transmitted over independent wireless channels. Since the RSUs are spaced 500 m apart and their hardware and backhaul links are mutually independent, the failure correlation among replicas is extremely low. Therefore, the independent failure assumption is adopted, and the multi-replica reliability is calculated via Eq. (18):

$$R_{multi} = 1 - (1 - R_{single})^R \tag{18}$$

The decision rule for the number of replicas R is: under the premise of satisfying $R_{multi} \geq r_{th}$, select the minimum number of replicas, and ensure that the number of replicas does not exceed the maximum number of replicas R_{max} (taken as $R_{max} = 3$ in this paper), that is:

$$R = \min\left\{ \arg\min_{R \in \{1,2,3\}} \left[1 - (1 - R_{single})^R \geq r_{th}\right], R_{\max} \right\} \tag{19}$$

Remark Conceptual distinction of replica-count decision rule. Multi-replica parallel redundancy is an existing technique. Existing redundancy-based offloading schemes for IoV often fix the number of replicas or consider only a single source of failure. In contrast, our rule determines the minimum number of replicas based on the joint reliability of transmission and computation, subject to the upper-bound constraint R_{max}; replica deployment is further constrained by vehicle dwell time; meanwhile, the additional delay and energy overhead introduced by replicas are explicitly incorporated into the optimization objective, thereby avoiding the indiscriminate consumption of system resources that would result from pursuing reliability alone.

Delay and energy consumption with reliability constraints

The multi-replica mechanism linearly amplifies the delay and energy consumption of a single replica. Therefore, the average delay with reliability constraints $T(j)^R$ and the average energy consumption with reliability constraints $E(j)^R$ are defined as follows:

Time delay with reliability constraints Let the offloading decision vector be $a = [a_1, a_2, ..., a_N]$ (where $a_i = 0$ indicates local computation by vehicle i, and $a_i \geq 1$ indicates offloading of vehicle i to the a_i-th RSU). Therefore, the time delay with reliability constraints is:

$$T(j)^R = \frac{1}{N} \cdot \sum_{i=1}^{N} \begin{cases} T(j)^{loc} \cdot R(a_i), a_i = 0 \\ max\, T(j)^{rsu}, a_i \geq 1 \end{cases} \tag{20}$$

Where $T(j)^{loc}$ represents the delay of a single copy computed locally; $maxT(j)^{rsu}$ denotes the maximum delay among all offloaded replicas (i.e., the completion delay of the last successfully executed replica); and $R(a_i)$ signifies the actual number of copies deployed by vehicle i.

Energy consumption with reliability constraints

$$E(j)^R = \frac{1}{N} \cdot \sum_{i=1}^{N} \begin{cases} E(j)^{loc} \cdot R(a_i), a_i = 0 \\ E(j)^{rsu} \cdot R(a_i), a_i \geq 1 \end{cases} \quad (21)$$

Where $E(j)^{loc}$ represents the energy consumption of a single replica for local computation; $E(j)^{rsu}$ denotes the energy consumption of a single replica offloaded to MEC; $R(a_i)$ signifies the actual number of replicas deployed for vehicle i.

Joint optimization problems

Decision variable

Let $x_j \in \{0,1\}$ denote the offloading decision for vehicle (task) j:

$\chi_j = 1$: task j is offloaded to the MEC server;

$\chi_j = 0$: task j is executed locally on the vehicle.

The number of replicas R_j is not an independent decision variable; it is determined by the single-replica reliability $R_{single}(j)$ via Eq. (19). For local execution ($\chi_j = 0$), transmission reliability $R_t = 1$, and $R_{single}(j)$ is computed solely from local computation reliability Eq. (17).

Objective function

For a given task j, the total delay and energy consumption under decision x_j are:

$$T(j) = (1 - \chi_j)T(j)^{loc} + \chi_j T(j)^{rsu} \quad (22)$$

$$E(j) = (1 - \chi_j)E(j)^{loc} + \chi_j E(j)^{rsu} \quad (23)$$

where $T(j)^{loc}$, $E(j)^{loc}$ are defined in (6)-(7), and $T(j)^{rsu}$, $E(j)^{rsu}$ are defined in (8)-(11).

The weighted system cost per task is:

$$W(j) = \eta T(j) + \lambda E(j) \quad (24)$$

Where η and λ are coefficients balancing delay and energy consumption, where $0 \leq \eta \leq 1, 0 \leq \lambda \leq 1$ and $\eta + \lambda = 1$. In practical applications, selecting suitable balancing factors needs to consider specific scenarios and requirements[35]. For example, in applications with higher real-time requirements, more emphasis may be placed on delay performance, thus larger values of η can be chosen; while in applications sensitive to energy consumption, more emphasis may be placed on energy consumption performance, thus larger values of λ can be chosen[36]. By continuously adjusting and optimizing the values of balancing factors, the optimal balancing solution suitable for specific application requirements can be found.

The overall optimization objective is:

$$Q = \min_{\{\chi_j\}_{j=1}^{N}} \sum_{j=1}^{N} W(j), \quad \forall j \in V \quad (25)$$

Constraints

$$
\begin{aligned}
&subject\ to \\
&C1: \quad \chi_j \in \{0,1\}, \quad \forall j \in V, \\
&C2: \quad T(j)^{tran} \leq \mathbb{E}[T_{res}^{j}(t)] \cdot \left(1 - \kappa \cdot \frac{\sigma_v}{v_0}\right), \quad \forall j \in V, \\
&C3: \quad f(j)^{loc} \geq 0, \quad \forall j \in V, \\
&C4: \quad \sum_{j \in s_m} C_j \cdot \chi_j \leq F_{\max}, \quad \forall i \in M, \\
&C5: \quad 1 - (1 - R_{single}(j))^{R_j} \geq r_{th}, \quad \forall j \in V.
\end{aligned} \tag{26}
$$

where $T(j)^{tran} = D_j / r_j$ from (9), $\mathbb{E}\left[T_{res}^{j}(t)\right]$ from (3), S_m is the set of vehicles offloading to RSU m, $R_{single}(j)$ from (12), and R_j is derived from (19).

Among these constraints, C1 specifies the vehicle's task offloading strategy, indicating that the vehicle can only choose one strategy: either offloading to the edge server or performing local computation. C2 ensures that task uploading is completed before vehicle j moves out of the coverage range of the edge server m it is connected to, where κ is the confidence coefficient, and σ_v/v_0denotes the velocity fluctuation coefficient. If a task cannot satisfy C2, it is forced to execute locally. C3 ensures that the local computing resources allocated to each vehicle cannot be negative. C4 indicates that the total computing resources allocated to each edge server must not surpass the server's maximum available capacity, highlighting the limited computation capability of edge servers. C5 ensures that the task completion reliability under the multi-replica mechanism satisfies the system-level reliability threshold, guaranteeing successful task execution through the multi-replica parallel redundancy mechanism in the dynamic IoV environment. The threshold $r_{th} = 0.98$ follows the typical reliability benchmark for safety-critical V2X applications, consistent with the requirements defined in 3GPP TR 22.886. Its sensitivity is implicitly demonstrated by : increasing the reliability threshold forces a higher number of replicas, which linearly amplifies delay and energy overhead. Our adaptive replica decision rule (Eq. (19)) always selects the minimum number of replicas to satisfy any given threshold, thus automatically optimizing the reliability-overhead trade-off. When $\chi_j = 0$ (local execution), the transmission reliability $R_t = 1$, and $R_{single}(j)$ is solely determined by the local computation reliability.

The joint optimization problem of computation offloading decision and resource allocation in this paper belongs to the NP-hard combinatorial optimization problem. Constrained by the discrete decision space, coupled multi-constraints, and highly dynamic vehicle environment, it is impossible to obtain the global optimal solution in polynomial time. Therefore, this paper adopts an improved heuristic algorithm for efficient solution, which is also a common treatment in this field.

Problem solving

Fox optimization algorithm

In 2023, Hardi et al.[37] introduced the FOX algorithm, inspired by the hunting tactics of the red fox. In snowy environments, the red fox stealthily approaches its prey. To locate prey, foxes employ a random search strategy in the search area. They detect prey by emitting ultrasonic waves. This random search strategy inspires the exploration phase of the FOX algorithm. Upon hearing prey sounds, the fox enters the exploitation phase. It estimates the distance to prey by measuring the sound travel time. Given that the speed of sound in air is approximately 343 m/s, the fox calculates the prey distance accordingly. During the attack, the prey may attempt to escape by leaping away. The fox then adjusts its jump timing based on the sound propagation time to ensure successful capture.The Earth's magnetic field influences the fox's jumping direction. The capture success rate reaches 82%

when jumping northeast, but drops to 18% in the opposite direction. Consequently, the fox employs two distinct capture strategies[38].

The overall flow of the FOX optimization algorithm is shown in Algorithm 1.

Improved hybrid fox optimization algorithm

In population-based meta-heuristic methods, the early optimization stage should emphasize extensive exploration of the entire solution space. This prevents premature convergence to local optima. In the later stage, the focus shifts to accelerating convergence toward high-quality solutions, thereby improving solution quality. However, the original FOX algorithm struggles to balance rapid convergence with avoiding local optima. Unlike PSO or GWO, FOX lacks built-in boundary-awareness and dynamic tracking components; its native uniform-random perturbation and fixed-probability jump logic perform poorly for the time-varying feasible regions of IoV offloading. While sine chaotic initialization, Lévy flight, and normal-distribution perturbation have been adopted in other meta-heuristics, our three enhancements are custom-tailored to remedy FOX-specific deficiencies and are tightly coupled with our discrete mapping and constraint-repair workflow for vehicular edge tasks, rather than isolated generic algorithmic tweaks. To overcome these limitations, we propose the Improved Hybrid Fox Optimization (IHFOX) algorithm, tailored for IoV scenarios which are characterized by high mobility, coupled constraints, and stringent reliability demands. The traditional FOX algorithm cannot simultaneously satisfy the requirements for fast convergence (to match vehicle dwell time), high accuracy (to respect resource constraints), and broad search coverage (to adapt to dynamic topologies). To this end, we propose a scenario-driven hybrid improvement strategy that combines the three techniques in a targeted manner with respective roles:

- Sine chaotic initialization[39]: solves the insufficient coverage of high-dimensional solution space caused by multiple vehicles and multiple RSUs in IoV, ensures the uniform distribution of initial offloading decisions, and avoids initial preference for local optima.
- Levy flight exploration[40]: copes with the sudden change of solution space caused by high-speed vehicle movement, realizes large-scale jump search, and quickly tracks available edge nodes and channel changes.
- Standard normal distribution update[41]: adapts to the strict constraints of edge resources. Its zero-mean and unit-variance properties concentrate perturbations near the current best solution for stable local optimization, while the unbounded tails allow occasional long jumps to escape local optima, with infeasible boundary violations corrected by the constraint repair mechanism.

This combination is customized for IoV scenarios rather than being a routine optimization of general metaheuristics. It effectively balances global exploration and local exploitation, making it suitable for dynamic, constrained, and high-reliability IoV offloading. Remark: Novelty positioning of IHFOX. The sine chaotic mapping, Lévy flight, and standard normal-distribution perturbation, as individual mechanisms, are already mature strategies in the metaheuristic literature, and we do not claim their individual novelty. The algorithmic contribution of IHFOX lies in two aspects. First, scenario-driven synergistic integration: targeting the triple coupled constraints of IoV offloading (high-dimensional discrete decisions, high-speed mobility, and strict resource boundaries), the three mechanisms form a closed loop—chaotic initialization ensures population diversity, providing an effective premise for Lévy flight; Lévy flight supplies large-span jumps to track the time-varying feasible region caused by vehicle mobility; and the normal-distribution fine search stabilizes convergence near resource constraint boundaries. This combinatorial logic is specifically tailored to the multiple constraints of IoV scenarios, and the performance gain arises from the interdependence among the three mechanisms rather than from the base mechanisms themselves. Second, cross-layer synergy with the reliability model: the multi-replica reliability constraint (C5) and the vehicle mobility constraint (C2) jointly define a time-varying feasible region, and the fast convergence of IHFOX is essential because the feasible region expires as vehicles leave the RSU coverage. This cross-layer synergy between reliability guarantees and heuristic search has not appeared in existing offloading literature.

Compared with traditional FOX, IHFOX significantly enhances search efficiency while maintaining robust global search capabilities.

In the IHFOX algorithm, each fox represents a computation offloading decision scheme, with its position in the search area defines the value of decision factor. An initial random value is first generated for each dimension. The fox population chaotic sequence is generated by applying the sine map for each fox (search agent) and each dimension, and the position of each fox is calculated and stored in the matrix XP_FOX using the generated chaotic sequence and the bounds of the problem (upper bound ub and lower bound lb). It ensures the diversity and universality of the initial search phase, and provides more possibilities for finding better solutions. The sine chaotic map is defined as:

$$\chi_{i+1} = s \cdot \sin(\pi \cdot \chi_i) \tag{27}$$

Here, χ_0 is the initial value and s is the scale factor, set to 0.9 to control the amplitude of the mapping. The sequence generated is used to map the search space, ensuring coverage of the entire feasible domain.

Discrete position mapping and constraint repair mechanism

Since computation offloading is a discrete decision-making problem, while the position update rules of IHFOX operate in continuous space, a deterministic mapping mechanism is required to convert continuous positions into discrete offloading decisions. The detailed procedures are as follows.

Step 1: Encoding scheme. Each fox position is an N-dimensional continuous vector, $x_i=[x_{i1},x_{i2},...,x_{iN}]$ where N is the number of vehicles. Each dimension $x_{ij} \in [0, M]$ corresponds to the offloading decision of vehicle j, and M is the number of RSUs.

Step 2: Discrete mapping. The continuous value x_{ij} is mapped to a discrete decision a_j via nearest-integer rounding:

$$a_j = \begin{cases} 0, \ if \ 0 \le x_{ij} < 0.5 \\ m, \ if \ m - 0.5 \le x_{ij} < m + 0.5, \ m = 1,2,...,M \end{cases} \tag{28}$$

where $a_j = 0$ indicates local execution, and $a_j = m$ indicates offloading to RSU m.

Step 3: Boundary handling. If $x_{ij} < 0$, it is set to 0; if $x_{ij} > M$, it is set to M. The mapped value is then re-applied to Eq. (28).

Step 4: Feasibility repair. After discrete mapping, each candidate solution is verified against constraints C2–C5. Constraint violations are resolved sequentially: Type-I reliability violation (C5) first, then Type-II server-resource overload violation (C4). The two types of infeasibility are handled as follows.

Type I — C5 violation (insufficient reliability). If the multi-replica reliability $R_{multi} < r_{th}$ for any task j, the number of replicas R_j is incremented (up to $R_{max} = 3$).

until the reliability constraint is satisfied. If R_{max} is reached and the constraint remains unsatisfied, that single task is forced to local execution ($a_j = 0$), where its reliability depends solely on the local CPU (Eq. (17)). Note that this per-task local fallback applies only to the reliability constraint and does not affect other tasks in the candidate solution. Type II — C4 violation (server resource overload). If the total CPU demand assigned to RSU m exceeds its capacity F_{max}, i.e., $\sum_{j \in s_m} C_j \cdot \chi_j \le F_{\max}$, a greedy reassignment procedure is triggered to restore feasibility with minimal performance degradation:

1. Identify the overloaded RSU $m^* = argmax_m \left(\sum_{j \in s_m} C_j - F_{\max} \right)$.

2. For each task j currently assigned to m^*, compute the cost increment ΔW_j for each feasible alternative action:

Migrate to local execution $(a_j = 0)$: $\Delta W_j^{loc} = \Delta W_j^{loc} - \Delta W_j^{rsu}$.

Migrate to another RSU $m' \neq m^*$: $\Delta W_j^{m'} = \Delta W_j^{m'} - \Delta W_j^{rsu}$, subject to m' having remaining capacity.

3. Select the task-action pair (j, a) with the minimum cost increment ΔW_j, and apply the reassignment. If all candidate migrations increase the total cost, select the pair yielding the smallest cost increment to enforce feasibility.
4. Repeat steps 1–3 until all RSU capacity constraints C4 are satisfied, or until a maximum repair iteration limit (set to N) is reached.

Fallback mechanism. In the rare case that the greedy repair procedure fails to restore feasibility within the iteration limit (e.g., under extremely tight resource constraints where no valid reassignment exists), the entire candidate solution is discarded and re-initialized using sine chaotic mapping to generate a new feasible starting point. This fallback is distinct from simply setting all agent elements to zero to force full local execution; it re-initializes the whole candidate via sine-chaotic mapping and serves only as a safety net to prevent population collapse, and is rarely triggered in normal simulation runs.

Step 5: Resource reallocation. For all feasible solutions, the computing resources of each RSU are proportionally reallocated among its assigned tasks based on workload:

$$F_i = \frac{C_j}{\sum_{k \in S_m} C_k} \cdot F_{max}, \forall\, j \in S_m \tag{29}$$

where S_m denotes the set of tasks offloaded to RSU m. This ensures full utilization of server capacity while strictly satisfying C4.

The time complexity of the repair mechanism is $O(N \cdot M)$, which does not change the polynomial order of the overall algorithm complexity.

In each iteration, employ the standard benchmark function Eq. (30), to determine the fitness value (fit) of each fox, represented by each row in the *XP_FOX* matrix, and the fitness values of all foxes are compiled into a vector named the fitness function (*Fit*).

$$f(x) = \sum_{j=1}^{n} \{\eta T(j) + \lambda E(j)\}, \forall\, j \in V \tag{30}$$

Compare the fitness values of each fox in *XP_FOX* with those of others to determine the best fitness value ($BestF$) and position ($BestXP_{it}$). To select the best solution from the possibilities, the fitness value of the new solution ($fit1$) is repeatedly compared with that of the previous solution (fit), updating the best score and position. The solution with the highest fitness value will be selected as the best solution. Figure 3 depicts the hunting mechanism of the IHFOX algorithm.

Exploitation phase

Set a random variable (r) to allocate a 50% chance of exploration or exploitation. At this stage, let $r \in [0,1]$ be a random number determining the probability of the optimal value (prey). Should the random number r exceed 0.18, it becomes necessary to determine the new position for the fox. Therefore, D_SouT_{it} is calculated as follows:

$$D\_SouT_{it} = V\_SouS * T\_SouT_{it} \tag{31}$$

Where T_SouT_{it} is a random number. $T\_SouT_{it} \in [0,1]$ and $it \in [1, 500]$denotes the number of iterations.

The speed of sound, determined by the position of the best search agent within the group, is represented as:

$$V\_SouS = \frac{BextXP_{it}}{T\_SouT_{it}} \tag{32}$$

D_SouT_{it} Representing one round trip to D_FoxP_{it}, the distance from the sensor to the object is half the distance measured by the acoustic wave[42], so D_FoxP_{it} is calculated as follows:

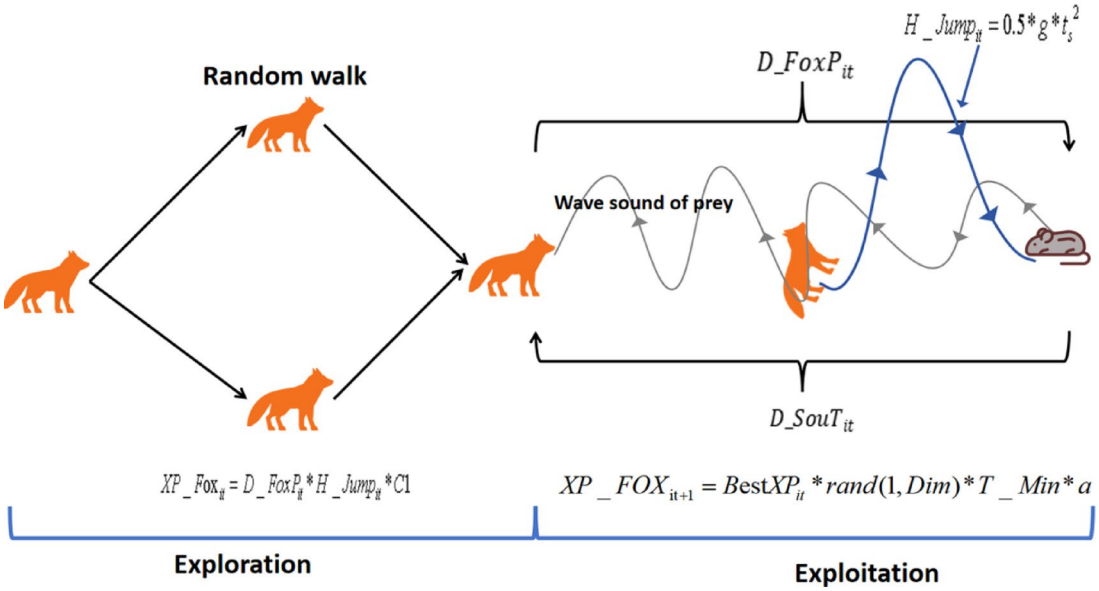


Fig. 3 The hunting mechanism of IHFOX algorithm.

$$D\_FoxP_{it} = D\_SouT_{it} * 0.5 \tag{33}$$

After determining the distance to the prey (i.e., the current best solution), the fox must update its position to leap and capture the prey. Thus, the fox needs to determine the height of its jump, calculated as shown below:

$$H\_Jump_{it} = 0.5 * g * t_s^2 \tag{34}$$

where g denotes the acceleration of gravity, which takes the value 9.81, and t_s denotes the average time of sound propagation. Since the jumping process has two phases of ascent and descent and takes equal time, the transition time ($Tran\_t$) is twice the sound propagation time. $Tran\_t$ is calculated as follows:

$$Tran\_t = \frac{\sum T\_SouT_{it}(i, :)}{Dim} \tag{35}$$

Then, H_Jump_{it} is multiplied by D_FoxP_{it} and the orientation variable K to obtain the new position of the fox, which is influenced by p. The new position is then scaled by a factor of $K1$ to determine the updated position of the fox. When $p \in [0.18,1]$, the fox jumps to the north-east to catch its prey and both H_Jump_{it} and D_FoxP_{it} are multiplied by $K1$ to increase the likelihood solution of reaching the new position and approaching the optimal position. Therefore, the new position is calculated as follows:

$$XP\_Fox_{it} = D\_FoxP_{it} * H\_Jump_{it} * K1 \tag{36}$$

When $p \in [0, 0.18]$ otherwise, the fox leaps in the reverse direction, multiplying both H_Jump_{it} and D_FoxP_{it} by $K2$, to get the new position as follows:

$$XP\_Fox_{it} = D\_FoxP_{it} * H\_Jump_{it} * K2 \tag{37}$$

Exploration phase

In this phase, the fox randomly modifies its search radius to locate prey, guided by the best position identified thus far. Given the fox's limited experience in jumping at this stage, to enhance its ability to roam to a more advantageous position, the search is adjusted according to the distance between the current position and the prey and

other factors, combined with the minimum time variable T_Min and the control factor a. The variables T_Min and control factor a are calculated as follows:

$$T\_Min = \text{Min}(Tran\_t) \tag{38}$$

$$a = 2 * \left(it - \frac{1}{It\_Max}\right) \tag{39}$$

where It_Max is the maximum number of iterations. Determining the variables T_Min and the control factor a is critical during the fox's exploration phase for approaching high-quality solutions. The best solution found so far significantly influences the exploration phase. In the FOX algorithm, the random exploration of prey by foxes is enhanced using rand(1, Dim)[43].

$$XP\_FOX_{it+1} = BestXP_{it} * \text{rand}(1, Dim) * T\_Min * a \tag{40}$$

In the original FOX algorithm, the exploration phase employs uniform random perturbation rand(1, Dim) (Eq. 40), which may lead to overly dispersed or concentrated search steps, resulting in unstable efficiency near constraint boundaries. In IHFOX, this is replaced by a standard normal distribution-based perturbation (implemented as $\mathcal{N}(0,1)$ via randn(1, Dim) in Eq. (41)). The zero-mean characteristic ensures unbiased exploration without systematic drift, while the unit variance concentrates the majority of random steps near the current best solution, which is particularly beneficial for fine-grained local exploitation under tight resource constraints (C3–C4). Additionally, the unbounded support of the normal distribution permits occasional long jumps to escape local optima. To handle the rare boundary violations (e.g., exceeding the feasible decision range $[0, M]$), the proposed constraint repair mechanism (Step 3 in Sect. 3.2) safely clips and remaps these solutions without introducing search bias. When $q \leq 0.8$, the exploration strategy for the new position is as follows:

$$XP\_FOX_{it+1} = BestXP_{it} * \mathcal{N}(0,1) * T\_Min * a \tag{41}$$

When $0.8 < q < 1$, the update of the new position incorporates Levy flight, generating Levy flight steps to simulate the jumping behavior of the fox. This method facilitates long-distance exploration in random searches. It helps avoid local optima and effectively explores different areas of the solution space. The step length L can be calculated using the following formula:

$$L = \frac{\mu}{|\upsilon|^{1/\beta}} \tag{42}$$

Herein, μ and ν are random variables, and the standard deviation σ of μ is calculated using the following formula:

$$\sigma = \left(\frac{\Gamma(1+\beta) \cdot \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \cdot \beta \cdot 2^{(\beta-1)/2}}\right)^{1/\beta} \tag{43}$$

The exploration strategy for the new position is as follows:

$$XP\_FOX_{it+1} = BestXP_{it} * \text{Levy}(1, Dim) * T\_Min * a \tag{44}$$

The global algorithm flowchart is shown in Fig. 4.

The overall flow of the IHFOX algorithm is shown in Algorithm 2:

For ease of reference, Table 3 summarizes the internal symbols specific to the IHFOX algorithm introduced above.

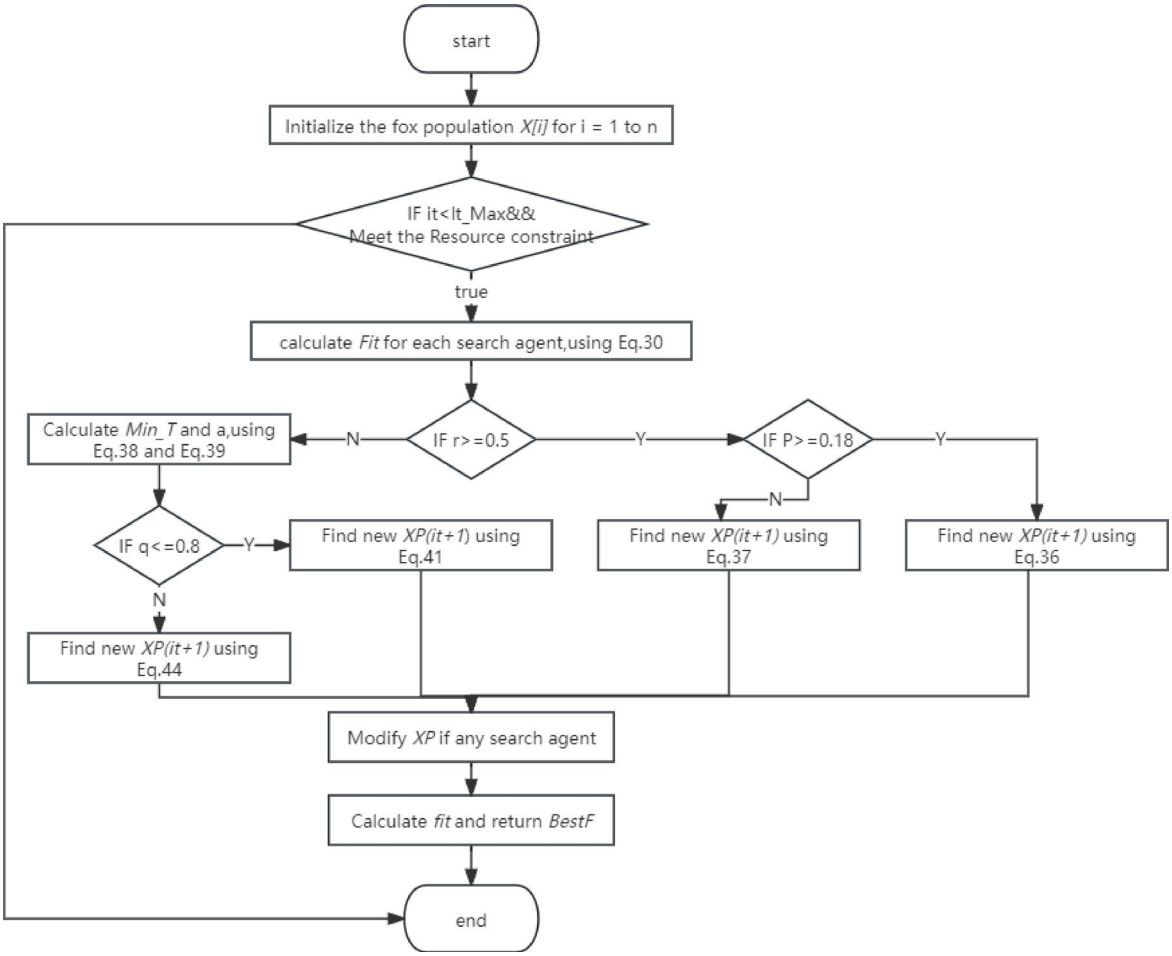


Fig. 4 Global algorithm flowchart.

Convergence and complexity analysis

Convergence and theoretical property analysis

Given the multi-vehicle task offloading problem is an NP-hard discrete combinatorial optimization problem with coupled hard constraints (reliability, resource, mobility), it is mathematically impossible to derive a finite-time upper bound to reach the strict global optimum for any heuristic algorithm. Instead, we decompose the theoretical analysis into three progressive layers with strictly limited conclusion strength, each supported by independent mathematical derivation and clearly stated inherent limitations.

Layer 1: ergodic coverage property of sine chaotic initialization *Core Statement.* The sine chaotic mapping used for population initialization can generate uniformly distributed initial search points over the entire discrete offloading decision space, effectively eliminating population clustering at local suboptimal regions in high-dimensional vehicle-RSU decision space.

Mathematical Support. The sine chaotic iteration formula:

$$\mathcal{X}_{i+1} = s \cdot \sin(\pi \cdot \mathcal{X}_i), s = 0.9 \tag{45}$$

Algorithm 2 IHFOX algorithm.

1:Initialize fox population $XP\ [i]\ (i\ =\ 1, 2, \ldots)$

2: **while** $it\ <\ It\_Max$ do

3: Initialize D_SouT, D_FoxP, T_SouT, V_SouS, H_Jump, T_Min, a, $BestXP$, $BestF$, L , σ

4: Map continuous positions to discrete decisions via Eq. (28)

5: Repair infeasible solutions (C2/C4 violations)

6: Reallocate RSU computing resources via Eq. (29)

7: Calculate the fitness value for each search variable using Eq.(30)

8: The population selects $BestXP$, $BestF$ for each iteration.

9: **if** $fit_i > fit_{i+1}$ then

10: $BestF = fit_{i+1}$

11: **end if**

12: **if** $r\ \geq\ 0.5$ then

13: **if** $p\ >\ 0.18$ then

14: Use Eq. (36) to find $XP\ (i\ +\ 1)$

15: **else** $p\ \leq\ 0.18$

16: Use Eq. (37) to find $XP\ (i\ +\ 1)$

17: **end if**

18: **else** $r\ <\ 0.5$

19: Calculate T_Min and a using Eq. (38) and (39).

20: **if** $q\ \leq\ 0.8$

21: Use Eq. (41) to find $XP\ (i\ +\ 1)$

22: **else** $0.8 <\ q\ < 1$

23: Calculate L and σ using Eq. (42) and (43).

24: Use Eq. (44) to find $XP\ (i\ +\ 1)$

25: **end if**

26: **end if**

Table 3 Internal symbols of IHFOX algorithm.

Symbol	Description
χ_i	Chaotic sequence value for sine mapping
s	Scale factor for sine chaotic map
$BestXP_{it}$	Best fox position (offloading decision) at iteration *it*
$BestF$	Best fitness value at current iteration
r	Random number in [0,1] for exploration/exploitation
p	Random number controlling jump direction
q	Threshold for Lévy flight activation
D_SouT_{it}	The distance the sound travels
D_FoxP_{it}	The distance between the fox and its prey
T_SouT_{it}	The time taken for the sound to travel
V_SouS_{it}	The speed of sound in air
H_Jump_{it}	Fox needs to determine the height of its jump
$K1, K2$	Orientation variables
T_Min	Minimum transition time
a	Control factor for exploration
L	Lévy flight step length
t_s	Average time of sound propagation
Dim	The problem dimension.

The sine map is proven ergodic over interval $[0,1]$in existing chaos theory literature. When mapping continuous chaotic values to discrete offloading decisions via Eq. (28), the initial fox agents are evenly sampled across all feasible $\{0,1,\ldots,M\}^N$ decision combinations.

Limited Conclusion Strength. This property only guarantees initial population diversity, and cannot ensure the algorithm can jump out of local optima in subsequent iterations. It is a necessary but insufficient condition for good convergence performance.

Experimental Verification. Corroborated by (initial population duplicate ratio ablation): IHFOX with chaotic initialization maintains below 4% duplicate solutions under 20–80 vehicles, while random initialization duplicates exceed 25%, directly verifying this theoretical property.

Layer 2: non-permanent trapping property of Lévy Flight exploration *Core Statement.* With Lévy flight heavy-tailed step update, for any local optimum attraction basin in the constrained offloading solution space, there always exists a strictly positive probability that the algorithm escapes this basin within a finite number of iterations. The algorithm cannot be permanently trapped in any single suboptimal solution.

Mathematical Support. Define the Hamming distance $H(X_{it}, X_{loc})$ to measure the separation between current solution and local optimum in discrete decision space. The Lévy flight step length L obeys heavy-tailed distribution $P(|L| > r) \propto r^{-\beta}$ $(0 < \beta \leq 2)$, which generates long-distance jumps with non-zero probability. For any finite-radius local attraction basin with Hamming threshold $r_0 > 0$, there exists $\rho > 0$ such that the single-step escape probability satisfies:

$$P(H(X_{it}, X_{loc}) > r_0) \geq \rho \tag{46}$$

Recursive deduction: The probability of being trapped for T consecutive iterations satisfies:

$$P(trapped\ for\ T) \leq (1-\rho)^T \underrightarrow{T \to \infty} 0 \tag{47}$$

Limited Conclusion Strength. This layer only proves no infinite stagnation at local optima, but two critical limitations remain. First, it does not guarantee the long jump will land in a better feasible solution (the jump may fall into another worse local basin or infeasible region, requiring constraint repair). Second, it cannot derive an explicit upper bound of iterations to escape any local optimum.

Experimental Verification. Corroborated by (sudden channel degradation experiment): The IHFOX equipped with Lévy flight recovers low-cost solutions within 10 iterations after channel deterioration, while the variant without Lévy flight remains trapped in high-fitness local states.

Layer 3: monotonic convergence of elite cost *Precondition.* Combining Layer 1 (diverse initial population) + Layer 2 (non-permanent local trapping) + elite retention mechanism (always record and retain the historical best solution $X_{best,it}$), we analyze the long-term asymptotic behavior of the algorithm's objective cost.

Core Conclusion. Let $Q(X^*)$ denote the theoretical minimal weighted system cost of the global optimal offloading scheme, and $Q(X_{best,it})$ denote the cost of the historical best solution at iteration it. Due to elitist preservation, the sequence $\{Q(X_{best,it})\}_{it=1}^{\infty}$ is monotonically non-increasing and bounded below by $Q(X^*)$. Therefore, by the monotone convergence theorem:

$$Q(X_{best,it}) \underline{it \to \infty} Q_{\infty} \geq Q(X^*) \tag{48}$$

where Q_{∞} is a random variable representing the asymptotic limit of the elite cost.

We explicitly do not claim $Q_{\infty} = Q(X^*)$. Establishing this equality would require proving that the global optimum is reachable with non-vanishing probability from any non-optimal state, which is intractable for NP-hard constrained problems with repair mechanisms that may discard improving jumps.

This monotonicity ensures that the algorithm does not diverge or oscillate, and that the final output is at least a stable fixed point of the search process. While this does not imply global optimality, it provides a rigorous guarantee of algorithmic stability and repeatability, which is the strongest form of convergence guarantee available for NP-hard combinatorial optimization under finite iterations.

What Can Be Established with Rigor. The combined effect of the three mechanisms guarantees three properties. First, the algorithm will not stagnate at a suboptimal cost indefinitely (from Layer 2). Second, the elite cost will converge to a stable value Q_{∞} (from monotonicity and boundedness). Third, the initial diversity (Layer 1) and escape capability (Layer 2) provide favorable conditions for approaching good solutions, but do not guarantee global optimality.

Inherent Limitations. Four critical limitations must be acknowledged. (i) No global optimum guarantee: Q_{∞} may be strictly greater than $Q(X^*)$. (ii) No finite-time convergence guarantee: this is only an asymptotic property under infinite iterations; for any fixed limited iteration budget (e.g., 100 iterations in simulation), we cannot mathematically prove the algorithm can reach the global optimum. (iii) No convergence rate closed-form bound: due to the NP-hard nature of the constrained discrete problem, we cannot derive a polynomial upper bound of iterations to narrow the optimality gap to an arbitrary small constant. (iv) Dependence on constraint repair: even if Lévy flight jumps to a new region, resource/mobility constraint repair may discard the new solution, slowing or preventing the descent of the elite cost.

Empirical Convergence Characterization. Although a closed-form polynomial convergence rate is theoretically unattainable due to the NP-hardness of the problem, we empirically characterize the algorithm's practical convergence behavior through the stabilization iteration count. As shown in, IHFOX reaches a stable cost plateau within approximately 30 iterations, which is less than the iteration budgets adopted in existing vehicular offloading literature (e.g., 100–200 iterations), demonstrating its practical efficiency for dynamic IoV scenarios. Furthermore, as reported in Tables 4 and 5, over 10 repeated simulations, IHFOX achieves the smallest mean cost (0.271) with the lowest coefficient of variation (2.95%), confirming stable near-optimal optimization performance under practical finite-iteration settings.

Summary of theoretical guarantees

The three-layer analysis collectively provides the following theoretically grounded guarantees, each with its corresponding limitation.

Table 4 Simulation data.

Category	Symbol	Value
Vehicle mobility	v_0	[20,40] m/s
	σ_v	0.1 v_0
	v_{max}	0.3 v_0
	κ	2
Communication channel	B	2 MHz
	N_0	−147dBm
	P_j	46dBm
	α	3.5
Task property	D_j	[0.5 ~ 2.0]MB
	C_j	[1000 ~ 2000]Megacycles
	T_{max}	100ms
Reliability constraint	r_{th}	0.98
	λ_m	10^{-5} 1/s
	λ_{loc}	10^{-4} 1/s
	R_{max}	3
Computing hardware	f_{loc}	[1.8 ~ 2.0] GHz
	F_i	400 GHz
	ζ	10^{-28} W·s³/cycle
	η	0.8
	λ	0.2
	R_m	500 m
Metaheuristic hyperparameters	P	30
	It_Max	100
	s	0.9
	q	0.8
	K_1, K_2	0.18,0.82
PSO exclusive	Inertia weight	0.9→0.4
	Cognitive/Social coefficients	2.0,2.0
GA exclusive	Crossover prob	0.5
	Mutation prob	0.005

Table 5 reports the mean, standard deviation (Std), 95% confidence interval (CI), and coefficient of variation (CV = Std/Mean × 100%) for the average total cost under a representative scenario (task data size = 1.5 MB, 20 vehicles).

Algorithm	Mean	Std	95% CI	CV (%)
IHFOX	0.271	0.008	[0.263, 0.279]	2.95
FOX	0.319	0.012	[0.307, 0.331]	3.76
PSO	0.399	0.018	[0.381, 0.417]	4.51
GA	0.659	0.031	[0.628, 0.690]	4.70
AOC	0.822	0.042	[0.780, 0.864]	5.11
LOC	2.460	0.087	[2.373, 2.547]	3.54

Layer 1 provides diverse initial coverage, which prevents premature bias toward any local subregion; however, this only affects initialization, not subsequent iterations.

Layer 2 provides non-stagnation, which ensures the algorithm cannot be permanently trapped in any single local optimum; however, this does not guarantee escape to a better solution.

Layer 3 provides monotonic elite convergence, which guarantees the historical best cost converges to a stable limit Q_∞; however, this does not guarantee $Q_\infty = Q(X^*)$.

These guarantees are sufficient for the practical effectiveness of IHFOX in dynamic IoV scenarios, while the explicit limitations acknowledged in each layer maintain mathematical rigor.

Computational complexity analysis

To evaluate the scalability of the IHFOX algorithm, the computational complexity is theoretically analyzed in this section. The problem scale parameters are defined as follows: the number of vehicles N, the number of RSUs M, the population size P, the maximum number of replicas $R_{max} = 3$, and the maximum iteration number It_Max. The problem dimension is $Dim = N$, where each vehicle corresponds to one offloading decision variable.

Fitness evaluation complexity For each candidate solution (fox), computing Eq. (29) requires: calculating transmission rate r_j via Eq. (4), which involves summing interference from vehicles in the same RSU taking $O(N)$ in the worst case; transmission reliability (Eqs. (13)–(15)) and computation reliability (Eqs. (16)–(17)) each cost $O(1)$; the multi-replica decision (Eqs. (18)–(19)) requires at most $R_{max} = 3$ attempts, hence $O(1)$; finally, summation over all N vehicles introduces an $O(N \cdot M)$ term, where the M factor comes from identifying the associated RSU for each vehicle. Consequently, a single fitness evaluation has complexity $O(N \cdot M)$.

Per-iteration complexity In each iteration, the algorithm performs fitness evaluation for all Pfoxes, costing $P \cdot O(N \cdot M) = O(P \cdot N \cdot M)$, and then updates positions using Eqs. (36)–(37) or (40)–(44), where each update is $O(N)$ per fox, totaling $O(P \cdot N)$. Therefore, the per-iteration complexity is $O(P \cdot N \cdot M)$.

Total complexity Over It_Max iterations, the overall time complexity of IHFOX is $T_{IHFOX} = O(It\_Max \cdot P \cdot N \cdot M)$. The space complexity is $O(P \cdot N)$ for storing the population matrix.

Comparison with baseline algorithms The original FOX and PSO share the same asymptotic complexity $O(It\_Max \cdot P \cdot N \cdot M)$. GA requires an additional sorting operation, resulting in $O(It\_Max \cdot PlogP \cdot N \cdot M)$, which is higher than IHFOX. AOC (average offloading) has complexity $O(N \cdot M)$ with no iterative search, but yields significantly inferior solution quality. LOC (local execution) is $O(N)$ but offers no offloading gain. Thus, IHFOX achieves superior optimization performance (see Sect. 4) without incurring extra asymptotic overhead compared to FOX and PSO, and its low-order polynomial complexity makes it suitable for real-time IoV offloading decisions. We note that the $O(N \cdot M)$ per-evaluation complexity is a worst-case bound derived from linear RSU scanning. In practical deployments with fixed RSU positions, vehicle-to-RSU association can be optimized to $O(NlogM)$ via spatial indexing structures such as KD-trees, further reducing runtime overhead without altering the algorithm's core search mechanism.

Performance evaluation

In this section, we validate the algorithms proposed for computation offloading and resource allocation strategies in vehicular network environments through numerical simulations implemented in a Python3.9 environment on a 2.70 GHz Intel Core i7 processor. We begin by introducing the settings for the simulation parameters. Then, to better validate the performance of the scheme proposed in this paper, the following five benchmark schemes were compared[44]:

1) LOC: All computation tasks are executed locally.
2)AOC: computation tasks are evenly distributed to edge servers for execution.
3) PSO: Particle Swarm Optimization algorithm is used to allocate computation offloading strategies.
4) GA: Genetic Algorithm is used to allocate computation offloading strategies.
5) FOX: The original Fox Search Algorithm is used to allocate computation offloading strategies.

All heuristic algorithms (PSO, GA, FOX, IHFOX) use the same population size (P = 30), maximum iterations (It_Max = 100), and the same random seed initialization for fair comparison.

Rationale for baseline selection.

The five benchmark schemes are chosen following a hierarchical logic to systematically validate the advantages of IHFOX.

LOC (local execution) and AOC (average offloading) serve as lower-bound and naive upper-bound references, respectively. LOC reflects the baseline performance without any offloading, while AOC represents a simple load-balancing heuristic that performs no intelligent search. Together, they establish the performance boundaries and highlight the necessity of optimization.

PSO and GA are selected as mainstream classical metaheuristics widely adopted in computation offloading research. They serve as standard state-of-practice benchmarks, and any newly designed heuristic must outperform them to verify its advancement over traditional swarm intelligence solvers.

FOX is the original algorithm upon which IHFOX is directly built. Including FOX enables an ablation-style analysis to isolate and quantify the specific contributions of the three proposed improvements (sine chaotic initialization, Lévy flight, and normal-distribution update), thereby verifying the incremental value of our modifications.

All four heuristic algorithms (PSO, GA, FOX, IHFOX) share the same training-free, online optimization paradigm and operate under identical population sizes, iteration counts, and random seeds. This ensures that performance differences stem solely from algorithmic search mechanisms, not from extrinsic factors. Deep reinforcement learning (DRL) methods such as DDPG, SAC, and TD3 are excluded from direct numerical comparison because they operate under a fundamentally different paradigm requiring extensive offline training and stable environment models. In the dynamic IoV scenarios addressed in this paper, DRL methods face inherent challenges including training instability due to sparse rewards from multi-replica redundancy, policy degradation under rapid topology changes, and inefficiency in high-dimensional discrete decision spaces. These methodological differences make fair numerical comparison infeasible without standardized training protocols. In the future, a hybrid framework of IHFOX and DRL can be explored: IHFOX provides reliable initial decisions, while DRL refines resource allocation during stable periods, combining the fast adaptability of metaheuristics with the long-term optimality of learning methods.

The simulation scenario described in this section is as follows: The highway simulation scenario adopts a unidirectional straight road with total length of 2000 m. Roadside Units (RSUs) integrated with MEC servers are uniformly deployed with an interval of 500 m, resulting in 4 RSUs covering the entire road segment. Each RSU has a wireless communication coverage radius $R_m = 500$. Vehicles are randomly generated following Poisson arrival distribution on the road entrance; each vehicle carries exactly one computation-intensive task. Vehicle mobility follows truncated normal speed fluctuation model defined in Eqs. (1)–(2). We enforce a hard handover constraint: all task transmission and computation must finish before vehicles exit the current RSU coverage (Constraint C2), otherwise tasks will be forced to local execution. DSRC communication (2 MHz bandwidth) is adopted for V2I uplink transmission, while fiber wired links connect all RSUs to the centralized VEC Controller with negligible transmission delay.Unless otherwise noted, parameters will follow the data in Table 4[45].

Next, we will comprehensively evaluate the performance of the system in terms of average total cost(i.e., the weighted system cost defined as $\eta \times \text{delay} + \lambda \times \text{energy}$), average energy consumption, and average delay from three dimensions: the central processing unit cycles required by the task (CPU), the scale of input data, and the number of tasks.

To ensure statistical robustness and account for the stochastic nature of heuristic algorithms, each experimental configuration was independently repeated 10 times with different random seeds (ranging from 1 to 10). This sample size follows the common practice in metaheuristic optimization literature, providing a balance between computational cost and statistical confidence. For clarity, error bars are omitted from the main figures, but the corresponding standard deviations, 95% confidence intervals, and coefficients of variation are summarized in Sect. 4.9.

Parameter sensitivity analysis

As illustrated in Figs. 5, 6 and 7, we analyze the sensitivity of core parameters to verify their rationality, so as to ensure the best performance of IHFOX in IoV edge computation offloading.

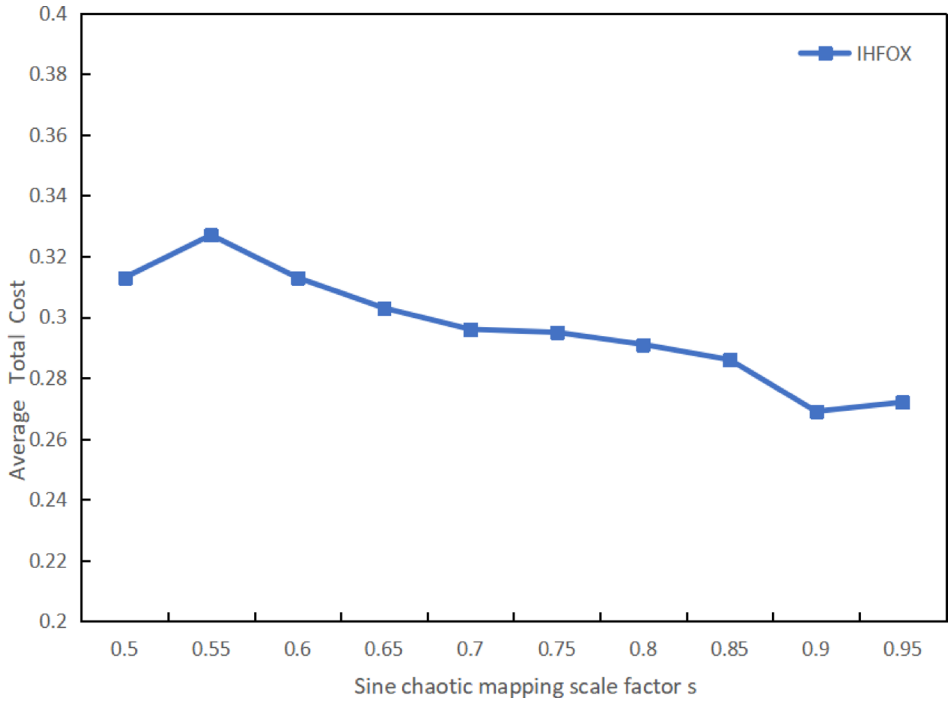


Fig. 5 Impact of scale factor s in sine chaotic mapping on average total cost.

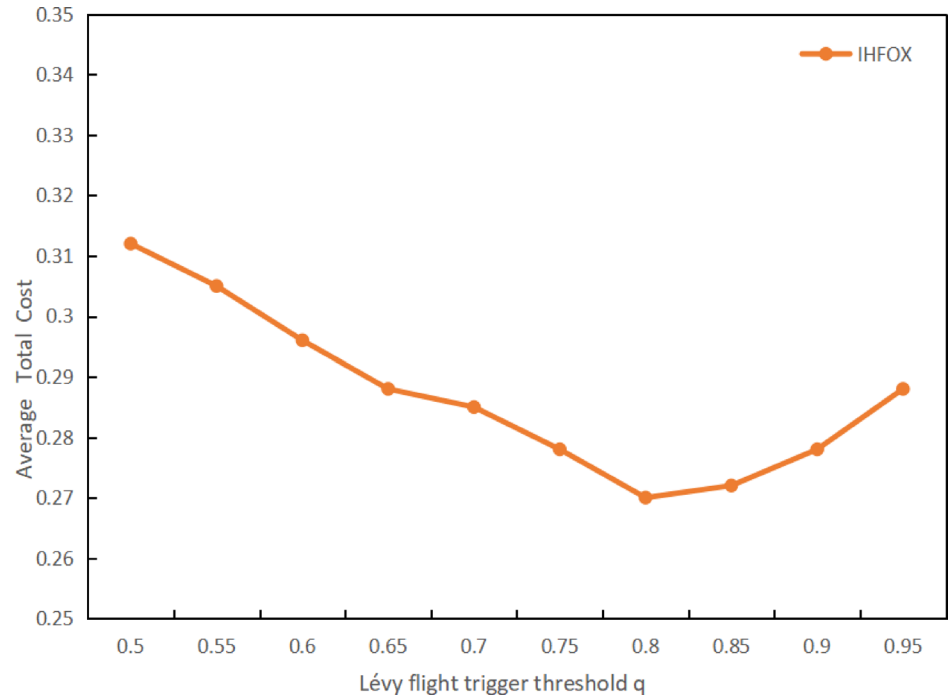


Fig. 6 Impact of Lévy flight trigger threshold q on average total cost.

Figure 5 illustrates the impact of the scale factor s in the sine chaotic mapping on the average total cost. The experiments are conducted with 20 vehicles, where each task requires 2000 Megacycles of CPU cycles and the input data size is 1.5 MB. As s increases from 0.5 to 0.9, the average total cost decreases from approximately 0.3 to 0.27, achieving the best performance among the tested values at $s = 0.9$. When s exceeds 0.9, the cost rises slightly to 0.275 at $s = 0.95$. This trend indicates that a proper scale factor can effectively control the amplitude of chaotic mapping, enhance the diversity of initial population distribution, and improve the exploration ability of

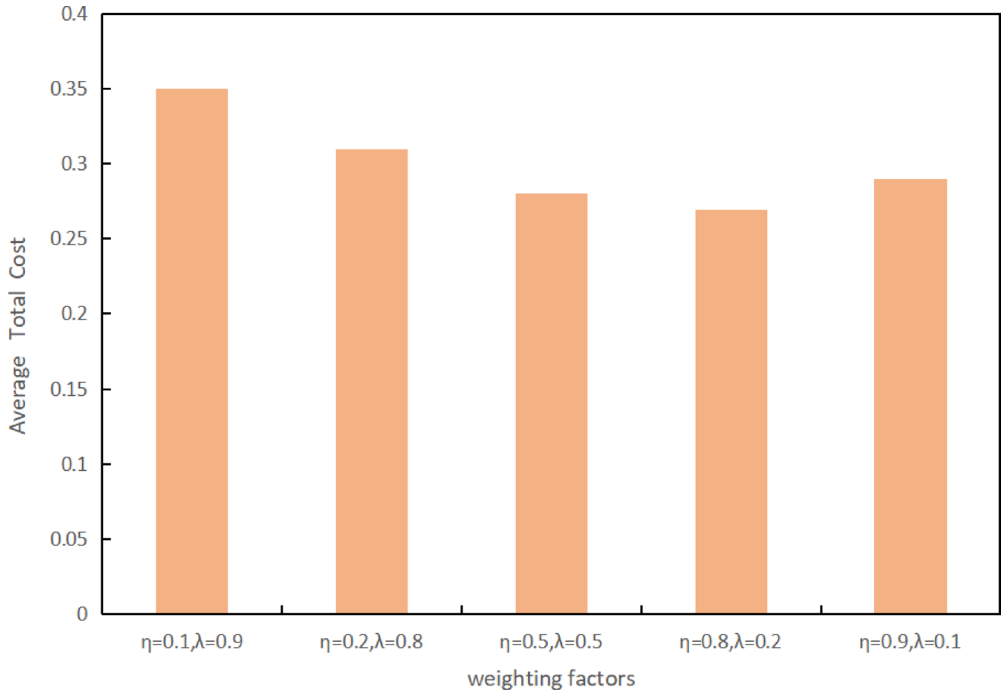


Fig. 7 Impact of weighting factors on average total cost.

the algorithm. The value $s = 0.9$ achieves the best balance between global search range and convergence stability. Therefore, the scale factor s of the sine chaotic mapping is set to 0.9 in the subsequent experiments.

Figure 6 illustrates the impact of the Lévy flight trigger threshold q on the average total cost under the same experimental setup as Fig. 5. As q increases from 0.5 to 0.8, the average total cost decreases from approximately 0.31 to 0.27, obtaining the lowest cost among the tested range at $q = 0.8$. When q exceeds 0.8, the cost gradually rises to 0.29 at $q = 0.95$. This indicates that $q = 0.8$ can properly control the activation frequency of Lévy flight, enabling the algorithm to perform large-scale jumps to escape local optima at critical moments while maintaining sufficient local exploitation ability. Therefore, the Lévy flight trigger threshold q is set to 0.8 in the subsequent experiments.

Figure 7. Impact of weighting factors on average total cost. The experiments are conducted with 20 vehicles, where each task requires 2000 Megacycles of CPU cycles and the input data size is 1.5 MB. As η increases from 0.1 to 0.9 (with λ decreasing correspondingly from 0.9 to 0.1), the average total cost decreases initially and achieves the best performance among the tested weighting configurations at $\eta = 0.8$, $\lambda = 0.2$, indicating that the proposed scheme achieves the best balance among the tested weight configurations between delay and energy consumption under this weighting configuration.

Convergence analysis

To validate the convergence behavior of the IHFOX algorithm, Fig. 8 illustrates the convergence curves of IHFOX and benchmark algorithms over 100 iterations. All algorithms are initialized with similar population distributions to ensure fair comparison. As depicted, IHFOX demonstrates superior convergence characteristics: it stabilizes after approximately 30 iterations with the lowest average total cost. In contrast, FOX, PSO, and GA require about 50, 60, and 70 iterations respectively to approach convergence. The rapid convergence of IHFOX benefits from the synergy of sine chaotic initialization, Lévy flight exploration, and standard normal distribution update, which effectively balance global exploration and local exploitation.

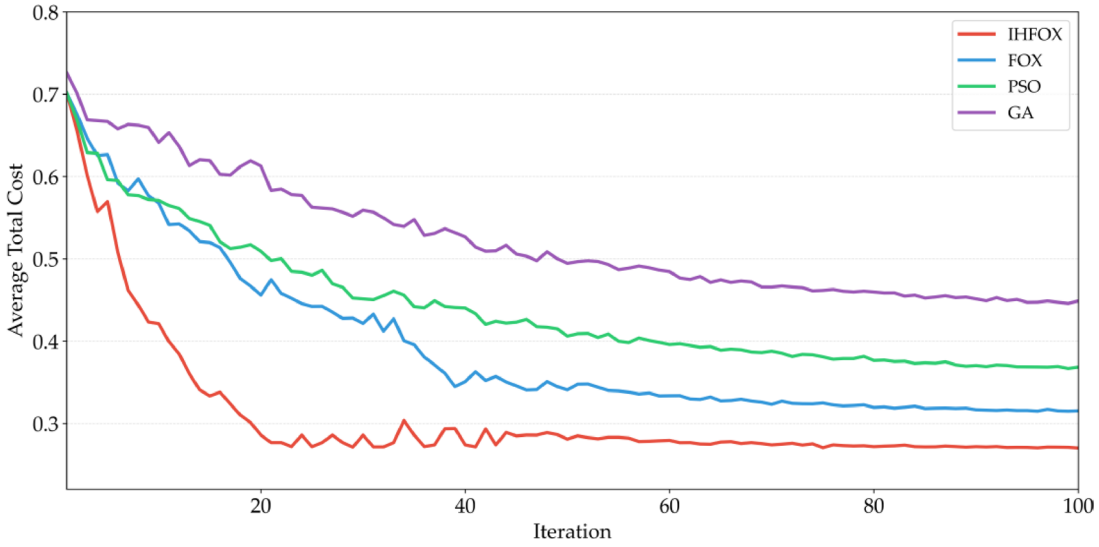


Fig. 8 Convergence comparison of IHFOX and benchmark algorithms.

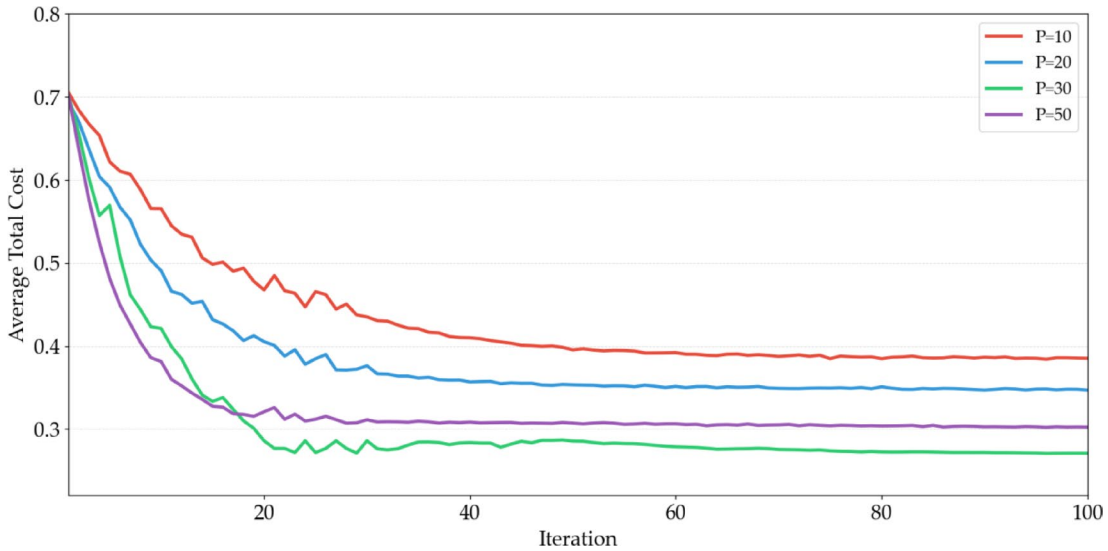


Fig. 9 Quantitative convergence summary under different population sizes.

Figure 9 Convergence curves of the proposed IHFOX algorithm under four different population sizes $P = \{10,20,30,50\}$. The vertical axis denotes the average total task offloading cost, and the horizontal axis represents the number of iterations. All curves show drastic fluctuations in the early 30 iterations due to global exploration and local optimum escape operations, while the curves become smooth and stable after the 30th iteration, entering the fine local exploitation stage. A moderate population size ($P = 30$) achieves the optimal balance: $P = 10$ suffers from insufficient initial diversity and premature convergence to suboptimal solutions, while $P = 50$ accelerates early descent but is eventually outperformed by $P = 30$ due to diminished selection pressure and redundant Lévy flight perturbations in later iterations. The discrete decision space and strict resource constraints (C3–C4) further amplify this effect for large populations, causing fragmented resource allocation across RSUs. This validates that the sine chaotic initialization strategy significantly improves the initial population diversity of IHFOX, but excessive population size dilutes the fine-grained exploitation efficiency near constraint boundaries.

Average total cost

Figure 10 shows the average total system overhead for a task required by the central processing unit cycle ranging from 1000 to 1800 Megacycles in an environment with an input data size of 1.5 MB per computation task across 20 vehicles. The graph clearly indicates that as the number of CPU cycles increases, the average total overhead for all algorithms also rises. In particular, the LOC algorithm shows the steepest increase, highlighting its inefficiency under heavy computational loads. In contrast, the IHFOX algorithm demonstrates a more stable performance advantage throughout the entire range of CPU cycles, with the lowest overhead that remains nearly constant. When a task demands 1800 Megacycles of CPU processing, the optimized scheme proposed in this paper reduces the overhead by approximately 89.0% compared to the local execution scheme, by about 67.0% compared to the average offloading scheme, by approximately 58.9% compared to the GA optimization scheme, by about 32.1% compared to the PSO optimization scheme, and by approximately 14.9% compared to the original FOX optimization scheme. These results indicate that the IHFOX algorithm significantly outperforms other algorithms in optimizing system overhead and computation efficiency and exhibits better adaptability to increases in computation resources, making it more suitable for environments with substantial resource constraints.

Figure 11 depicts the relationship between the volume of task data and the average total cost in the system, set within a context where the central processing unit cycle needed for 20 vehicles totals 2000 Megacycles. Due to the fact that tasks in the LOC scheme are only computed locally and not offloaded, the numerical value of the system's average total cost tends to stabilize, but with the maximum cost value. The delay curves for the other five offloading schemes show a gradual increase, indicating longer delays as the volume of task data grows. Under these circumstances, the delay used by IHFOX remains significantly lower than the others. When the amount of task data is 1.5 MB, compared with LOC, AOC, GA, PSO, and FOX, the optimization scheme proposed in this paper reduces the cost by approximately 89.7%, 68.3%, 54.3%, 35.3%, and 22.9% respectively. This indicates that the IHFOX algorithm has more advantages in handling large-scale data and can maintain a lower total cost.

Figure 12 shows the relationship between the number of tasks and the total average system overhead when the required CPU cycles for the task are 2000 Megacycles and the input data size for each computing task is 1.5 MB. When processing 20 tasks, IHFOX demonstrates an average total cost significantly lower than other

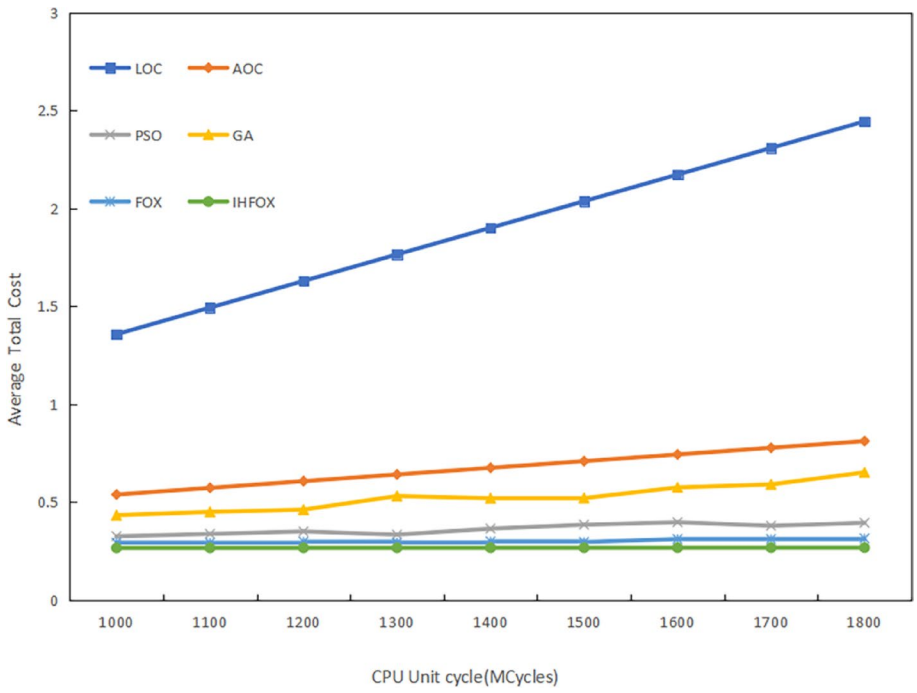


Fig. 10 The relationship between the CPU cycles needed for a task and its average total cost.

https://doi.org/10.1038/s41598-026-71807-5

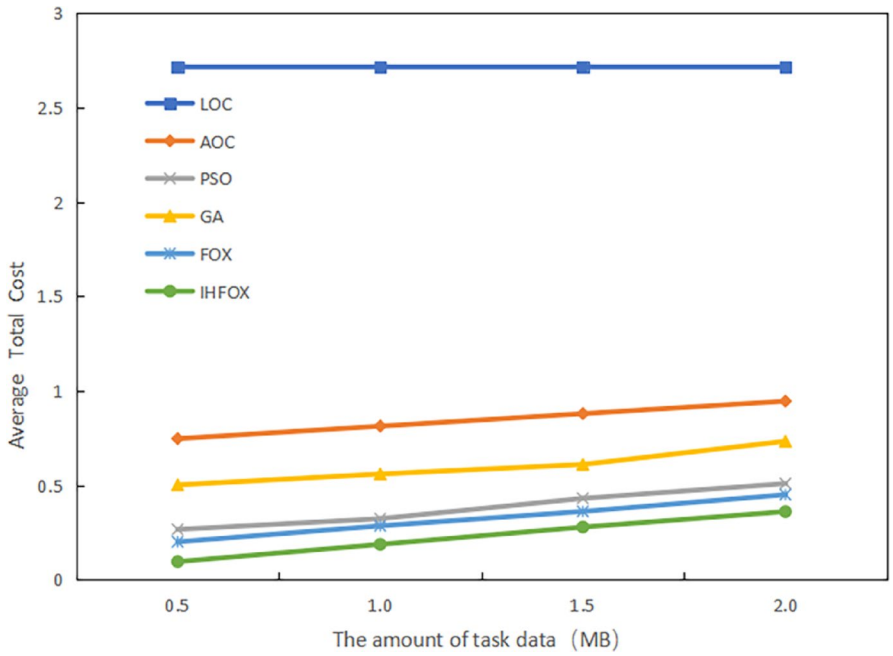


Fig. 11 The relationship between the amount of task data and average total cost.

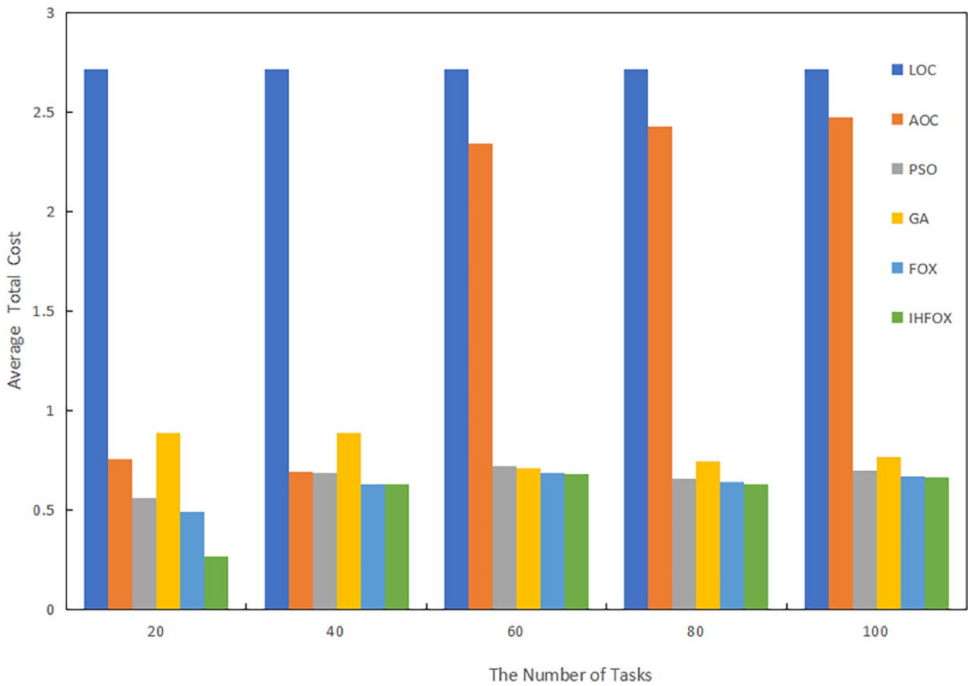


Fig. 12 The relationship between the number of tasks and the average total cost.

methods, showing a reduction of nearly 90.1% compared to LOC. This suggests that the IHFOX strategy may offer the optimal cost-benefit balance in processing small-scale tasks. As the task size increased to 80, the optimization strategy presented in this paper achieved cost reductions of approximately 76.8%, 74.0%, 15.6%, 4.7%, and 1.4% compared to LOC, AOC, GA, PSO, and FOX, respectively. It can be observed that the performance advantage of IHFOX over the traditional FOX algorithm gradually narrows as the number of tasks increases to 80. This phenomenon is reasonable and consistent with the characteristics of heuristic algorithms. When the task scale becomes large and the system load tends to be saturated, the optimization space for both algorithms is compressed, and the performance gap between well-designed heuristic algorithms will naturally decrease.

Nevertheless, IHFOX still maintains a stable and consistent improvement under high-load conditions, which verifies its effectiveness and scalability in dense IoV scenarios.

Average energy cost

Figure 13 depicts an environment with 20 vehicles, each with a computation task input data size of 1.5 MB. It can be observed from the figure that as the CPU cycle increases gradually from 1000 MCycles to 1800 MCycles, the energy consumption of all algorithms shows an upward trend. When the task requires 1800 Megacycles of CPU cycles to complete, the proposed optimization solution in this paper reduces average energy consumption by approximately 86.2%, 60.9%, 49.7%, 25.5%, and 11.1% compared to LOC, AOC, GA, PSO, and FOX, respectively. This result indicates that the IHFOX algorithm possesses higher efficiency and stability in resource-limited environments, and markedly outperforms other algorithms in optimizing system energy consumption.

Figure 14 illustrates the relationship between task data volume and the average energy consumption within the system in an environment with 20 vehicles, where the central processing unit cycle required for tasks is 2000 Megacycles. The average energy consumption of the LOC system tends to stabilize, but the energy consumption is maximal. The remaining offloading schemes show a continuous increase in average energy consumption within the system as the task data volume increases. Compared with AOC, the optimization solution proposed in this paper not only optimizes task offloading decisions but also has an effect on energy consumption. When the task data volume is 1.5 MB, the energy consumption of this solution is reduced by approximately 40.4%.

Figure 15 illustrates the relationship between the number of tasks and the average energy cost of the system when the central processing unit cycle required for each task is 2000 Megacycles and the input data size for each computation task is 1.5 MB. As the number of tasks escalates, the average energy consumption of the LOC stabilizes, yet it remains at its maximum. The remaining offloading schemes show an increasing trend in average energy consumption within the system as the task data volume increases. The optimization solution presented in this paper not only refines task offloading decisions compared to standard offloading schemes but also impacts energy consumption effectively. When the task count reaches 80, the energy consumption of the

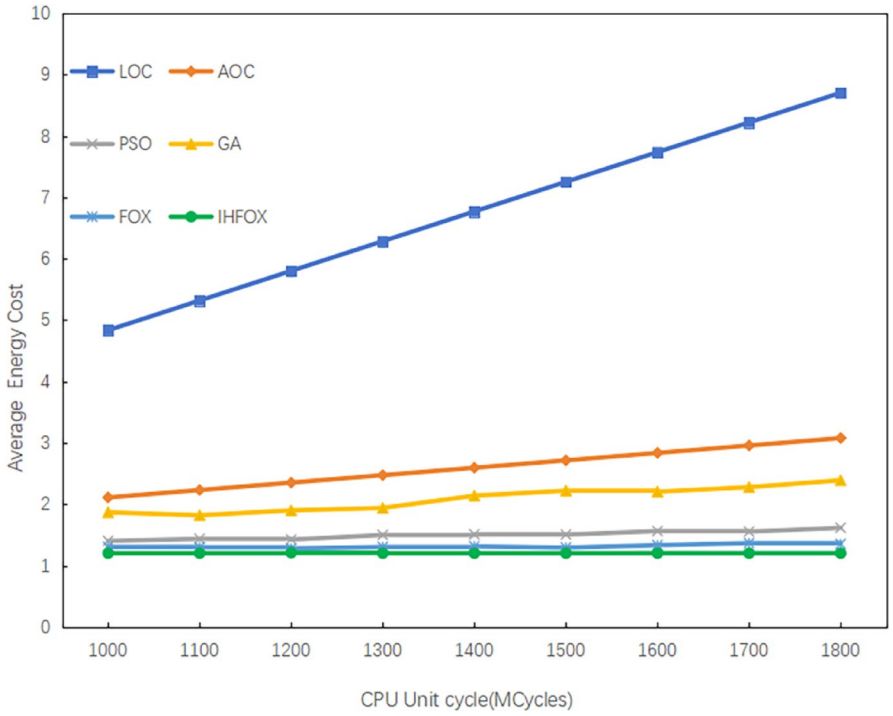


Fig. 13 The relationship between the number of CPU cycles required for a task and the average energy cost.

https://doi.org/10.1038/s41598-026-71807-5

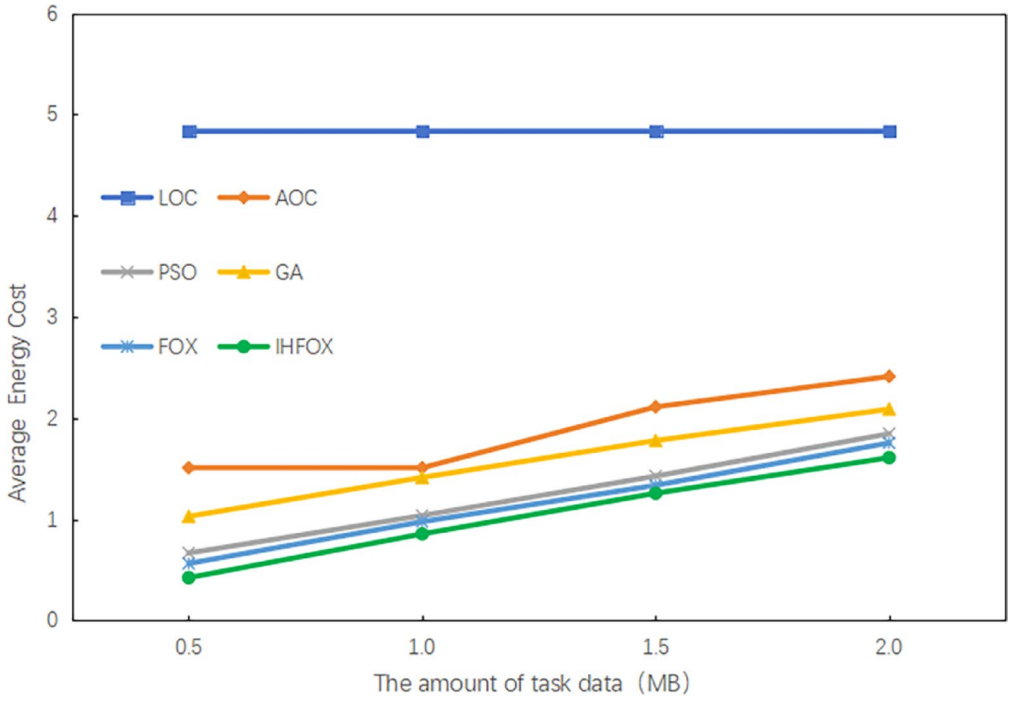
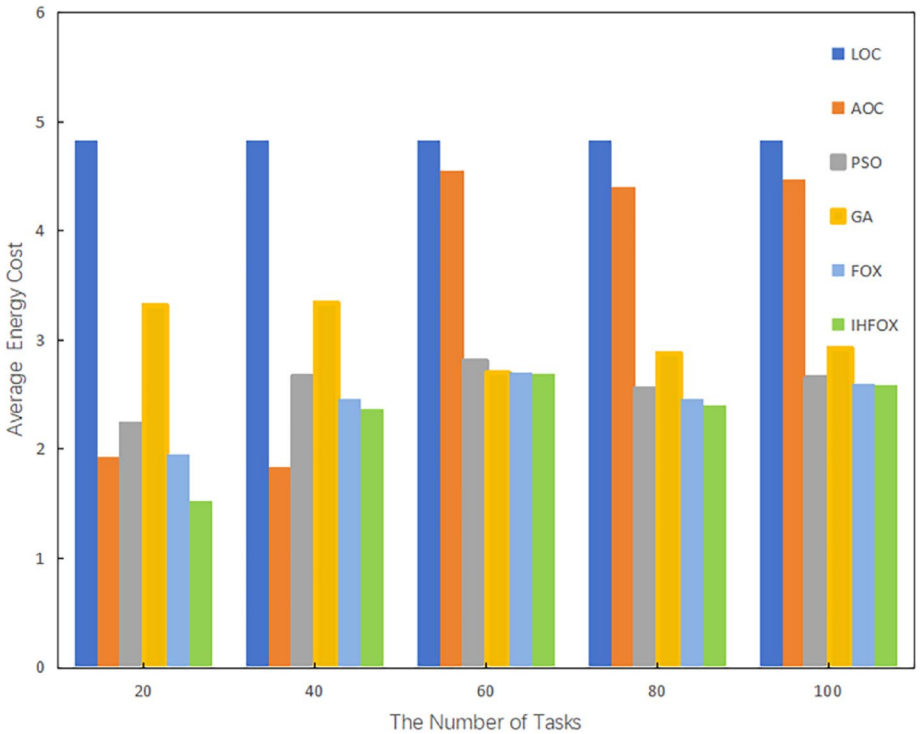


Fig. 14 The relationship between the amount of task data and the average energy cost.

Fig. 15 The relationship between the number of tasks and the average energy cost.



proposed optimization solution decreases by approximately50.2%, 45.4%, 16.5%, 6.0%, and 2.3% compared to LOC, AOC, GA, PSO and FOX, respectively.

Average delay

Figure 16 presents the average system delay under the environment of 20 vehicles, where each computation task has an input data size of 1.5 MB, and the central processing unit cycle ranges from 1000 to 1800 Megacycles. It can be observed from the figure that as the CPU cycle gradually increases from 1000 MCycles to 1800 MCycles, the delay of all algorithms shows an upward trend. The delay growth of LOC, AOC, and GA algorithms is more pronounced, while the IHFOX algorithm exhibits the lowest delay growth rate. Specifically, the average delay of the LOC algorithm increases from about 0.48 to nearly 0.9, almost doubling. In contrast, the average delay of the IHFOX algorithm has been maintained at around 0.032, and compared to the GA optimization solution, the average delay is reduced by approximately 77.6%. In conclusion, as the CPU cycle requirement for tasks escalates, the optimization solution proposed in this paper significantly outperforms other algorithms in optimizing system delay.

Figure 17 examines the relationship between the volume of task data and the average system delay in a scenario involving 20 vehicles, where the central processing unit cycle needed for 20 vehicles totals 2000 Megacycles. Due to the local execution scheme where tasks are only computed locally without offloading and without transmission delay, the average system delay tends to stabilize, but it takes the longest time. The remaining five offloading schemes show a continuous increase in average system delay with the increase of task data volume, but the fluctuation is not significant. This is because the transmission rate of the task upload link is fast, and the transmission delay can be negligible. When the task data volume is 1.5 MB, the proposed optimization solution in this paper reduces the delay by approximately 92.6%, 75.2%, 61.7%, 43.0%, and 30.5% compared to LOC, AOC, GA, PSO, and FOX, respectively. The solution proposed in this paper significantly achieves higher computation efficiency in delay optimization compared to other optimization solutions.

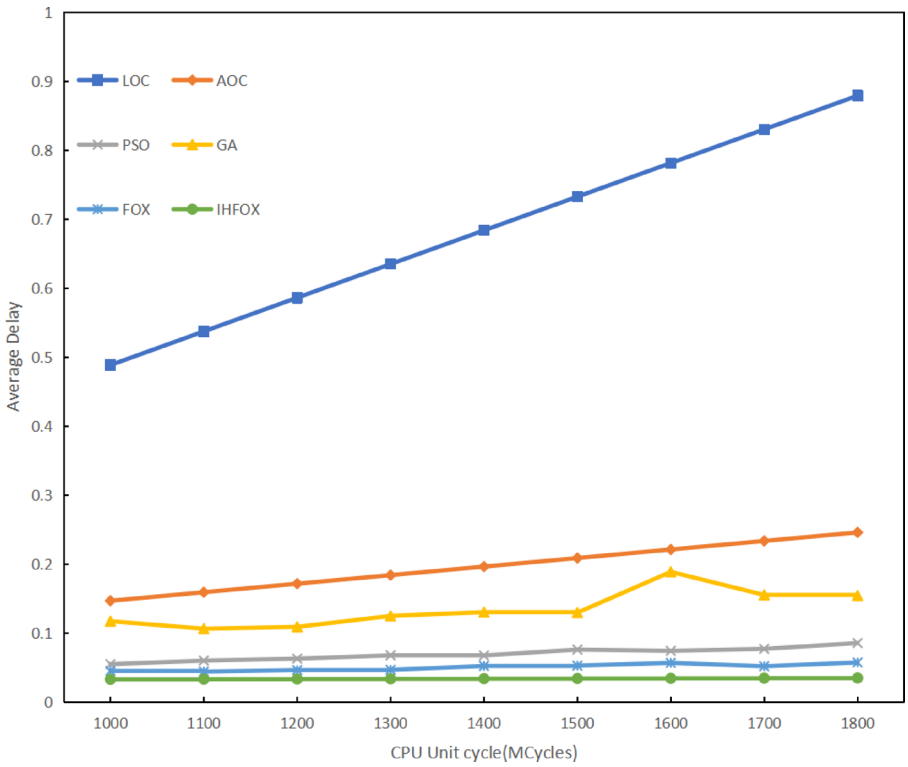


Fig. 16 The relationship between the number of CPU cycles required for a task and the average system delay.

https://doi.org/10.1038/s41598-026-71807-5

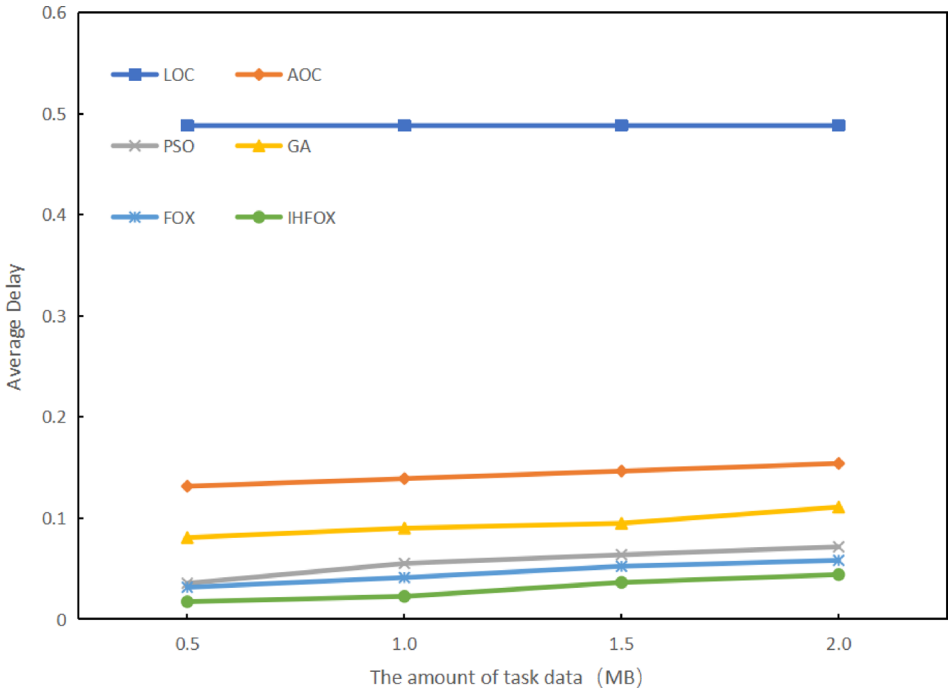


Fig. 17 The relationship between the amount of task data and the average system delay.

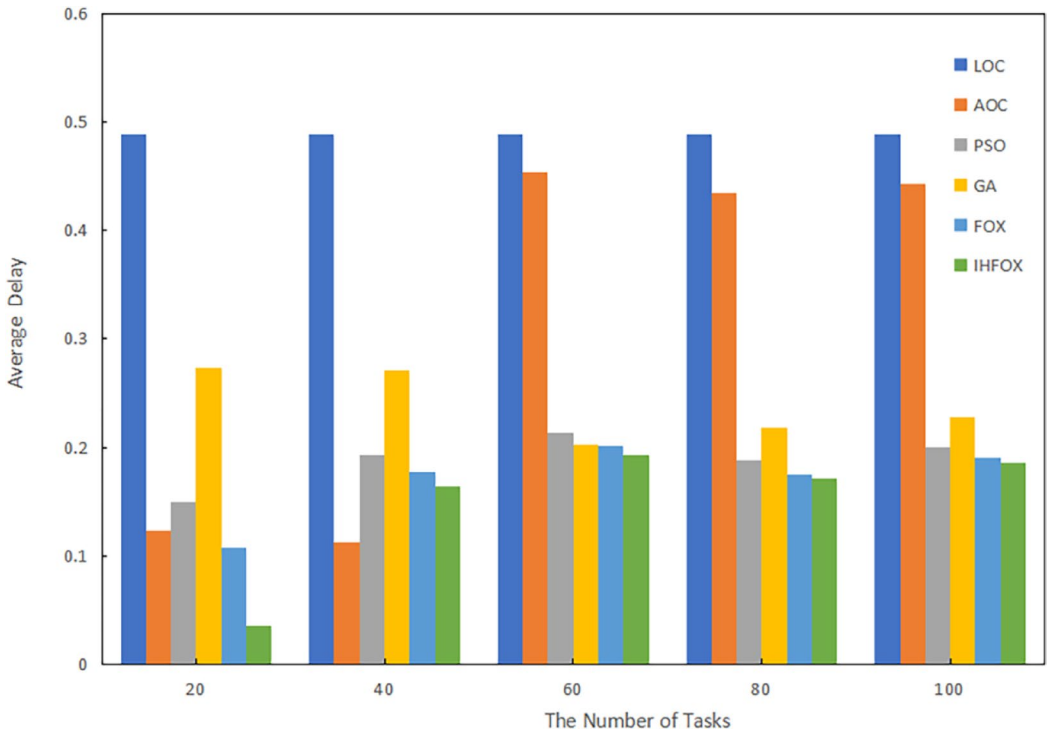


Fig. 18 The relationship between the number of tasks and the average system delay.

Figure 18 illustrates the relationship between the number of tasks and system delay when the central processing unit cycle required for tasks is 2000 Megacycles and the input data size for each computation task is 1.5 MB. As the task count rises, the average system delay associated with the local execution scheme tends to stabilize, but energy consumption is maximal; the delay cost of the AOC algorithm increases rapidly; PSO and GA show

a moderate level of delay growth; FOX and IHFOX algorithms not only demonstrate low delay when handling fewer tasks but also maintain relatively low growth rates when handling more tasks, indicating the effectiveness of their optimization strategies.

Figure 19 further analyzes the effects of different vehicle speed fluctuation intensities on the system's average total cost and task completion rate. As the speed fluctuation intensity increases from 0 to 0.3, the average total cost rises from approximately 0.278 to 0.313, while the task completion rate decreases from 0.986 to 0.943. This trend indicates that dynamic variations in vehicle speed increase the uncertainty of communication and computation, thereby raising system costs and slightly reducing the task completion rate. However, even at the maximum fluctuation intensity, the task completion rate remains above 94%, and the increase in total cost is controlled within 12%. In scenarios such as the Internet of Vehicles, where vehicles move at high speeds and link states are highly variable, the proposed algorithm maintains a high task completion rate while keeping cost growth within an acceptable range, demonstrating good robustness and practicality.

Figure 20 Analysis of the relationship between task data volume and average total cost in ablation experiments involving 20 vehicles, where the total CPU cycles required amount to 2000 Megacycles. IHFOX-w/o-Chaos represents the IHFOX algorithm without sine chaotic initialization, IHFOX-w/o-Levy denotes the IHFOX algorithm without Lévy flight exploration, and IHFOX-w/o-Norm indicates the IHFOX algorithm without standard normal distribution update. The results show that removing any single component leads to performance degradation: excluding chaotic initialization increases cost by approximately 7.8%, removing Lévy flight raises cost by about 9.3%, and omitting standard normal distribution update results in a 5.2% cost increase compared to the complete IHFOX. Among the three ablated variants, IHFOX-w/o-Norm achieves the lowest cost, suggesting that the standard normal distribution update contributes the least individually; however, the full IHFOX still outperforms all ablated versions. This validates that the fusion of three improvement strategies is synergistically designed for the triple coupled constraints of the Internet of Vehicles, and any single component removal is insufficient to achieve optimal performance.

Quantitatively, compared with the full IHFOX, removing sine chaotic initialization, Lévy flight, and normal-distribution update increases the average total cost by 7.8%, 9.3%, and 5.2%, respectively, indicating that Lévy

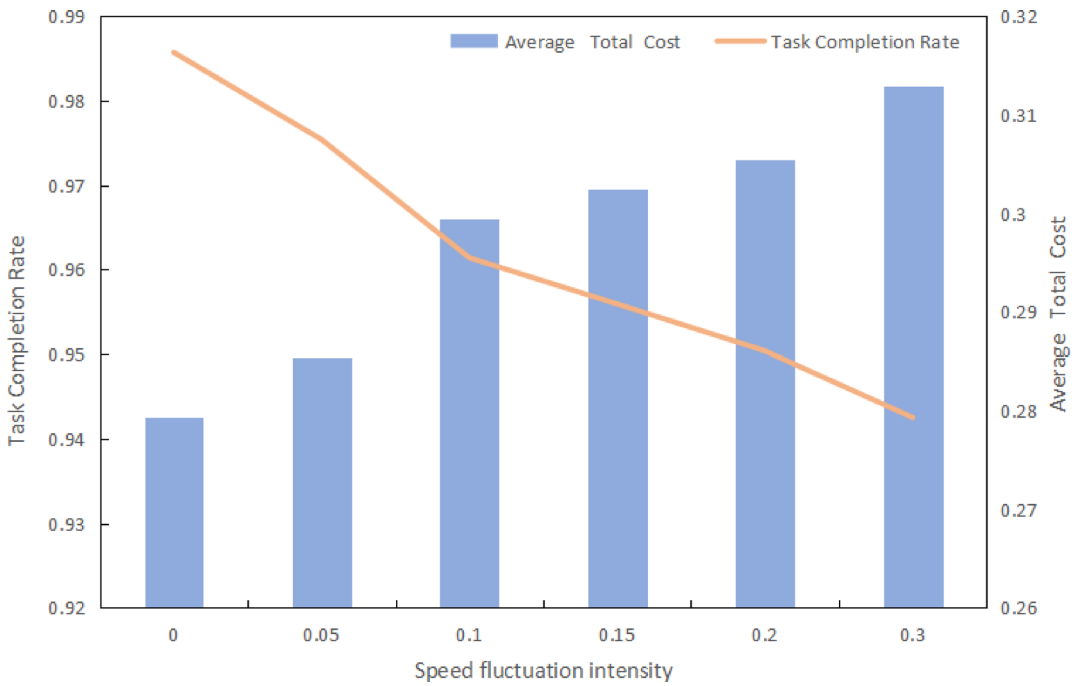


Fig. 19 The impact of speed fluctuation intensity on task completion rate and average total cost.

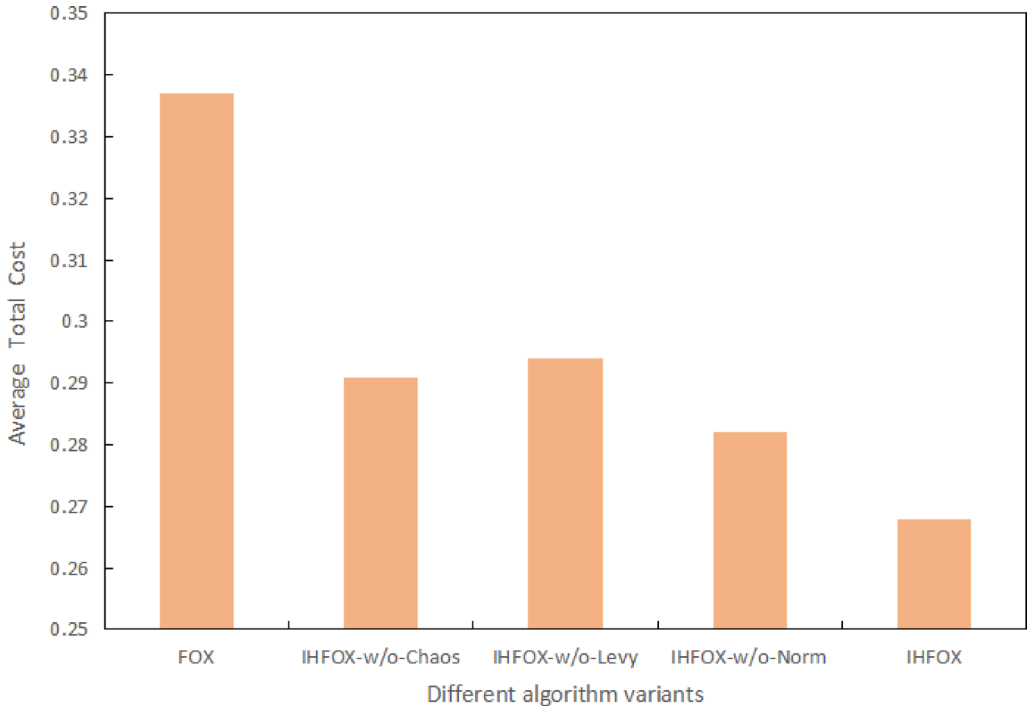


Fig. 20 Comparison of ablation studies.

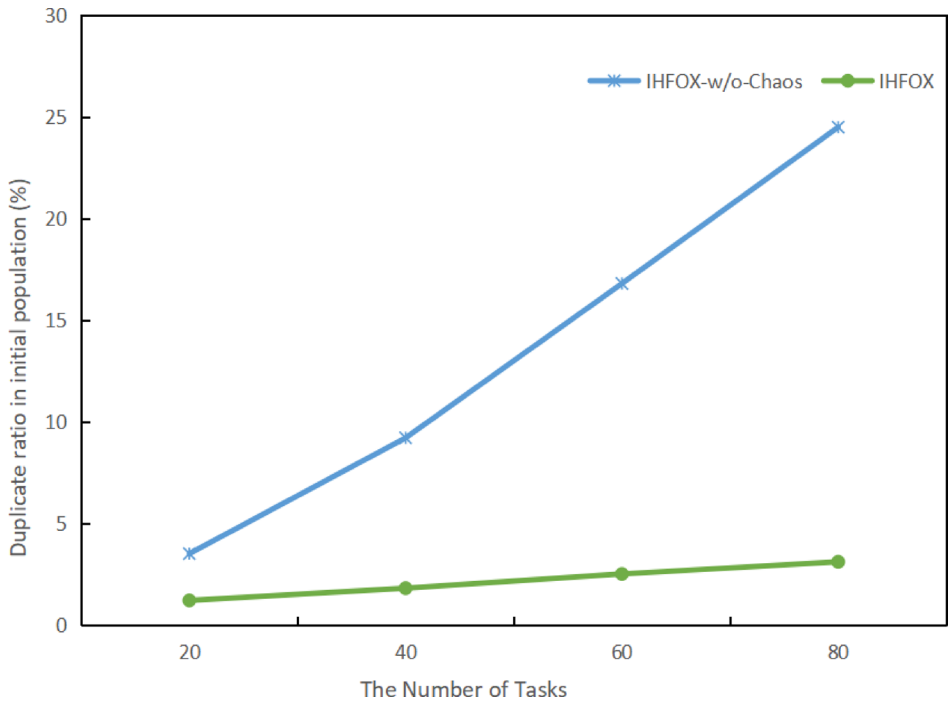


Fig. 21 Variation of initial solution duplication ratio with the number of tasks.

flight contributes the largest individual performance gain in the tested scenarios. Pairwise Wilcoxon signed-rank tests between each ablated variant and the full IHFOX yield p-values of 0.031, 0.022, and 0.041, respectively, confirming that each improvement mechanism delivers a statistically significant contribution.

To further validate the specific role of each improvement mechanism in IHFOX, we conduct three targeted experiments corresponding to the three proposed components. Figure 21 investigates the initial population

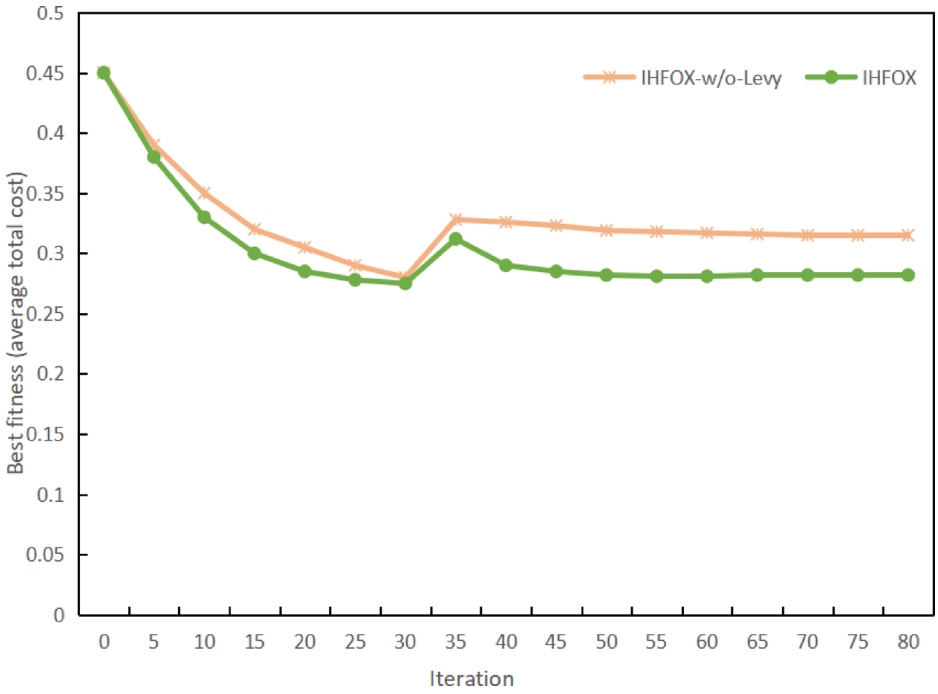


Fig. 22 Algorithm recovery after sudden channel degradation.

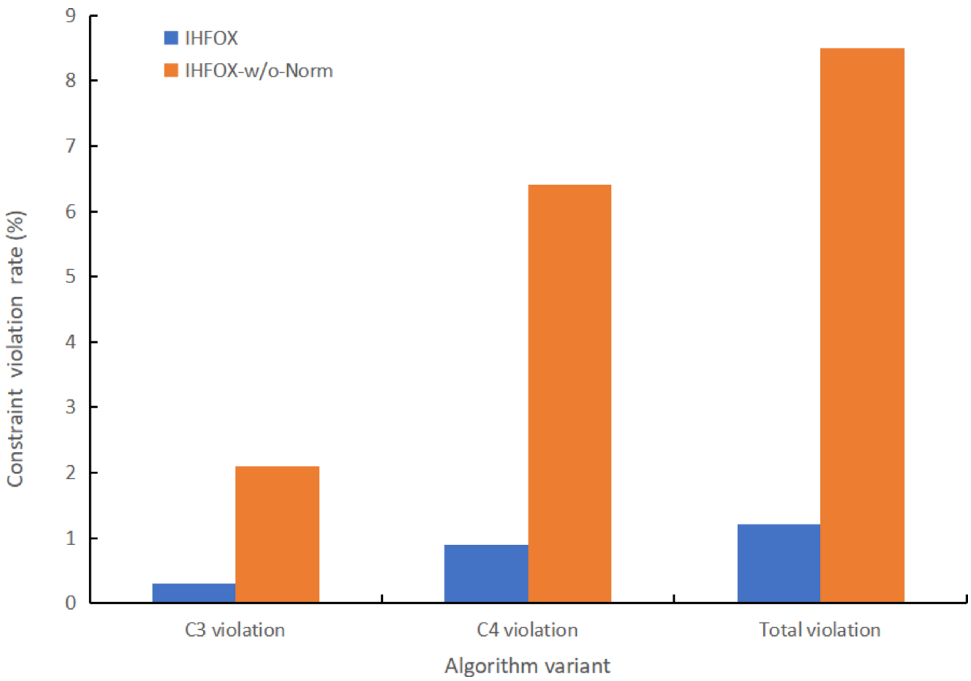


Fig. 23 Constraint violation rate under tight resources.

diversity by measuring the duplicate ratio under different numbers of vehicles, demonstrating that sine chaotic initialization effectively covers the high-dimensional offloading decision space. Figure 22 evaluates the adaptability to dynamic environments by suddenly degrading the channel condition at the 30th iteration, showing that Lévy flight enables rapid recovery from performance loss. Figure 23 examines the search stability near resource constraints under tight edge server capacity, confirming that the standard normal distribution update significantly reduces constraint violations.

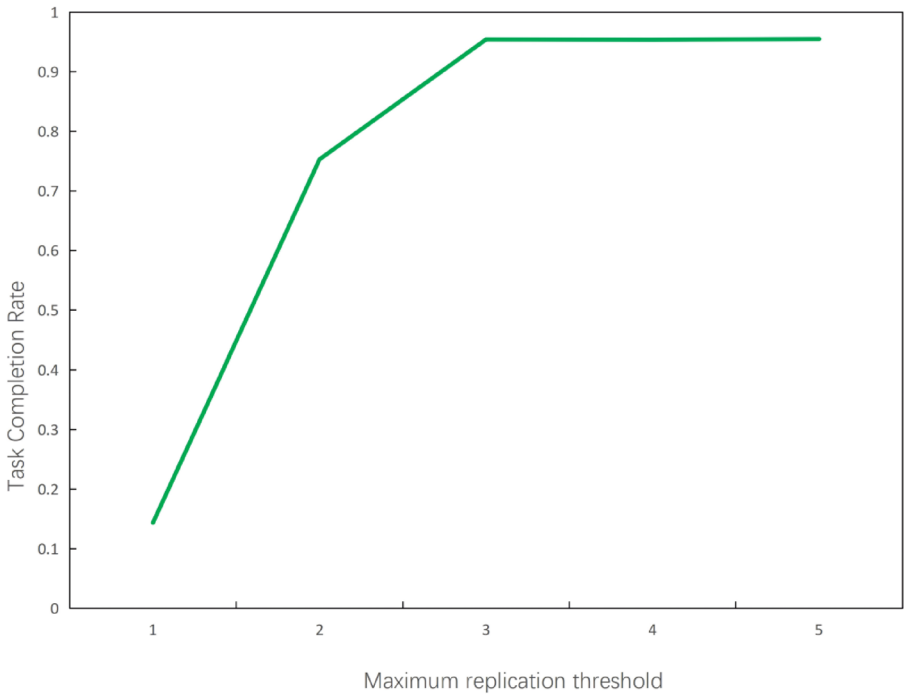


Fig. 24 Average completion rate under different numbers of replication thresholds.

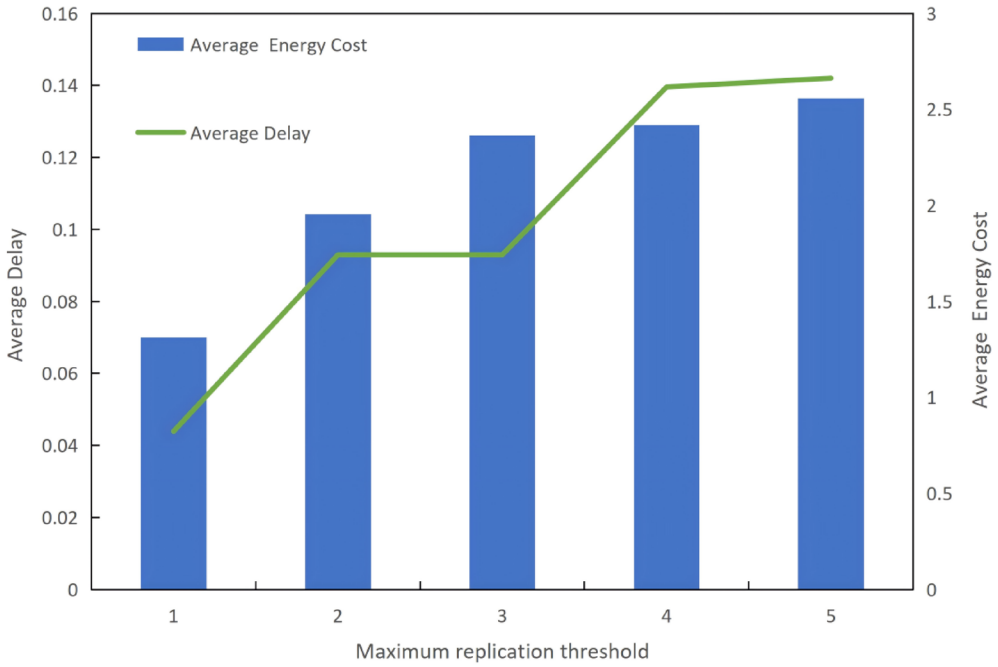


Fig. 25 Average delay and energy consumption under different replication thresholds.

Reliability

To further evaluate the performance of the proposed IHFOX algorithm in enhancing task completion reliability, this section analyzes from three perspectives: task completion rate, the impact of the replica mechanism on completion rate, and the trade-off between the number of replicas and system overhead. The experimental results are shown in Figs. 24, 25 and 26.

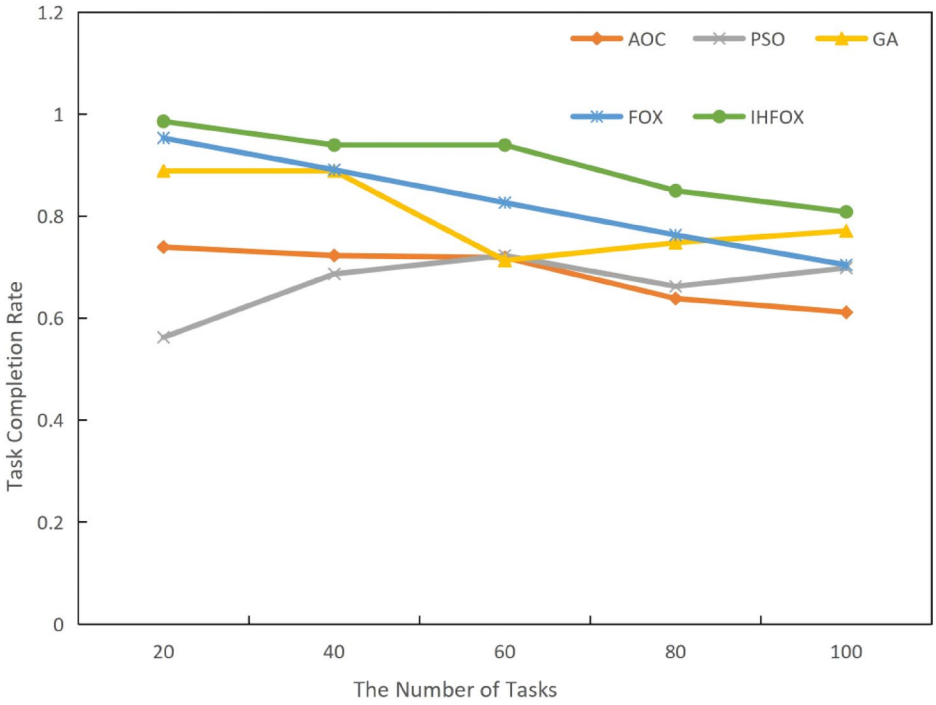


Fig. 26 Task completion rate under different task counts.

Before analyzing the experimental results, we first clarify two core concepts and their relationship. The reliability threshold r_{th}=0.98 is a hard constraint (Constraint C5) imposed on each individual task in the model, requiring that the theoretical multi-replica reliability $R_{multi}(j) = 0.98$ for each task. The task completion rate is a system-level metric obtained from simulation statistics, defined as the ratio of the number of tasks actually completed successfully to the total number of tasks. The threshold r_{th} is derived from the theoretical model and represents the lower bound of reliability for a single task, while the task completion rate reflects the real operational status of the system, which is influenced by dynamic factors such as vehicle mobility, resource contention, and random simulation events. Therefore, a task completion rate lower than r_{th} is a normal deviation between the theoretical model and the actual scenario, and does not constitute a violation of Constraint C5. All experiments in this paper satisfy the reliability constraints. In the following, we evaluate the practical performance of the proposed scheme in conjunction with the task completion rate.

Figure 24 illustrates the trend of the system's average task completion rate under different replication thresholds (i.e., the number of replicas). The experiment is set with 20 tasks, a single task data size of 1.5 MB, and a maximum offloading task count of 5 for the MEC server. As can be seen from the figure, when the replication threshold increases from 1 to 3, the task completion rate rapidly rises from approximately 14.3% to 95.4%, showing a significant improvement. This indicates that in the vehicle-to-everything (V2X) environment, where uncertainties exist in both wireless transmission and computation processes, introducing a multi-replica mechanism can effectively enhance the probability of successful task completion. When the number of replicas exceeds 3, the growth rate of the completion rate tends to level off, indicating that under the conditions set in this paper, 3 replicas are already sufficient to meet the system reliability requirements.

Figure 25 further analyzes the impact of different replica thresholds on the system's average delay and energy consumption. As the number of replicas increases, the system's average energy consumption rises from 1.31 J to 2.56 J, and the average delay also increases from approximately 0.044s to 0.142s. This trend indicates that while the replica mechanism improves task completion rates, it also incurs additional resource overhead. However, considering that the task completion rate has increased from 14.3% to over 95%, this overhead is acceptable in practical applications. Especially in scenarios with extremely high reliability requirements such as intelligent

Table 6 Statistical summary of average delay.

Algorithm	Mean	Std	95% CI	CV (%)
IHFOX	0.032	0.001	[0.031, 0.033]	3.13
FOX	0.046	0.002	[0.044, 0.048]	4.35
PSO	0.056	0.003	[0.053, 0.059]	5.36
GA	0.143	0.007	[0.136, 0.150]	4.90
AOC	0.229	0.010	[0.219, 0.239]	4.37
LOC	0.869	0.032	[0.837, 0.901]	3.68

Table 7 Statistical summary of average energy consumption.

Algorithm	Mean	Std	95% CI	CV (%)
IHFOX	1.195	0.035	[1.160, 1.230]	2.93
FOX	1.365	0.042	[1.323, 1.407]	3.08
PSO	1.715	0.051	[1.664, 1.766]	2.97
GA	2.580	0.068	[2.512, 2.648]	2.64
AOC	2.965	0.089	[2.876, 3.054]	3.00
LOC	8.683	0.305	[8.378, 8.988]	3.51

transportation and autonomous driving, moderate redundant replicas are a necessary means to ensure system stability.

Figure 26 illustrates the performance of the IHFOX algorithm compared to other benchmark algorithms (AOC, PSO, GA, FOX) in terms of task completion rate under varying numbers of tasks. The experiment was set with a CPU cycle requirement of 2000 M for a single task and a data volume of 1.5 MB. As evident from the figure, IHFOX maintains the highest completion rate across all task sizes. When the number of tasks is 20, the completion rate of IHFOX reaches 98.57%, significantly outperforming FOX (95.31%), GA (88.85%), AOC (74.00%) and PSO (56.23%). As the number of tasks increases, the completion rates of all algorithms decline, but the decrease is minimal for IHFOX, demonstrating stronger scalability and robustness. This is attributed to IHFOX's comprehensive consideration of replica redundancy, resource constraints, and vehicle mobility during task scheduling, thereby maintaining a high task success rate in dynamic environments.

Statistical significance analysis

To verify the statistical reliability of the above experimental results, this subsection provides a comprehensive statistical summary based on the 10 independent runs performed for each configuration.

Table 5. Statistical summary of average total cost.

As shown in Table 5, IHFOX achieves the lowest mean cost (0.271) with the smallest standard deviation (0.008) and coefficient of variation (2.95%), indicating that it provides both the best solution quality and the highest stability among all compared algorithms. Notably, the CV of IHFOX is substantially lower than those of GA (4.70%) and AOC (5.11%), confirming its robustness against stochastic variations in task generation and channel conditions.

Similarly, Tables 6 and 7 present the statistical summaries for average delay (in seconds) and average energy consumption (in joules), respectively, under the same representative scenario.

It can be observed that IHFOX also attains the lowest mean values and competitive CVs for both delay and energy consumption, further corroborating its stability.

To quantitatively verify whether the performance improvement of IHFOX has statistical significance rather than random fluctuation, pairwise Wilcoxon signed-rank non-parametric tests are conducted between IHFOX and each baseline algorithm (FOX, PSO, GA, AOC, LOC) under all parameter configurations (different CPU cycles, task data volume, vehicle numbers).

Test setup: The significance level is set as $\alpha = 0.05$. The null hypothesis H_0 is defined as follows: (i) For average total cost: There is no significant difference in average total cost between IHFOX and the compared baseline

algorithm; (ii) For average delay: There is no significant difference in average delay between IHFOX and the compared baseline algorithm; (iii) For average energy consumption: There is no significant difference in average energy consumption between IHFOX and the compared baseline algorithm.

Test conclusion: For all pairwise comparison groups, the calculated p-values are consistently less than 0.05, so we fully reject the null hypothesis H_0.

Statistical conclusion: For all three metrics—average total cost, average delay, and average energy consumption—the pairwise Wilcoxon signed-rank tests yield p-values consistently below 0.05 across all parameter configurations and baseline comparisons. Therefore, the performance improvements brought by IHFOX are statistically significant and cannot be attributed to simulation randomness.

Discussion

Core mechanisms

Experimental results demonstrate that IHFOX maintains stable advantages in system overhead, delay, and energy consumption across various task scales and data sizes. This performance stems from the synergy between the dual-dimension reliability model and the improved optimization algorithm. The transmission-computation combined reliability model accurately characterizes task failure risks. The multi-replica redundancy mechanism satisfies system reliability requirements with the minimum number of replicas. Sine chaotic initialization, Lévy flight, and standard normal distribution update operate complementarily. They effectively balance global exploration and local exploitation, rendering the algorithm more adaptable to the highly dynamic and heavily constrained IoV environment.

Analysis of abnormal phenomena

The optimization gain diminishes under high load. As shown in Fig. 12, when task count increases from 20 to 80, the performance gap between IHFOX and FOX narrows from 14.9% to 1.4%. This reflects the shrinking optimization space under resource saturation. As Constraint C4 (server resource capacity) tightens, the feasible offloading decision space is severely restricted. Consequently, the differential exploration space of heuristic algorithms decreases. This is consistent with the complexity analysis in Sect. 3.4. The marginal gain of metaheuristic improvements declines when system load approaches saturation. This indicates that for ultra-dense IoV scenarios, algorithm optimization alone faces a performance ceiling. Supplementary measures, such as denser RSU deployment or hierarchical edge-cloud offloading, become necessary.

To further diagnose this saturation effect, we examine resource utilization under 80 tasks: the average MEC server load reaches 94.7% of F_{max}, and the constraint repair mechanism (Step 4 in Sect. 3.2) triggers in over 80% of iterations, confirming that the feasible decision space is severely compressed. Under such conditions, the minimum achievable cost is bounded by resource capacity rather than search efficacy; all offloading schemes gradually converge toward the same resource-saturation floor as task density grows further. Even so, IHFOX retains a small but consistent advantage over FOX due to its superior initialization and boundary search, while the simple AOC baseline degrades faster under overload due to its naive load-balancing logic.

Practical deployment constraints

This study simplifies several practical deployment factors. We assume fiber connections between RSUs and MEC servers feature high bandwidth, low delay, and zero packet loss. We do not consider fiber link failures, congestion, or power consumption. We also assume ideal parallel multi-replica uploads without self-interference. The DSRC channel follows the theoretical Rayleigh fading model, whereas real networks suffer from burst packet

loss and hidden terminals. Consequently, actual reliability falls below theoretical values. Although distributed multi-replica deployment improves reliability, it exposes sensitive vehicle data to multiple RSUs. This introduces risks of data leakage and false result injection. We will address these security issues, including data encryption optimization and injection attack prevention, in future work.

Practical implications

The optimal trade-off between reliability and system overhead is achieved when the number of replicas is set to 3. From the observed results, we note that when channel conditions are favorable and vehicle speeds are low, a smaller number of replicas (e.g., 1–2) may already provide sufficient reliability gains, whereas in scenarios with high mobility or degraded SINR, increasing the replica count to 3 yields a more notable improvement in task completion rate. This observation suggests a potential direction for designing adaptive replica strategies, although further quantitative criteria (e.g., SINR or speed thresholds) would be required to formulate a definitive decision rule. When vehicle density becomes excessively high and the system approaches saturation, expanding edge computing resources is more effective than further algorithm optimization for improving system performance.

Conclusion

This paper comprehensively addresses computation offloading for multi-vehicle, multi-task vehicular edge networks. We pay particular attention to mobility, energy consumption, and delay. Our vehicle-road-cloud collaborative offloading model and the Improved Hybrid Fox Optimization (IHFOX) algorithm optimize resource allocation and system performance. They accelerate convergence toward stable near-optimal feasible solutions and significantly improve computation offloading efficiency and reliability. Simulation results validate the effectiveness of our strategy, particularly in reducing system overhead, delay, and energy consumption.

We plan to investigate partial offloading of divisible tasks in IoV computing. We will also address challenges more closely related to mobility. Future research will further evaluate IHFOX performance under diverse network conditions and in practical application scenarios. Additionally, security and privacy considerations in IoV edge computing scenarios require careful attention. We anticipate that these efforts will advance IoV technology and provide strong technical foundations for intelligent transportation systems.

Author contributions Conceptualization, Y.L. and Q.S.; methodology, Y.L. and Q.S.; software, Y.L. and B.Y.; validation, B.Y. and C.L.; investigation, F.L. and C.L.; writing—original draft preparation, Q.S. and F.L.; writing—review and editing, Y.L. and Q.S.; supervision, Y.L. and Q.S.; All authors have read and agreed to the published version of the manuscript.

Funding Key Scientific Research Project of the Education Department of Jilin Province: JJKH20251099KJ.

Data availability The source code of this paper is publicly available in the GitHub repository: https://github.com/SqChao/IHFOX.git.

Declarations

Competing interests The authors declare no competing interests.

References

1. Li, K. et al. Computation Offloading in Resource-Constrained Multi-Access Edge Computing. *IEEE Trans. Mob. Comput.* **23**, 10665–10677. https://doi.org/10.1109/TMC.2024.3383041 (2024).
2. Sun, G. et al. Profit Maximization of Independent Task Offloading in MEC-Enabled 5G Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **25**, 16449–16461 (2024).
3. Fan, C. et al. Auto-updating intrusion detection system for vehicular network: A deep learning approach based on cloud-edge-vehicle collaboration. *IEEE Trans. Veh. Technol.* **73**(10), 15372–15384. https://doi.org/10.1109/tvt.2024.3399219 (2024).
4. Wang, X. et al. AMTOS: An ADMM-Based Multilayer Computation Offloading and Resource Allocation Optimization Scheme in IoV-MEC System. *IEEE Internet Things J.* **11**, 30953–3096410 (2024).
5. Farimani, M. K. et al. Deadline-Aware Task Offloading in Vehicular Networks Using Deep Reinforcement Learning. *Expert Syst. Appl.* **249**, 123622. https://doi.org/10.1016/j.eswa.2024.123622 (2024).
6. Wu, H. et al. Collaborative caching relay algorithm based on recursive deep reinforcement learning in mobile vehicle edge network. *Ad Hoc Netw.* **152**, 103313 (2024).
7. Xia, Z., Dong, L. & Jiang, F. A spiking reinforcement trajectory planning for UAV-assisted MEC systems. *IEEE Access* **12**, 54452–54465. https://doi.org/10.1109/ACCESS.2024.3389288 (2024).
8. Dai, P. et al. Meta reinforcement learning for multi-task offloading in vehicular edge computing. *IEEE Trans. Mob. Comput.* **23**(3), 2123–2138 (2023).
9. Qi, K. et al. Reconfigurable intelligent surface assisted VEC based on multi-agent reinforcement learning. *IEEE Commun. Lett.* https://doi.org/10.1109/LCOMM.2024.3451182 (2024).
10. Tong, Z. et al. Computation offloading for energy efficiency maximization of sustainable energy supply network in IIoT. *IEEE Trans. Sustain. Comput.* **9**(2), 128–140. https://doi.org/10.1109/TSUSC.2023.3313770 (2023).
11. Ye, H., Li, J. & Lu, Q. Deep reinforcement learning for dependent task offloading in multi-access edge computing. *IEEE Access* **12**, 166281–166297 (2024).
12. Tao, M., Li, X., Ota, K. & Dong, M. Single-Cell Multiuser Computation Offloading in Dynamic Pricing-Aided Mobile Edge Computing. *IEEE Trans. Comput. Social Syst.* **11** (2), 3004–3014. https://doi.org/10.1109/TCSS.2023.3308563 (2024).
13. Guo, M. et al. Joint scheduling and offloading schemes for multiple interdependent computation tasks in mobile edge computing. *IEEE Internet Things J.* **11**, 5718–5730 (2023).
14. Liu, D., Sun, F., Wang, W. & Dev, K. Distributed computation offloading with low latency for artificial intelligence in vehicular networking. *IEEE Commun. Stand. Mag.* **7**, 74–80. https://doi.org/10.1109/mcomstd.0003.2100100 (2023).
15. Hazarika, B., Singh, K., Biswas, S. & Li, C.-P. DRL-based resource allocation for computation offloading in IoV networks. *IEEE Trans. Ind. Inform.* https://doi.org/10.1109/tii.2022.3168292 (2022).
16. Geng, L. et al. Deep-Reinforcement-Learning-Based Distributed Computation Offloading in Vehicular Edge Computing Networks. *IEEE Internet Things J.* **10** (14), 12416–12433. https://doi.org/10.1109/JIOT.2023.3247013 (2023).
17. Zhou, X. et al. DAG-based dependent tasks offloading in MEC-enabled IoT with soft cooperation[J]. *IEEE Trans. Mob. Comput.* **23** (6), 6908–6920 (2023).
18. Shi, J., Du, J., Wang, J., Wang, J. & Yuan, J. Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning. *IEEE Trans. Veh. Technol.* https://doi.org/10.1109/tvt.2020.3041929 (2020).
19. Mirjalili, S. Genetic Algorithm. In *Studies in Computational Intelligence,Evolutionary Algorithms and Neural Networks* 43–55 (2019).
20. Fu, H. et al. An Improved Genetic Algorithm with Chromosome Replacement and Rescheduling for Task Offloading. *Int. J. Adv. Comput. Sci. Appl. (IJACSA)* (9), https://doi.org/10.14569/IJACSA.2023.01409107.
21. Chen, C. et al. Delay-optimized V2V-based computation offloading in urban vehicular edge computing and networks. *IEEE Access* https://doi.org/10.1109/access.2020.2968465 (2020).
22. Li, C., Zhang, Y. & Luo, Y. DQN-enabled content caching and quantum ant colony-based computation offloading in MEC. *Appl. Soft Comput.* 109900. https://doi.org/10.1016/j.asoc.2022.109900 (2023).
23. Yu, H. et al. Multi-timescale multi-dimension resource allocation for NOMA-edge computing-based power IoT with massive connectivity. *IEEE Trans. Green Commun. Netw.* https://doi.org/10.1109/tgcn.2021.3076582 (2021).
24. Liu, C.-F., Bennis, M., Debbah, M. & Poor, H. V. Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing. *IEEE Trans. Commun.* https://doi.org/10.1109/tcomm.2019.2898573 (2019).
25. Sun, J., Gu, Q., Zheng, T., Dong, P. & Qin, Y. Joint communication and computing resource allocation in vehicular edge computing. *Int. J. Distrib. Sens. Netw.* **15**155014771983785. https://doi.org/10.1177/1550147719837859 (2019).
26. Gao, H., Wang, X., Wei, W., Al-Dulaimi, A. & Xu, Y. Com-DDPG: Task offloading based on multiagent reinforcement learning for information-communication-enhanced mobile edge computing in the Internet of Vehicles. *IEEE Trans. Veh. Technol.* **73**(1), 348–361. https://doi.org/10.1109/TVT.2023.3309321 (2024).

27. Zheng, Y., Zhou, H., Chen, R., Jiang, K. & Cao, Y. SAC-based Computation Offloading and Resource Allocation in Vehicular Edge Computing, IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), New York, NY, USA, pp. 1–6, (2022). https://doi.org/10.1109/INFOCOMWKSHPS54753.2022.9798187
28. Yao, L., Xu, X., Bilal, M. & Wang, H. Dynamic edge computation offloading for Internet of Vehicles with deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **24**(11), 12991–12999. https://doi.org/10.1109/TITS.2022.3178759 (2023).
29. Sun, Z. et al. BARGAIN-MATCH: A Game-Theoretical Approach for Resource Allocation and Task Offloading in Vehicular Edge Computing Networks. *IEEE Trans. Mob. Comput.* **23**, 1655–1673 (2023).
30. Zadobrischi, E. & Havriliuc, Ș. Enhancing scalability of C-V2X and DSRC vehicular communication protocols with LoRa 2.4 GHz in the scenario of urban traffic systems. *Electronics* **13**(14), 2845. https://doi.org/10.3390/electronics13142845 (2024).
31. Zhou, T. et al. Narrowbeam channel measurements and characterization in vehicle-to-infrastructure scenarios for 5G-V2X communications. *IEEE Internet Things J.* **11**(9), 16074–16086. https://doi.org/10.1109/JIOT.2024.3352116 (2024).
32. Liang, R. et al. A Multi-Head Ensemble Multi-Task Learning Approach for Dynamical Computation Offloading. In Proceedings of the GLOBECOM 2023 – IEEE Global Communications Conference, 6079–6084. (2023).
33. Yin, L. et al. Joint task offloading and resources allocation for hybrid vehicle edge computing systems[J]. *IEEE Trans. Intell. Transp. Syst.* **25** (8), 10355–10368 (2024).
34. Shu, W., Feng, X. & Guo, C. An adaptive alternating direction method of multipliers for vehicle-to-everything computation offloading in cloud–edge collaborative environment. *IEEE Internet Things J.* **11**(22), 35802–35811 (2024).
35. Luo, Q., Li, C., Luan, T. H., Shi, W. & Wu, W. Self-learning based computation offloading for Internet of Vehicles: Model and algorithm. *IEEE Trans. Wirel. Commun.* https://doi.org/10.1109/twc.2021.3071248 (2021).
36. Cao, Z. et al. Dependent task offloading in edge computing using GNN and deep reinforcement learning. *IEEE Internet Things J.* **11**(12), 21632–21646 (2024).
37. Mohammed, H. & Rashid, T. FOX: A FOX-inspired optimization algorithm. *Appl. Intell.* https://doi.org/10.1007/s10489-022-03533-0 (2023).
38. ALRahhal, H. & Jamous, R. AFOX: A new adaptive nature-inspired optimization algorithm. *Artif. Intell. Rev.* https://doi.org/10.1007/s10462-023-10542-z (2023).
39. Mirjalili, S. SCA: A sine cosine algorithm for solving optimization problems. *Knowl. Based Syst.* **96**, 120–133 (2016).
40. Yang, X. S. & Deb, S. Cuckoo search via Lévy flights[C]//2009 World congress on nature & biologically inspired computing (NaBIC). Ieee, : 210–214. (2009).
41. Liang, J. J. et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evol. Comput.* **10**(3), 281–295. https://doi.org/10.1109/TEVC.2005.857610 (2006).
42. Jumaah, M. A., Ali, Y. H. & Rashid, T. A. An improved FOX optimization algorithm using adaptive exploration and exploitation for global optimization. *PLoS One* **20**(9), e0331965. https://doi.org/10.1371/journal.pone.0331965 (2025).
43. Hai-ping, H., Yan, Y., Zhu, Q. & Zheng, G. Generation and Frame Characteristics of Predefined Evenly-Distributed Class Centroids for Pattern Classification. IEEE Access,IEEE Access (2021).
44. Ke, H., Wang, H., Sun, W. & Sun, H. Adaptive computation offloading policy for multi-access edge computing in heterogeneous wireless networks. *IEEE Trans. Netw. Serv. Manag.* https://doi.org/10.1109/tnsm.2021.3118696 (2022).
45. Cong, Y. L., Sun, W. X. & Yue, K. Research on Task Offloading Strategy of Internet of Vehicles Based on Improved Hybrid Genetic Algorithm. *J. Commun.* 77–85 (2022).

Authors and Affiliations

Yubao Liu[1] · Quanchao Sun[1] · Bocheng Yan[1] · Fengru Li[1] · Chenhao Li[1]

✉ Yubao Liu
liuyb@ccu.edu.cn

[1] College of Computer Science and Technology, Changchun University, Changchun 130022, China