计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
[doi: 10.15888/j.cnki.csa.010104] [CSTR: 32024.14.csa.010104]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

车联网任务卸载的动态双种群哈里斯鹰优化①

蒋庆龙[1], 金子龙[2]

[1](南京信息工程大学 软件学院, 南京 210044)
[2](浙江理工大学 信息科学与工程学院 (网络空间安全学院), 杭州 310018)
通信作者: 金子龙, E-mail: zljin@outlook.com

摘　要: 在车联网 (Internet of Vehicles, IoV) 与移动边缘计算 (mobile edge computing, MEC) 深度融合的背景下, 如何在保障低时延与高可靠服务质量的同时实现高效任务卸载成为关键问题. 哈里斯鹰优化算法 (Harris hawks optimization, HHO) 虽具较强的全局搜索能力, 但在种群初始化、探索-开发切换及多样性维持方面仍存在不足, 易陷入早熟收敛. 为此, 本文提出一种动态双种群哈里斯鹰优化算法 (dynamic dual-population Harris hawks optimization, DDHHO). 该算法引入动态双种群协同演化机制 (DDPC), 并结合 L-T 混沌初始化、非线性逃逸能量及非线性跳跃策略, 实现全局探索与局部开发的自适应平衡. 实验结果表明, 在车联网 MEC 任务卸载场景中, DDHHO 在系统总代价指标上较混合策略哈里斯鹰优化算法 (MSHHO)、原始 HHO、MASSFOA、IPSO 与 PSO 分别降低约 2.5%、3.2%、4.9%、6.0% 和 7.9%; 在时延与能耗联合优化中亦展现出更快的收敛速度与更高的稳定性. 研究结果验证了 DDHHO 的有效性与优越性, 为车联网 MEC 资源管理提供了一种高效、稳定且可扩展的优化方案.

关键词: 移动边缘计算; 车联网; 任务卸载; 哈里斯鹰优化算法; 动态双种群协同演化机制

引用格式: 蒋庆龙,金子龙.车联网任务卸载的动态双种群哈里斯鹰优化.计算机系统应用. http://www.c-s-a.org.cn/1003-3254/10104.html

Dynamic Dual-population Harris Hawks Optimization for Task Offloading in Internet of Vehicles

JIANG Qing-Long[1], JIN Zi-Long[2]

[1](School of Software, Nanjing University of Information Science & Technology, Nanjing 210044, China)
[2](School of Information Science and Engineering (School of Cyber Science and Technology), Zhejiang Sci-tech University, Hangzhou 310018, China)

Abstract: With the deep integration of the Internet of Vehicles (IoV) and mobile edge computing (MEC), achieving efficient task offloading while ensuring low latency and high reliability has become a key challenge. Although the Harris hawks optimization (HHO) algorithm demonstrates strong global search capability, it still suffers from limitations in population initialization, exploration–exploitation transition, and diversity maintenance, making it prone to premature convergence. To address these issues, this study proposes a dynamic dual-population Harris hawks optimization (DDHHO) algorithm. The proposed algorithm introduces a dynamic dual-population co-evolution (DDPC) mechanism, combined with L-T chaotic initialization, nonlinear escape energy, and a nonlinear jump strategy, to adaptively balance global exploration and local exploitation. Experimental results show that in the IoV-MEC task offloading scenario, DDHHO reduces the total system cost by approximately 2.5%, 3.2%, 4.9%, 6.0%, and 7.9% compared with the mixed-strategy HHO (MSHHO), original HHO, MASSFOA, IPSO and PSO, respectively. Moreover, DDHHO exhibits faster convergence speed and higher stability in joint latency-energy optimization. These results verify the effectiveness and

① 基金项目: 国家自然科学基金 (62271264); 浙江省“尖兵领雁+X”重大科技计划 (2025C02033); 浙江理工大学科研启动基金 (25222238-Y)
收稿时间: 2025-09-08; 修改时间: 2025-10-10; 采用时间: 2025-10-20; csa 在线出版时间: 2026-01-15

superiority of DDHHO, providing an efficient, stable, and scalable optimization solution for resource management in IoV-MEC systems.

Key words: mobile edge computing (MEC); Internet of Vehicles (IoV); task offloading; Harris hawks optimization algorithm; dynamic dual-population co-evolution

近年来, 随着车联网 (Internet of Vehicles, IoV) 与移动边缘计算 (mobile edge computing, MEC) 技术的深度融合与快速发展, 边缘计算卸载策略已成为提升车联网系统性能、优化资源配置的关键手段[1−4]. 在典型车联网场景中, 车辆通过 V2V (vehicle-to-vehicle) 和 V2I (vehicle-to-infrastructure) 通信, 与邻近车辆或路侧单元 (road side unit, RSU) 交互, 实现任务的分布式卸载与协同处理. 车辆在行驶过程中会持续产生大量实时数据, 如环境感知信息、路径规划计算、视频识别任务及多媒体数据流等. 如何在有限的通信带宽与计算资源下对这些任务进行高效处理, 是实现智能驾驶、交通疏导与车路协同等应用的基础. 边缘计算卸载通过将部分计算任务分发至邻近的边缘节点 (如 RSU、MEC 服务器或闲置车辆) 执行, 能够显著降低车载设备的计算负载, 减少数据传输时延与能耗, 从而提升整体服务质量与系统效能[5,6].

传统的车联网 MEC 卸载研究主要聚焦于降低任务执行时延与车辆能耗, 通常通过优化计算资源分配、信道调度或任务分片策略实现性能提升. 然而, 在真实动态场景中, 这类方法仍存在 3 方面不足: 其一, 车辆高速移动导致网络拓扑频繁变化, 使得卸载链路易中断; 其二, 车辆计算能力、任务规模与通信条件差异较大, 资源异构性显著; 其三, 时延、能耗与能量成本等多目标优化之间存在冲突[7]. 因此, 如何在高动态、强约束环境下实现多目标协同优化, 成为车联网边缘计算研究的关键挑战.

为应对上述问题, 智能优化算法逐渐被引入 MEC 卸载领域. 其中, 群智能算法因其无需梯度信息、适应性强、易于全局搜索等优点, 成为当前研究热点. 近年来, 粒子群优化 (PSO)、灰狼优化 (GWO)、差分进化 (DE) 等算法已被广泛应用于车联网卸载决策与资源分配中, 但仍存在一定局限. 例如, PSO 在后期易陷入局部最优, GWO 收敛速度较慢, DE 参数依赖性强, 对复杂多峰问题表现不稳定. 哈里斯鹰优化算法 (Harris hawks optimization, HHO) 作为一种新型仿生智能算法, 模拟鹰群的围捕与突袭行为, 具备较强的全局搜索能力与跳跃式寻优特性[8−10]. 然而, 已有研究指出, 原始 HHO 在以下方面存在不足: ① 初始种群多采用均匀随机方式生成, 导致搜索空间覆盖不均; ② 能量模型采用线性衰减形式, 探索-开发阶段切换僵化; ③ 算法缺乏多样性保持机制, 易出现早熟收敛. 针对这些问题, 部分研究提出了改进思路, 如引入混沌初始化、多阶段能量模型或混合变异算子, 但在车联网复杂任务卸载场景下仍存在全局探索不足、收敛稳定性差的问题.

为此, 本文提出一种基于动态双种群协同演化机制 (dynamic dual-population co-evolution, DDPC) 的动态双种群哈里斯鹰优化算法 (dynamic dual-population Harris hawks optimization, DDHHO). 该机制将种群划分为探索群与开发群, 通过并行进化与信息交互保持种群多样性与搜索平衡, 结合 L-T 混沌初始化增强初始分布均衡性, 并引入非线性逃逸能量与非线性跳跃策略作为辅助机制, 实现全局探索与局部开发的自适应动态调节.

本文的主要贡献如下.

(1) 构建融合任务时延与能耗的 IoV-MEC 卸载模型, 涵盖本地计算、V2V 与 V2I 这 3 种模式, 为算法优化提供理论基础.

(2) 提出动态双种群哈里斯鹰优化算法, 引入双种群协同演化机制, 结合 L-T 混沌初始化、非线性能量与跳跃策略, 实现全局探索与局部开发的自适应平衡, 有效缓解 HHO 的早熟收敛问题.

(3) 通过基准函数与 IoV-MEC 仿真验证 DDHHO 的性能, 结果表明其在收敛速度、精度、时延、能耗及综合代价方面均优于 MSHHO、HHO、PSO、IPSO 与 MASSFOA, 表现出更强的鲁棒性与可扩展性.

1 相关工作

车联网与移动边缘计算的深度融合已成为实现低时延与高可靠任务处理的关键技术. 然而, 该领域仍面临车辆高速移动、网络拓扑频繁变化、无线信道不确

定及任务类型多样化等挑战. Wang 等人[11]系统分析了 VEC 卸载方案的分类与挑战, 为后续策略设计提供了参考; Cho 等人[12]构建协同卸载模型, 将任务分片并分布式卸载至沿途多个路侧单元 (RSU), 显著降低了能耗.

为应对 IoV 环境中的高动态性与多目标权衡问题, 智能优化算法被广泛应用于任务卸载决策. 基于多臂老虎机 (multi-armed bandit, MAB) 框架的自适应学习方法无需频繁交换状态信息即可实现高效卸载. Sun 等人[13]的策略在动态拓扑下将时延降低约 30%. Cheng 等人[14]基于粒子群优化 (PSO) 构建卸载机制, 结合时延与能耗模型并引入自适应惯性因子, 有效提升了多车辆场景下的决策效率与稳定性. Shi 等人[15]提出一种基于深度强化学习 (deep reinforcement learning, DRL) 的车辆-边缘-云三层协作卸载算法, 在考虑车辆移动性与动态信道条件下显著降低系统总时延和能耗 (约 25% 下降). Huang 等人[16]进一步针对高动态异构 VEC 环境构建多目标任务卸载优化模型, 提出高效强化学习方法, 在时延、能耗与负载均衡这 3 方面实现更优的收敛稳定性与全局优化性能.

相比深度学习方法在迁移成本和泛化能力方面的限制, 元启发式与仿生算法更适合应对 IoV 场景的不确定性与多目标优化需求. Liu 等人[9]提出结合变异算子的动态 PSO 算法, 显著改善 MEC 网络的延迟与能耗表现. Materwala 等人[17]研究遗传、蚁群与粒子群的混合模型, 验证其在多任务、多目标卸载中的全局搜索与实时响应能力. 刘建华等人[18]构建时延–能耗联合优化模型, 并采用 NSGA-II 进行卸载决策, 在仿真中实现总成本约 30% 的降低与更快的收敛速度.

近年来, 哈里斯鹰优化算法因其仿生捕猎机制在连续优化中表现突出[10]. 在 IoV 场景中, Ali 等人[19]提出了基于 HHO 的 VANET 聚类算法 (HHOCNET), 有效减少簇首比例并降低通信开销. Priya 等人[20]将 HHO 与深度信念网络结合 (DBA-HHO), 在 MEC 异构任务卸载中显著提升时延与能耗性能. 其他研究亦验证了 HHO 在智能建筑边缘卸载中的能耗优化潜力[21]. Padmakala 等人[22]采用对立学习策略增强 HHO 的收敛与跳出局部能力, 在边缘计算任务调度中取得更优性能. Min 等人[23]面向高速移动车联网提出“多任务卸载-资源管理”联合优化框架, 在高动态 V2X 网络中兼顾时延与能效, 整体性能优于传统 PSO 与 GWO.

尽管 HHO 在连续优化问题中表现优异, 但其在 IoV 卸载场景中的研究仍较有限. 原始 HHO 存在种群初始化不均、探索与开发切换缺乏自适应性以及多样性维持不足等问题, 导致在高动态、多目标及跨节点协同环境中性能受限. 针对上述不足, 本文提出动态双种群哈里斯鹰优化算法 (DDHHO). 该算法引入混沌初始化、非线性能量调节与跳跃机制以增强对车辆移动和信道变化的适应性, 并通过动态双种群协同演化机制 (DDPC) 保持种群多样性, 实现全局搜索与局部收敛的自适应平衡.

综上, DDHHO 在 IoV-MEC 场景中不仅突破了原始 HHO 的局限, 还为高动态、高并发和多目标优化的任务卸载提供了一种高效、稳定且可扩展的解决方案.

2 系统模型

2.1 网络模型

本文研究的移动边缘计算 (MEC) 系统部署如图 1 所示. 系统由多辆处于运动状态的车辆、路侧单元 (RSU) 及其附属的 MEC 服务器组成. 车辆在行驶过程中会产生计算任务, 任务可通过 3 种方式执行: 本地计算 (local computing): 任务直接由发起车辆的车载计算单元处理; 车-车卸载 (V2V): 任务通过直连或多跳传输方式卸载至邻近的闲置车辆执行; 车-基础设施卸载 (V2I): 任务经无线链路传输至 RSU, 并交由 MEC 服务器计算处理.

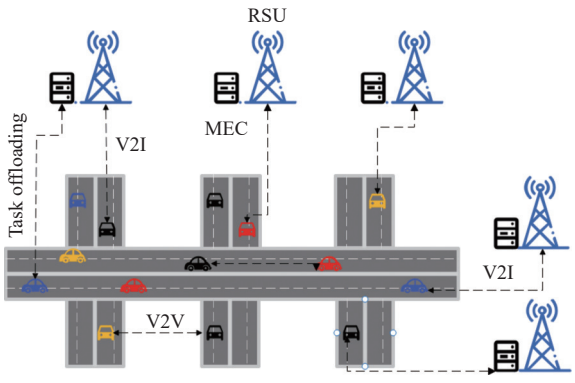


图 1 边缘计算网络示例图

如图 1 所示, MEC 服务器部署在 RSU 节点处, 通过 V2I 链路与任务车辆进行数据交互. V2I 链路适合处理计算资源消耗较大、对计算能力要求高的任务, 但由于车辆与 RSU 的相对位置变化较快, 通信链路存

在较高时延与能耗. V2V 链路用于车辆间直接传输任务数据, 将任务分配给附近具有闲置算力的车辆执行, 该方式通信延迟较低, 但受限于邻车的计算能力. 假设基站覆盖范围内, 任务车辆集合记为$V=\{v_1,v_2,\cdots,v_n\}$, 每辆任务车都有各自的一个任务, 任务定义为 T, 周围可提供计算资源的闲置车辆集合记为$C=\{c_1,c_2,\cdots,c_s\}$. 车辆与 MEC 服务器或闲置车辆之间的无线信道服从瑞利衰落信道模型. 任务车辆与接收端 (闲置车辆 n 或 MEC 服务器) 之间的传输速率可表示为:

$$R_{v,d}=B_v\log_2\left(1+\frac{P_m h_{v,d}}{\sigma^2}\right) \tag{1}$$

其中, P_m表示任务车辆的发射功率; $h_{v,d}$表示车辆与接收端之间的信道增益; B_v表示车辆的分配信道带宽; σ^2表示背景噪声功率.

2.2 任务模型

设任务集合为$T=\{t_1,t_2,\cdots,t_n\}$, 任务t_i有如下主要属性定义: Q_i表示任务数据量; ϕ_i表示计算强度; μ_i表示车辆v_i的本地计算能力 (CPU 周期/s); $\tau_i^{\max}$表示任务完成的最大允许时延. 分配给车辆v_i的 MEC 服务器计算能力为μ_i^{MEC}, 闲置车辆c_i的计算能力为μ_i^{V2V}. 假设任务可分割, 且可同时在本地、邻近闲置车辆及 MEC 服务器上并行执行. 设任务t_i的卸载比例向量为:

$$a_i=[a_i^{\mathrm{loc}},a_i^{\mathrm{V2V}},a_i^{\mathrm{MEC}}] \tag{2}$$

其中, a_i^{loc}表示任务在本地执行的比例, a_i^{V2V}表示任务卸载至邻车的比例, a_i^{MEC}表示任务卸载至 MEC 服务器的比例, 并且$a_i^{\mathrm{loc}}+a_i^{\mathrm{V2V}}+a_i^{\mathrm{MEC}}=1$. 当某一比例为 0 表示该任务不在对应节点执行, 比例为 1 表示完全在该节点执行.

2.3 计算时延与能耗模型

(1) 本地计算

任务t_i在车辆本地执行的数据量为$a_i^{\mathrm{loc}}\times Q_i$, 其计算时延为:

$$T_i^{\mathrm{loc}}=\frac{a_i^{\mathrm{loc}}\times Q_i\times\phi_i}{\mu_i} \tag{3}$$

对应能耗为:

$$E_i^{\mathrm{loc}}=P_i^{\mathrm{loc}}T_i^{\mathrm{loc}} \tag{4}$$

其中, P_i^{loc}为车辆本地计算功率.

(2) 闲置车辆计算 (V2V)

任务t_i卸载至闲置车辆c_i的数据量为$a_i^{\mathrm{V2V}}\times Q_i$, 其计算时延包括传输时延、计算时延:

$$T_i^{\mathrm{loc}}=\frac{a_i^{\mathrm{V2V}}Q_i}{R_{v,c}}+\frac{a_i^{\mathrm{V2V}}Q_i\phi_i}{\mu_i^{\mathrm{V2V}}} \tag{5}$$

$$E_i^{\mathrm{V2V}}=P_v^{\mathrm{tx}}\frac{a_i^{\mathrm{V2V}}Q_i}{R_{v,c}}+P_c^{\mathrm{V2V}}\frac{a_i^{\mathrm{V2V}}Q_i\phi_i}{\mu_i^{\mathrm{V2V}}} \tag{6}$$

其中, P_v^{tx}为车辆传输功率, P_c^{V2V}为闲置车辆计算功率.

(3) MEC 服务器计算 (V2I)

任务t_i卸载至闲置车辆c_i的数据量为$a_i^{\mathrm{MEC}}\times Q_i$, 其总时延为:

$$T_i^{\mathrm{MEC}}=\frac{a_i^{\mathrm{MEC}}Q_i}{R_{v,\mathrm{MEC}}}+\frac{a_i^{\mathrm{MEC}}Q_i\phi_i}{\mu_i^{\mathrm{MEC}}} \tag{7}$$

能耗为:

$$E_i^{\mathrm{MEC}}=P_v^{\mathrm{tx}}\frac{a_i^{\mathrm{MEC}}Q_i}{R_{v,\mathrm{MEC}}}+P_{\mathrm{MEC}}\frac{a_i^{\mathrm{MEC}}Q_i\phi_i}{\mu_i^{\mathrm{MEC}}} \tag{8}$$

其中, P_{MEC}为边缘服务器的计算功率.

2.4 卸载优化模型

卸载优化模型综合考虑任务时延与能耗, 以实现 IoV-MEC 环境下的系统效用最小化.

设任务 i 的总时延与总能耗分别为:

$$T_{\mathrm{total},i}=T_i^{\mathrm{loc}}+T_i^{\mathrm{V2V}}+T_i^{\mathrm{MEC}} \tag{9}$$

$$E_{\mathrm{total},i}=E_i^{\mathrm{loc}}+E_i^{\mathrm{V2V}}+E_i^{\mathrm{MEC}} \tag{10}$$

系统效用函数定义为:

$$U_i=\mu T_{\mathrm{total},i}+(1-\mu)E_{\mathrm{total},i} \tag{11}$$

其中, $\mu\in[0,1]$为时延与能耗的权重系数. 优化目标为:

$$\begin{cases} U_{\min}=\min\dfrac{1}{n}\left(\displaystyle\sum_{i=1}^{n}U_i\right) \\ \text{s.t.}\begin{cases} 0\leqslant a_i^{\mathrm{loc}},a_i^{\mathrm{V2V}},a_i^{\mathrm{MEC}}\leqslant 1,\forall i\in[1,n] \\ a_i^{\mathrm{loc}}+a_i^{\mathrm{V2V}}+a_i^{\mathrm{MEC}}=1,\forall i\in[1,n] \\ T_i^{\mathrm{loc}}+T_i^{\mathrm{V2V}}+T_i^{\mathrm{MEC}}\leqslant\tau_i^{\max},\forall i\in[1,n] \end{cases} \end{cases} \tag{12}$$

3 改进哈里斯鹰优化算法

3.1 原始哈里斯鹰优化算法

哈里斯鹰优化算法 (HHO) 是一种源于哈里斯鹰群体协作捕食行为的仿生智能优化方法, 其流程如图 2 所示. 算法运行过程可概括为以下几个阶段.

(1) 初始化阶段

设定问题维度、解空间上下界、最大迭代次数、适应度函数以及哈里斯鹰种群规模. 通过随机方

式生成初始种群个体的位置向量, 并确保其满足解空间约束.

(2) 初始适应度计算

基于预设的适应度函数, 对初始种群中每个个体进行适应度评估, 并将适应度值最优的个体位置记录为猎物位置.

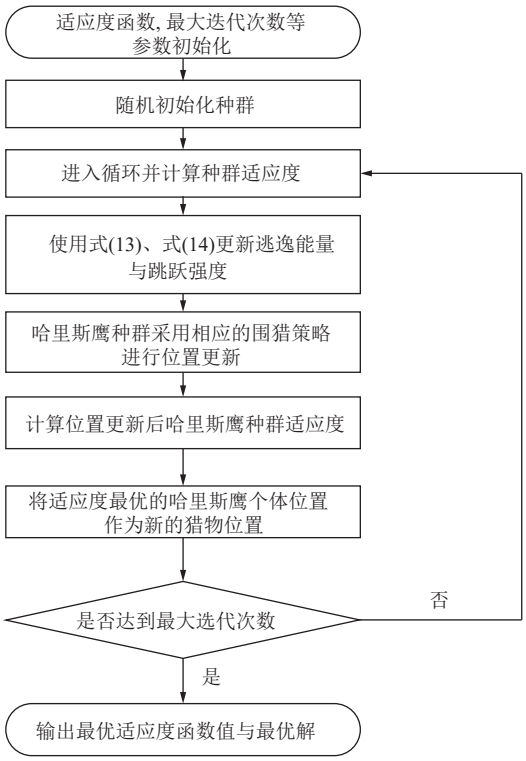


图 2 HHO 流程图

(3) 位置更新阶段

在迭代过程中, 根据当前猎物的逃逸能量与跳跃强度动态选择合适的捕食策略, 对鹰群个体的位置进行更新. 猎物逃逸能量 E 与跳跃强度 J 的计算公式分别如式 (13) 和式 (14) 所示:

$$E = 2E_0\left(1 - \frac{t}{T_{\max}}\right) \tag{13}$$

$$J \sim U(0,2) \tag{14}$$

其中, E 表示逃逸能量因子, $T_{\max}$ 为最大迭代次数, E_0 为从区间 $U(-1, 1)$ 中均匀采样得到的初始随机能量值, t 表示当前迭代次数; J 表示跳跃强度, 其值由区间 (0, 2) 内的均匀分布随机生成.

(4) 适应度更新

对位置更新后的种群重新计算适应度值, 并将其中最优个体的位置更新为新的猎物位置.

(5) 终止条件判断

重复执行步骤 (3) 和步骤 (4), 直至达到最大迭代次数或满足收敛条件, 最终输出全局最优适应度值及对应的解向量.

HHO 算法的整体运行流程如图 2 所示, 其核心思想是通过模拟哈里斯鹰在不同能量状态下对猎物的追捕策略, 实现全局搜索与局部开发的动态平衡.

3.2 动态双种群哈里斯鹰优化算法

尽管 HHO 在全局寻优方面表现较好, 但在种群初始化、探索与开发的切换以及收敛稳定性方面仍存在不足. 为此, 本文提出了动态双种群哈里斯鹰优化算法, 通过多种机制的协同增强, 有效提升算法的性能. 其主要改进措施如下.

(1) L-T 混沌映射策略

群智能优化算法在 IoV 卸载中的应用已被广泛验证, 其在提升时延与能效方面表现出显著优势. 然而, 传统 HHO 算法依然存在若干不足, 亟需进一步改进. 具体而言, 初始种群的分布特性对后续全局搜索与局部开发的平衡具有决定性作用. 传统 HHO 多采用均匀随机方式生成初始个体位置, 虽实现简便, 但往往导致种群在搜索空间中的分布不均, 从而削弱全局探索能力, 降低收敛速度, 并易陷入局部最优.

为此, 本文引入 L-T 混沌映射策略 (logistic-tent chaos mapping) 对初始种群进行优化分布. 该方法结合了 logistic 映射的强混沌性与 tent 映射的均匀遍历性, 通过先执行 logistic 映射产生初始混沌值, 再经 tent 映射进行二次变换, 使得生成的混沌序列在区间[0, 1]内分布更为均匀且保持较强的随机性, 从而显著提升初始种群多样性和全局探索能力. L-T 混沌映射的核心公式如下:

$$x_{n+1} = \begin{cases} \left[r x_n (1 - x_n) + \dfrac{4 - r}{2} \right] \bmod 1, & x_n < 0.5 \\ \left[r x_n (1 - x_n) + \dfrac{(4 - r)(1 - x_n)}{2} \right] \bmod 1, & x_n \geqslant 0.5 \end{cases} \tag{15}$$

其中, x_n 为第 n 次迭代后的值, r 为控制参数. 图 3 展示了 logistic 映射、tent 映射与 L-T 混合映射在 500 次迭代下生成的混沌序列分布情况, 设置初始值 x_0=0.3, logistic 控制参数 r=3.9, tent 变换阈值为 0.5, 从图 3 中可见, L-T 混合映射生成的序列在全局范围内分布更加均衡, 无明显的集中区域, 能够有效改善初始种群的全局探索性能.

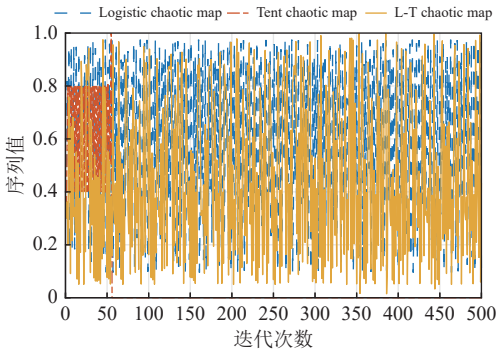


图 3　3 种映射的混沌序列分布对比图

(2) 非线性逃逸能量

在 HHO 算法中, 猎物的逃逸能量 E 是控制算法从全局探索阶段向局部开发阶段过渡的关键参数. 然而, 原始 HHO 算法采用线性衰减策略, 使得 E 值从 2 均匀递减至 0. 这种线性变化模式过于僵化, 在实际优化过程中会导致以下问题: 前期探索不足, 难以充分覆盖搜索空间; 后期开发转入过快, 容易错失全局最优解附近的潜在优质解. 为克服这些不足, 本文设计了一种非线性逃逸能量衰减模型, 其公式如式 (16):

$$E_1 = 2\left(1-\left(\frac{t}{T_{\max}}\right)^k\right).\frac{1+\lambda\sin(2\pi t/T_{\max})}{2} \tag{16}$$

$$E = E_0 E_1 \tag{17}$$

其中, E 为新的能量衰减因子, $T_{\max}$ 为最大迭代次数, t 为当前迭代次数, E_0 为初始随机能量值, λ 为调节系数, k>1 控制衰减速率.

图 4 展示了原始线性逃逸能量与改进非线性逃逸能量随迭代次数变化的对比曲线. 为了直观体现两种曲线的差异, 绘图时固定 E_0=1; 而在后续数值实验中, E_0 仍按 $E_0 \sim U(-1,1)$ 随机生成, 以确保算法的一致性与鲁棒性. 由图 4 可见, 非线性能量曲线在迭代初期衰减更快, 使算法能够迅速逼近潜在优质解域; 而在迭代后期, 其衰减趋势趋于平缓, 从而有效延长了局部开发阶段的持续时间. 这种“前快后缓”的衰减机制在全局探索与局部开发之间实现了更合理的平衡, 显著提升了算法在复杂优化问题中的寻优效率与收敛稳定性.

(3) 非线性跳跃策略

在 HHO 算法中, 猎物的跳跃强度是 0–2 之间的随机数, 导致算法在寻优过程中很难具备较好光谱学与光谱分析的收敛性, 使得算法收敛精度低. 考虑到在迭代后期由于能量衰减, 导致跳跃强度受到较大的影响, 提出了如式 (18) 所示的跳跃策略.

$$J = 2(1-rand)\mathrm{e}^{-\theta\frac{t}{T_{\max}}} \tag{18}$$

其中, $rand$ 表示随机数, 通常从区间[0, 1]内均匀采样, 用来增加随机性, θ 表示衰减系数.

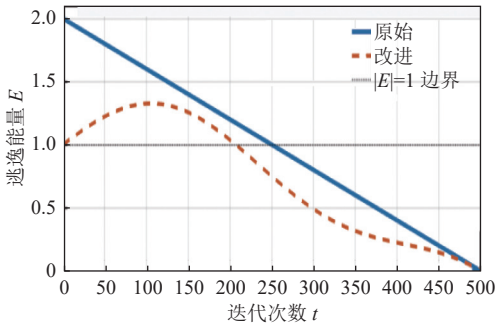


图 4　逃逸能量对比图

设置参数最大迭代次数 T=500, 衰减系数 θ=3, 如图 5(a) 所示, 衰减型跳跃策略下, 跳跃强度 J 在前期保持较大值并伴随随机波动, 有利于全局探索; 随着迭代次数增加, 指数衰减因子逐步减小跳跃幅度, 使算法在后期集中于局部搜索, 从而提升收敛稳定性与精度. 相比之下, 图 5(b) 的原始策略在整个过程中均呈现大幅随机波动, 缺乏动态调节, 虽能在早期避免陷入局部最优, 但在后期易破坏已找到的优良解, 导致收敛震荡和稳定性不足. 综合来看, 衰减型跳跃策略实现了从强探索到稳定开发的平滑过渡, 在保持多样性的同时显著提升了后期收敛性能.

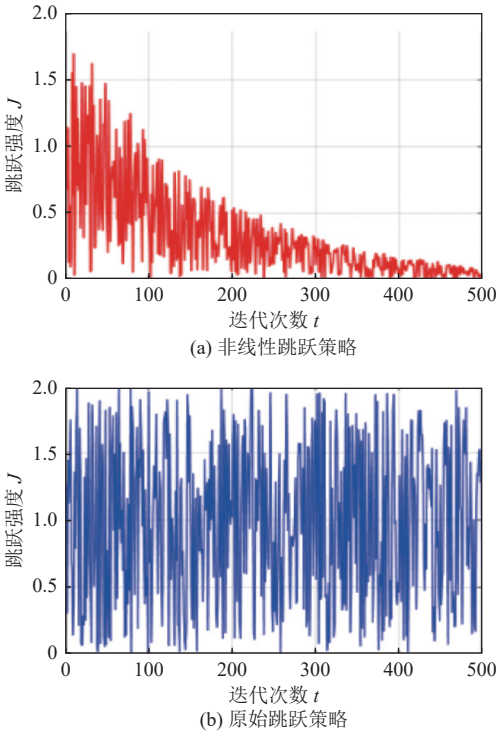


图 5　跳跃强度对比图

(4) 动态双种群协同演化机制

为进一步提升算法的全局搜索能力与局部开发精度, 克服单一种群易陷入局部最优的问题, 本文引入动态双种群协同演化机制 (dynamic dual-population co-evolution, DDPC), 从全局策略层面对算法搜索过程进行调控. 该机制将整个哈里斯鹰种群划分为两个功能差异化的子种群: 探索种群 (exploration group) 与开发种群 (exploitation group). 其中, 探索种群采用更激进的跳跃策略和较大的更新步长, 专注于搜索空间的全局遍历; 而开发种群则以猎物当前位置为中心, 利用更保守的步长进行局部精细搜索. 两类子种群在每轮迭代中分别独立更新位置, 并基于当前适应度动态调整各自规模与资源分配.

探索种群位置更新公式如下:

$$X_i^{t+1} = X_i^t + \alpha \cdot Levy(\lambda) \tag{19}$$

其中, α为步长系数取值为 0.5–2.0 之间, $Levy(\lambda)$表示利用$Levy$飞行分布生成的随机跳跃向量, 用于增强搜索范围.

对于开发种群中的个体, 其位置更新为:

$$X_j^{t+1} = X_j^t + \beta \cdot \left(X_{\text{prey}}^t - X_j^t\right) \tag{20}$$

其中, $\beta \in (0, 1)$ 为局部调整系数, X_{prey}^t表示当前猎物最优位置, 该公式确保开发种群在优解周围进行精细搜索.

设第 t 代中探索子种群与开发子种群的最优适应度分别为$f_{\text{best}}^{\text{explore}}$和$f_{\text{best}}^{\text{exploit}}$. 在每轮迭代过程中, 根据二者的相对适应度表现, 动态调整两个子种群的规模, 其更新方式如下:

$$\gamma = \frac{f_{\text{best}}^{\text{exploit}}}{f_{\text{best}}^{\text{exploit}} + f_{\text{best}}^{\text{explore}}} \tag{21}$$

$$\begin{cases} N_{\text{explore}}^{t+1} = (1+\gamma) \cdot N \\ N_{\text{exploit}}^{t+1} = \gamma \cdot N \end{cases} \tag{22}$$

其中, N 为总种群规模, γ 用于衡量当前搜索更偏向开发还是探索阶段.

每轮迭代结束后, 分别从探索子种群与开发子种群中选取适应度最优的个体, 记为$X_{\text{best}}^{\text{explore}}$和$X_{\text{best}}^{\text{exploit}}$. 在二者中取适应度更优者作为候选, 并用其更新猎物的全局位置:

$$X_{\text{prey}}^{t+1} = \text{argmin}\left(f(X_{\text{best}}^{\text{exploit}}), f(X_{\text{best}}^{\text{explore}})\right) \tag{23}$$

该机制保证了信息流动与方向引导作用, 同时避免某一子种群的过度支配.

DDHHO 的整体执行流程如图 6 所示.

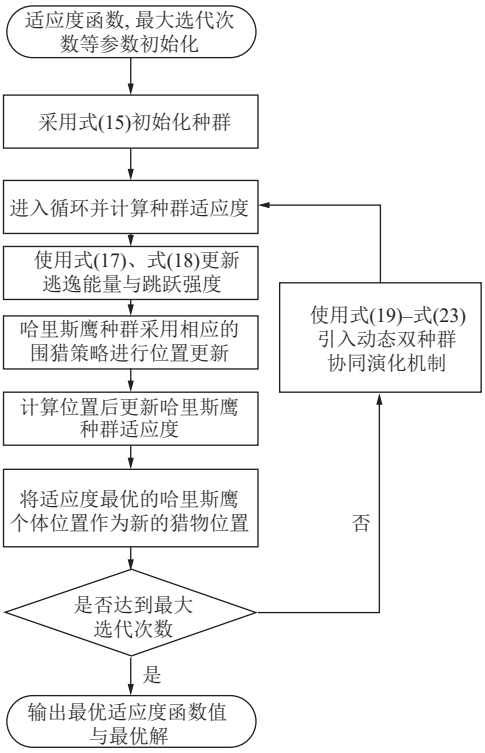


图 6 DDHHO 流程图

基于 DDHHO 的车联网任务卸载决策的伪代码如算法 1.

算法 1. 基于 DDHHO 的车联网任务卸载决策

输入: 任务集信息 T, 种群规模 N, 最大迭代次数$T_{\max}$, 权重系数μ, 闲置车辆数 s, 跳跃强度衰减系数 β 等.

输出: x_{prey}.

1) 系统信息读取与预处理, 采集/读取任务规模、计算强度、本地算力、V2V/V2I 信道、服务器算力等环境参数; 构建时延/能耗计算所需的所有静态/动态量

2) 初始化 (基于 L-T 混沌映射), 生成长度为 $3n$ 的混沌序列并映射到 [0, 1]; 组装得到 N 个初始解$\{x_j\}_{j=1}^n$, 使用式 (2) 对每个任务执行可行性修复, 保证非负且和为 1, 计算每个个体的$f(x_j)$, 记录当前最优作为“猎物”位置x_{prey}

3) while $t<T_{\max}$ do

4) 使用式 (17) 和式 (18) 对非线性逃逸能量与跳跃强度更新, 依据 E 自适应调度“探索开发”强度 (前快后稳)

5) 动态双种群协同演化 (DDPC) 分配, 依据上一代探索/开发子群最优适应度, 使用式 (22) 更新两子群规模N_{explore}^{t+1}和N_{exploit}^{t+1}, 选择适应度较优个体作为精英集, 保留x_{prey}

6) 使用式 (19) 和式 (20) 分别对探索种群位置和开发种群位置进行更新

7) 适应度评估与全局最优更新, 对所有个体计算时延和能耗, 结合时延和能耗权重系数, 选取最优个体x_{gbest}, 若$f(x_{\text{gbest}})<f(x_{\text{prey}})$, 对$x_{\text{prey}}$进行更新

8) 依据两子群最优改进幅度, 使用式 (21) 更新N_{explore}^{t+1}和N_{exploit}^{t+1}的比例, 维持多样性与收敛精度的动态平衡
9) end while
10) 输出x_{prey}

4 实验与分析

4.1 测试环境和参数

所有实验均在 Matlab R2022a 平台下完成, 运行环境为 Windows 11、Intel Core i7-12700 处理器和 16 GB 内存. 为保证公平性, 在相同条件下对比了本文提出的动态双种群哈里斯鹰优化算法 (DDHHO) 与原始 HHO、改进粒子群优化算法 (IPSO)[24]、改进果蝇优化算法 (MASSFOA)[25]、经典 PSO 及混合策略哈里斯鹰优化算法 (MSHHO)[26]. 各算法参数设置一致, 如表 1 所示.

4.2 基准函数实验与有效性验证

为了从全局搜索能力、收敛精度与稳定性这 3 个方面全面评估所提算法的性能, 本文选取了 6 个经典基准测试函数 (见表 2). 其中, F_1 Sphere、F_2 Schwefel 2.21 与 F_3 Quartic with Noise (ε=1E−5) 为单峰函数, 用于验证算法的收敛精度、抗噪声能力及全局寻优性能; F_4 Griewank 与 F_5 Ackley 为多峰函数, 用于检验算法在复杂地形下跳出局部最优的能力; F_6 Kowalik's Function 为固定维度的复杂非线性函数, 用于测试算法在低维约束优化问题中的鲁棒性. 除 F_6 固定为 4 维外, 其余函数均设为 30 维. 所有算法均在相同参数条件下独立运行 30 次, 评价指标包括最优值 (Best)、平均值 (Aver) 与标准差 (STD), 实验结果如表 3 所示.

表 1 实验参数设置

参数	取值
卸载车辆数	10–50
服务器数量	10–30
闲置车辆数	5–10
任务大小Q_i (MB)	3–7
任务复杂度ϕ_i (cycles/bit)	10–20
车辆设备计算能力μ_i (cycles/bit)	7.5×10^7
边缘服务器计算能力μ_i^{MEC}(cycles/bit)	$5.5\times10^8-8\times10^8$
闲置车辆计算能力μ_i^{V2V} (cycles/bit)	7.2×10^7
信道增益$h_{v,d}$ (Hz)	1.3×10^{-4}
带宽B_v (MHz)	1
时延与能耗的权重系数μ	0.5
最大迭代次数$T_{\max}$	2 500
计算任务最大容忍时延$\tau_i^{\max}$ (ms)	800–1 000
噪声功率σ^2 (w)	5×10^{-14}
种群大小SN	55
边缘服务器设备功率P_{MEC} (kw)	0.3–0.4
车辆传输功率P_v^{tx} (w)	0.2–0.3
衰减系数θ	3
步长系数α	0.5–2
局部调整系数β	0–1
振荡幅度因子λ	0.3–0.5
非线性衰减指数k	2
Logistic控制参数r	3.9

表 2 基准函数参数

Function	Formula	Dim	Range	$f_{\min}$
$F1$	$f_1(x)=\sum_{i=1}^{n}x_i^2$	30	[−100, 100]	0
$F2$	$f_2(x)=\max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	30	[−100, 100]	0
$F3$	$f_3(x)=\sum_{i=1}^{n}ix_i^4+\epsilon\cdot\text{random}[0,1)$	30	[−1.28, 1.28]	0
$F4$	$f_4(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left\{\frac{x_i}{\sqrt{i}}\right\}+1$	30	[−600, 600]	0
$F5$	$f_5(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)+20+\text{e}$	30	[−32, 32]	0
$F6$	$f_6(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_1x_2)}{b_i^2+b_1x_3+x_4}\right]^2$	4	[−5, 5]	0.000 307 5

(1) 数值实验结果分析

由表 3 可见, DDHHO 在 6 个测试函数中均取得最佳或次优性能, 整体表现优于其他算法. 在单峰函数 F_1 和 F_2 上, DDHHO 能稳定逼近理论最优解, 其平均值和标准差显著小于对比算法, 展现出更强的全局寻优能力与收敛精度. 例如, 在 F_1 中, DDHHO 的最优值

为 3.00E−120, 远低于 HHO (1.37E−118) 与 MSHHO (6.50E−119), 说明双种群协同机制在探索初期有效提升了全局搜索效率; 在 F_2 中, DDHHO 的平均值和标准差分别为 4.00E−07 与 1.80E−07, 表现出优异的稳定性.

表 3 基准函数实验结果对比

Function	Indicator	IPSO	MASSFOA	PSO	HHO	MSHHO	DDHHO
F_1	Best	1.44E−109	2.37E−110	1.25E−45	1.37E−118	6.50E−119	**3.00E−120**
	Aver	1.16E−103	3.72E−109	5.41E−40	2.00E−117	8.20E−119	**7.50E−119**
	STD	4.40E−103	1.02E−109	2.36E−39	8.15E−118	7.10E−118	**9.0E−119**
F_2	Best	6.00E−07	4.00E−07	3.00E−03	7.00E−07	5.50E−07	**1.00E−07**
	Aver	1.80E−06	1.20E−06	7.00E−03	3.10E−07	2.50E−07	**4.00E−07**
	STD	7.00E−07	4.50E−07	3.20E−03	4.80E−05	4.20E−05	**1.80E−07**
F_3	Best	9.25E−05	2.26E−04	5.12E−04	6.43E−05	4.80E−05	**3.71E−05**
	Aver	6.0E−04	1.47E−03	1.02E−03	9.45E−05	8.20E−05	**5.73E−05**
	STD	2.8E−04	1.36E−03	7.85E−04	9.48E−05	6.10E−05	**5.75E−06**
F_4	Best	3.00E−09	2.20E−09	1.20E−06	6.00E−11	4.20E−11	**1.50E−12**
	Aver	2.50E−08	2.10E−08	3.10E−06	4.00E−10	1.60E−11	**3.00E−11**
	STD	2.10E−08	1.90E−08	1.80E−06	3.50E−10	2.80E−11	**2.50E−11**
F_5	Best	4.78E−15	4.34E−16	8.24E−14	**3.38E−16**	**3.38E−16**	**3.38E−16**
	Aver	7.17E−15	4.61E−15	1.4E−13	**3.38E−16**	**3.38E−16**	**3.38E−16**
	STD	1.07E−15	2.46E−15	5.5E−14	**5.22E−16**	**5.22E−16**	**5.22E−16**
F_6	Best	3.39E−04	3.46E−04	6.24E−04	3.21E−04	3.15E−04	**3.08E−04**
	Aver	6.10E−03	1.20E−03	1.03E−03	4.83E−04	4.10E−04	**3.95E−04**
	STD	1.40E−02	2.60E−03	8.47E−04	7.20E−04	6.80E−04	**5.50E−05**

在多峰函数 F_3–F_5 中, DDHHO 同样展现出较强的全局搜索能力和跳出局部最优的能力. 以 F_3 为例, DDHHO 的平均值仅为 5.73E−05, 标准差为 5.75E−06, 均优于 HHO 与 MSHHO, 表明非线性跳跃策略与动态双种群进化机制有效改善了算法在复杂多峰地形下的搜索鲁棒性. F_5 中, DDHHO 的结果接近理论最优值 (3.38E−16), 与 MSHHO 和 HHO 表现相当但略优, 说明本文算法在局部开发阶段依然保持较高稳定性.

在低维非线性函数 F_6 上, DDHHO 取得 3.08E−04 的最优值和最小的标准差 (3.95E−04), 性能明显优于其他算法, 表明该算法在小规模非线性优化任务中同样具备良好的鲁棒性与稳定性.

(2) 收敛特性分析

图 7 展示了典型函数 F_3、F_5 与 F_6 的收敛曲线. 从图 7 中可以看出, DDHHO 在早期阶段下降速度最快, 能在较少迭代次数内逼近最优解; 中后期收敛曲线趋于平稳, 表现出良好的收敛稳定性. 相比之下, HHO 与 MSHHO 在中期阶段出现一定程度的震荡或停滞, 而 PSO 与 MASSFOA 均存在明显的早熟收敛现象. MSHHO 虽引入 *Levy* 飞行与非线性能量调节, 但在后期仍存在局部开发不足的问题. 相比之下, DDHHO 的动态双种群协同演化机制 (DDPC) 在探索与开发子群之间实现了自适应信息交换, 使算法在收敛速度与稳定性方面取得了更优平衡.

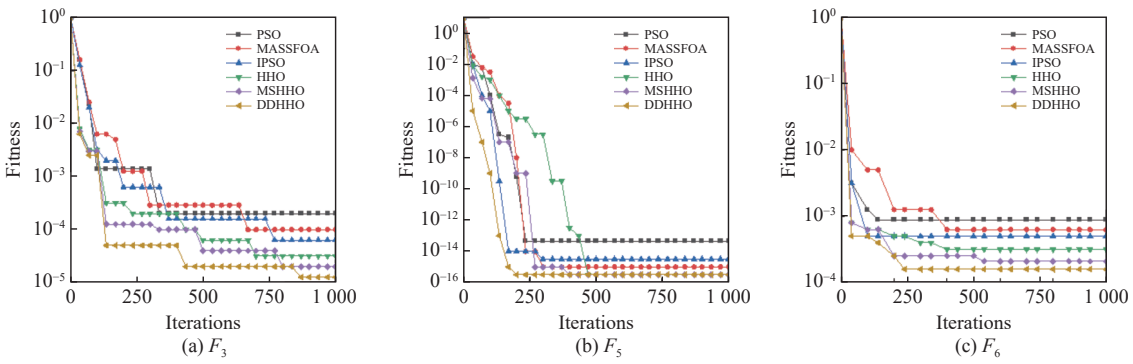


图 7 基准函数收敛对比图

(3) 统计显著性验证

为验证各算法性能差异的统计显著性, 本文采用 Wilcoxon 符号秩检验, 结果如表 4 所示. 从表 4 中可见, DDHHO 与 5 种对比算法在大多数函数上均存在显

著差异 (p<0.05 或 p<0.01), 说明本文算法的性能提升具有统计学意义. 特别是与 MSHHO 的对比中, DDHHO 在 F_1、F_2、F_5 函数上均显著优于 MSHHO (p<0.05), 验证了动态双种群与非线性跳跃策略在提升全局收敛性方面的有效性.

4.3 IoV 任务卸载仿真与性能分析

为全面验证 DDHHO 在车联网 MEC 任务卸载场景中的优化性能, 本文从平均时延、平均能耗与综合代价这 3 个核心指标出发, 将其与 MSHHO、HHO、MASSFOA、IPSO 及 PSO 这 5 种算法进行系统对比. 实验参数参考典型 IoV-MEC 场景及相关文献, 保证任务规模、车辆数量、MEC 服务器配置及通信与计算参数在各实验中保持一致 (见表 1). 在统一条件下, 各算法独立运行, 并记录每次迭代后的关键性能数据, 结果如图 8 所示.

首先, 从平均时延 (图 8(a)) 来看, DDHHO 在初期迭代阶段下降速度最快, 表现出最强的收敛能力. 约在 500 次迭代后, DDHHO 的时延曲线趋于平稳, 最终稳定在约 315 ms, 较 HHO、MSHHO、MASSFOA、IPSO 与 PSO 分别降低约 2.8%、2.3%、4.1%、5.6% 和 7.4%. 其中, MSHHO 通过混合策略改进了原始 HHO 的局部开发能力, 使其在时延性能上略优于 HHO, 但整体仍不及 DDHHO 的动态双种群协同机制所带来的全局收敛速度.

表 4 Wilcoxon 符号秩检验结果表

Function	DDHHO vs. HHO	DDHHO vs. IPSO	DDHHO vs. MASSFOA	DDHHO vs. PSO	DDHHO vs. MSHHO
F_1	2.1E–08 (p<0.01)	4.5E–07 (p<0.01)	6.8E–06 (p<0.01)	3.4E–09 (p<0.01)	5.2E–05 (p<0.01)
F_2	3.6E–09 (p<0.01)	2.1E–08 (p<0.01)	9.7E–07 (p<0.01)	5.6E–08 (p<0.01)	7.5E–04 (p<0.05)
F_3	9.8E–05 (p<0.01)	3.4E–03 (p<0.05)	1.2E–03 (p<0.01)	7.3E–06 (p<0.01)	8.6E–03 (p<0.05)
F_4	1.9E–03 (p<0.05)	7.2E–04 (p<0.01)	2.4E–02 (p<0.05)	6.8E–05 (p<0.01)	3.1E–02 (p<0.05)
F_5	4.7E–05 (p<0.01)	5.3E–04 (p<0.01)	6.1E–03 (p<0.05)	1.5E–06 (p<0.01)	9.2E–04 (p<0.01)
F_6	7.6E–04 (p<0.01)	1.8E–03 (p<0.05)	3.5E–02 (p<0.05)	2.2E–07 (p<0.01)	4.5E–03 (p<0.05)

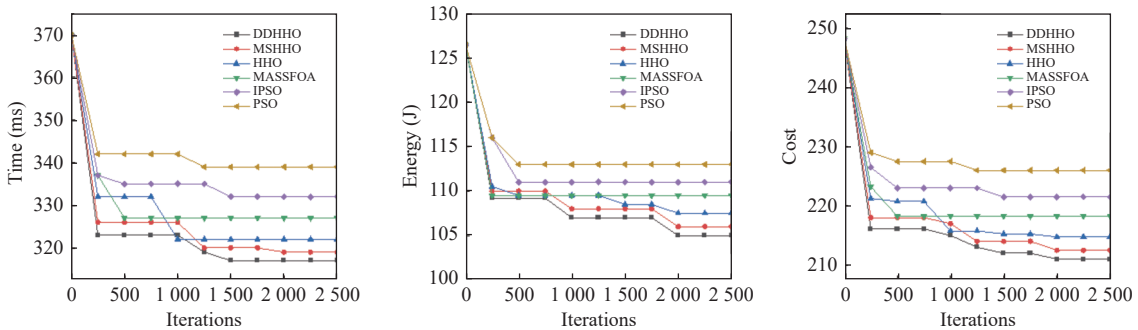


图 8 不同算法在迭代次数上的比较

其次, 在平均能耗 (图 8(b)) 方面, DDHHO 的能耗曲线同样呈现出明显优势, 能耗在前期快速下降后保持平稳, 最终稳定在约 105 J. 相较 HHO、MSHHO、MASSFOA、IPSO 与 PSO, 分别降低约 2.7%、2.2%、5.0%、6.8% 和 8.8%. 可见, DDHHO 通过非线性能量调节与跳跃机制在能耗优化中实现了更强的全局平衡能力; 而 MSHHO 尽管改进了搜索路径的适应性, 但在稳定性方面仍略逊一筹.

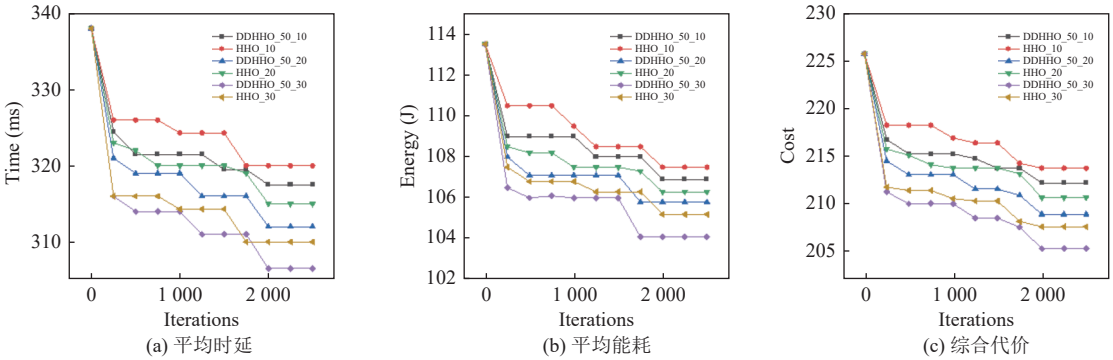
最后, 从综合代价 (图 8(c)) 来看, DDHHO 在多目标权衡上表现最优. 其代价函数在整个迭代过程中下降幅度最大, 并在收敛后保持最低水平, 最终稳定在约 210, 较 HHO、MSHHO、MASSFOA、IPSO 与 PSO 分别降低约 3.0%、2.5%、4.8%、6.0% 和 7.9%. 这表明 DDHHO 在时延与能耗的联合优化中具有更高的适应性与稳定性, 而 MSHHO 相较 HHO 在局部开发与能效平衡上已有显著改进.

为评估 DDHHO 的有效性, 开展实验以对比其与 HHO 在不同移动边缘计算 (MEC) 服务器数量下的收敛情况. 实验中, MEC 服务器数量分别设定为 10 台、20 台和 30 台 .

具体而言, DDHHO_50_10 代表在 50 辆任务车辆、10 台 MEC 服务器配置下 DDHHO 算法的性能表现; DDHHO_50_20 代表 50 辆任务车辆、20 台 MEC 服务器配置下 DDHHO 算法的性能; DDHHO_50_30

则是在 50 辆任务车辆、30 台 MEC 服务器场景下 DDHHO 算法的情况. 同理, HHO_50_10、HHO_50_20、HHO_50_30 分别对应 50 辆任务车辆在 10 台、20 台、30 台 MEC 服务器条件下 HHO 算法的性能表现. 实验中卸载任务量固定为 4 MB, 其余参数设置见表 1, 实验结果呈现于图 9 .

如图 9 所示, 在 MEC 服务器数量分别为 10 台、20 台与 30 台的 3 种配置下, DDHHO 在平均时延、平均能耗及综合代价这 3 项核心性能指标上均表现出优于原始 HHO 的收敛特性与稳定性.



(a) 平均时延 (b) 平均能耗 (c) 综合代价

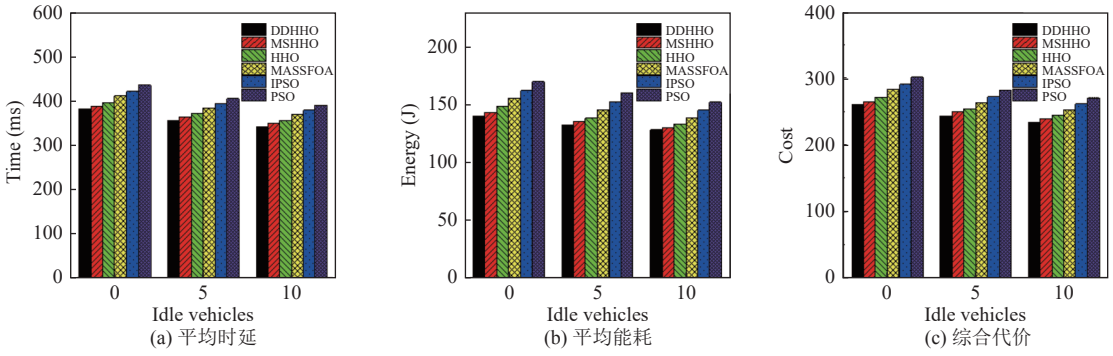
图 9 DDHHO 和 HHO 在不同服务器数下的比较

首先, 从平均时延 (图 9(a)) 来看, DDHHO 在所有服务器配置下均实现了更快速的下降趋势, 且最终收敛值显著低于 HHO. 例如, 在 30 台服务器配置下, DDHHO 最终稳定在约 308 ms, 较 HHO 降低约 2.8%, 体现出其在大规模并行计算资源条件下的时延优化能力. 随着服务器数量增加, DDHHO 的收敛曲线更加平滑, 说明其在多资源场景中能够更高效地调度与分配任务, 减少等待与传输延迟.

其次, 在平均能耗 (图 9(b)) 方面, DDHHO 在迭代初期便迅速拉开与 HHO 的差距, 并全程保持较低的能耗水平. 在 30 台服务器场景中, 能耗稳定值约为 103.8 J, 相较于 HHO 降低 2.9%. 这种能耗优化能力得益于 DDHHO 在任务分配过程中能更精准地选择计算节点, 避免冗余计算与频繁迁移.

最后, 在综合代价 (图 9(c)) 方面, DDHHO 同样表现最优, 且随着服务器数量的增加, 其优势进一步放大. 在 30 台服务器场景中, DDHHO 的最终综合代价约为 204, 明显低于 HHO 的 208, 降幅约为 2%. 这表明 DDHHO 能够在多服务器协同计算条件下, 实现性能与成本的动态均衡优化.

为进一步验证所提算法在引入闲置车辆后的性能表现, 本文设计了不同闲置车辆数量下的对比实验. 实验中分别考虑闲置车辆数量为 0、5、10 这 3 种情形, 并在相同的任务规模与服务器数量条件下, 对 DDHHO、MSHHO、HHO、MASSFOA、IPSO 与 PSO 这 6 种算法进行比较. 实验结果如图 10 所示.



(a) 平均时延 (b) 平均能耗 (c) 综合代价

图 10 不同空闲车辆数量下的实验结果

首先, 从 平均时延 (图 10(a)) 来看, 随着闲置车辆数量的增加, 所有算法的时延均呈下降趋势. 这是由于闲置车辆的参与为任务分配提供了更多的计算资源, 减少了任务车辆与 MEC 服务器之间的长距离传输负担. 其中, DDHHO 始终保持最低的时延水平, 在闲置车辆数为 10 时, 其平均时延稳定在约 340 ms,

较 MSHHO、HHO、MASSFOA、IPSO 和 PSO 分别降低约 3.2%、4.1%、6.5%、8.2% 和 9.7%. 可以看出, MSHHO 相比原始 HHO 在局部开发阶段表现更优, 但整体仍略逊于 DDHHO 的动态双种群协同策略.

其次, 从平均能耗 (图 10(b)) 来看, 随着闲置车辆的增加, 系统整体能耗持续下降. 这是因为部分计算任务被卸载至邻近的闲置车辆执行, 从而显著降低了长距离通信造成的能量消耗. 在所有算法中, DDHHO 的能耗表现最优, 在 10 辆闲置车辆条件下, 其能耗较 MSHHO、HHO、MASSFOA、IPSO 和 PSO 分别降低约 2.9%、3.7%、6.2%、8.5% 和 10.1%. 这表明 DDHHO 在能效优化上具有更高的全局调度能力与资源利用率.

最后, 从综合代价 (图 10(c)) 来看, DDHHO 始终保持最低的代价水平, 并在闲置车辆数量增加时展现出最显著的下降幅度. 在 10 辆闲置车辆配置下, DDHHO 的综合代价约为 242, 较 MSHHO、HHO、MASSFOA、IPSO 和 PSO 的最终收敛值均明显更低, 说明 DDHHO 能够在多目标优化中同时兼顾任务时延与能耗, 实现系统性能与能效的协同提升.

为进一步验证任务规模对卸载性能的影响, 本文设计了任务规模分别为 3 MB、4 MB、5 MB、6 MB 和 7 MB 的实验, 并在其他参数保持一致的条件下, 对 DDHHO、MSHHO、HHO、MASSFOA、IPSO 与 PSO 这 6 种算法进行了对比. 实验结果如图 11 所示.

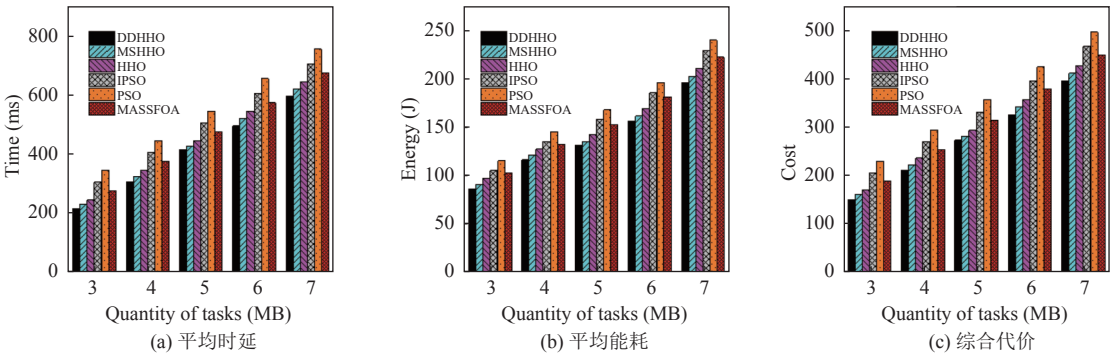


图 11 不同算法在不同任务规模下的比较

首先, 从平均时延 (图 11(a)) 来看, 随着任务规模的增大, 各算法的时延均有所上升, 这是由于计算与传输负载增加所致. 但 DDHHO 在全程保持最低水平, 展现出优异的稳定性与收敛速度. 当任务规模为 7 MB 时, DDHHO 的平均时延较 MSHHO、HHO、MASSFOA、IPSO 和 PSO 分别降低约 3.6%、4.1%、6.3%、8.2% 和 9.5%, 且其增长曲线斜率最小, 说明该算法在高负载场景下能有效抑制延迟积累. MSHHO 在局部开发阶段的改进使其优于 HHO, 但整体仍略逊于 DDHHO 的双种群动态平衡机制.

其次, 从平均能耗 (图 11(b)) 来看, 任务规模增大导致整体能耗上升, 但 DDHHO 始终保持最低能耗水平. 在 7 MB 任务条件下, DDHHO 的能耗较 MSHHO、HHO、MASSFOA、IPSO 和 PSO 分别降低约 3.3%、3.8%、5.6%、7.1% 和 8.7%. 这表明 DDHHO 在能量优化方面具有更强的全局资源调度能力, 而 MSHHO 虽改善了 HHO 的能耗表现, 但在多维能量协调中仍不及 DDHHO 稳定.

最后, 从综合代价 (图 11(c)) 来看, DDHHO 在所有任务规模下均保持最优表现. 当任务规模为 7 MB 时, 其代价值较原始 HHO、MSHHO、MASSFOA、IPSO 和 PSO 的结果分别降低约 4.2%、4.5%、6.9%、8.0% 和 9.5%. DDHHO 通过动态双种群协同演化与非线性能量调节机制, 成功实现了时延与能耗的自适应平衡, 展现出较强的全局优化能力与负载鲁棒性.

为全面验证所提算法在真实 IoV-MEC 场景中的适用性与可扩展性, 本文在 10 台随机分布的 MEC 服务器 环境下开展仿真实验. 车辆规模依次设定为 10、20、30、40 与 50, 其余参数保持一致. 在此条件下, 重点测量系统总时延、总能耗与综合代价这 3 项核心指标. 实验结果如图 12 所示.

首先, 在总时延 (图 12(a)) 方面, 随着车辆数量增加, 所有算法的时延均呈线性上升趋势. 这是由于系统整体通信与计算负载加重所致. 值得注意的是, DDHHO 的时延曲线始终位于最低, 且斜率最小, 表明其具备更强的可扩展性和稳定性. 当车辆数达到 50 辆

时, DDHHO 的总时延较 MSHHO、HHO、MASSFOA、IPSO 和 PSO 分别降低约 3.6%、4.0%、6.1%、7.3% 和 8.9%. MSHHO 在局部开发阶段表现出较优的收敛性能, 相比 HHO 有一定改进, 但仍略逊于 DDHHO 的动态双种群协同搜索策略.

其次, 从总能耗 (图 12(b)) 来看, DDHHO 在各车辆规模下的能耗均保持最低水平, 体现了其优异的能效调度能力. 在 50 辆车辆配置下, DDHHO 的总能耗相较 MSHHO、HHO、MASSFOA、IPSO 与 PSO 分别降低约 3.1%、3.5%、5.0%、6.4% 和 7.9%. 这说明 DDHHO 能在不同负载条件下保持能量分配的动态平衡, 有效降低系统整体能耗.

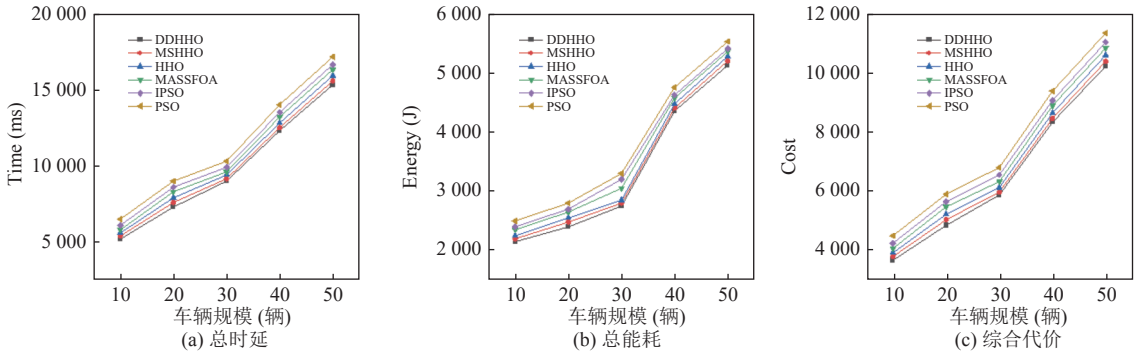


图 12 不同算法在不同车辆规模下的性能对比

最后, 从综合代价 (图 12(c)) 来看, DDHHO 在所有车辆规模下的代价曲线始终最低, 且随车辆数增加时其增长幅度最小, 表现出较强的鲁棒性. 在车辆数为 50 辆时, DDHHO 的综合代价较 MSHHO、HHO、MASSFOA、IPSO 与 PSO 分别降低约 2.5%、3.2%、4.9%、6.0% 和 7.9%. 可见, DDHHO 在多目标优化中实现了性能与成本的双重平衡.

综上所述, MSHHO 相比 HHO 在能耗与时延方面已有明显改进, 而 DDHHO 在全局搜索、自适应调度与多目标优化能力上进一步突破.

5 结论与展望

本文针对车联网移动边缘计算 (MEC) 任务卸载中存在的高动态性、资源受限与多目标优化难题, 提出了一种动态双种群哈里斯鹰优化算法 (DDHHO). 该算法以动态双种群协同演化机制 (DDPC) 为核心, 结合 L-T 混沌初始化、非线性能量调节与非线性跳跃策略, 在保持种群多样性的同时实现全局搜索与局部开发的动态平衡, 有效缓解了原始 HHO 的早熟收敛问题. 实验结果表明, DDHHO 在不同任务规模、MEC 服务器数量及车辆数量下均显著优于 HHO、MASSFOA、IPSO、PSO 和 MSHHO 等算法, 在收敛速度、寻优精度与成本优化方面表现突出, 验证了其在复杂 IoV-MEC 环境中的高效性、鲁棒性与可扩展性.

未来研究可进一步将 DDHHO 与深度强化学习、预测建模或异构资源调度机制结合, 以增强算法的自适应性与在线决策能力; 同时, 可拓展其在车载缓存管理、跨域协同计算与智能调度等场景的应用.

综上, DDHHO 通过创新的双种群演化机制与多重改进策略, 为车联网 MEC 任务卸载提供了一种高效、稳定、可扩展的优化方案, 在智能交通与分布式计算领域具有广阔的应用潜力与推广价值.

参考文献

1 Feng C, Han PC, Zhang X, *et al*. Computation offloading in mobile edge computing networks: A survey. Journal of Network and Computer Applications, 2022, 202: 103366. [doi: 10.1016/j.jnca.2022.103366]

2 Boukerche A, Soto V. Computation offloading and retrieval for vehicular edge computing: Algorithms, models, and classification. ACM Computing Surveys, 2021, 53(4): 80. [doi: 10.1145/3392064]

3 张依琳, 梁玉珠, 尹沐君, 等. 移动边缘计算中计算卸载方案研究综述. 计算机学报, 2021, 44(12): 2406–2430. [doi: 10.11897/SP.J.1016.2021.02406]

4 Du JB, Yu FR, Chu XL, *et al*. Computation offloading and resource allocation in vehicular networks based on dual-side cost minimization. IEEE Transactions on Vehicular Technology, 2019, 68(2): 1079–1092. [doi: 10.1109/TVT.2018.2883156]

5 Zhang SQ, Yi N, Ma Y. A survey of computation offloading with task types. IEEE Transactions on Intelligent

Transportation Systems, 2024, 25(8): 8313–8333. [doi: 10.1109/TITS.2024.3410896]

6 Zhang K, Mao YM, Leng SP, *et al*. Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading. IEEE Vehicular Technology Magazine, 2017, 12(2): 36–44. [doi: 10.1109/MVT.2017.2668838]

7 Zhang J, Guo HZ, Liu JJ, *et al*. Task offloading in vehicular edge computing networks: A load-balancing solution. IEEE Transactions on Vehicular Technology, 2020, 69(2): 2092–2104. [doi: 10.1109/TVT.2019.2959410]

8 You Q, Tang B. Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. Journal of Cloud Computing: Advances, Systems and Applications, 2021, 10(1): 41. [doi: 10.1186/s13677-021-00256-4]

9 Liu YP, Huang W, Wang LP, *et al*. Dynamic computation offloading algorithm based on particle swarm optimization with a mutation operator in multi-access edge computing. Mathematical Biosciences and Engineering, 2021, 18(6): 9163–9189. [doi: 10.3934/mbe.2021452]

10 Heidari AA, Mirjalili S, Faris H, *et al*. Harris hawks optimization: Algorithm and applications. Future Generation Computer Systems, 2019, 97: 849–872. [doi: 10.1016/j.future.2019.02.028]

11 Wang MW, Yi HL, Jiang F, *et al*. Review on offloading of vehicle edge computing. Journal of Artificial Intelligence and Technology, 2022, 2(4): 132–143. [doi: 10.37965/jait.2022.0120]

12 Cho H, Cui Y, Lee J. Energy-efficient cooperative offloading for edge computing-enabled vehicular networks. IEEE Transactions on Wireless Communications, 2022, 21(12): 10709–10723. [doi: 10.1109/TWC.2022.3186590]

13 Sun YX, Guo XY, Song JH, *et al*. Adaptive learning-based task offloading for vehicular edge computing systems. IEEE Transactions on Vehicular Technology, 2019, 68(4): 3061–3074. [doi: 10.1109/TVT.2019.2895593]

14 Cheng J, Guan DJ. Research on task-offloading decision mechanism in mobile edge computing-based Internet of Vehicle. EURASIP Journal on Wireless Communications and Networking, 2021, 2021(1): 101. [doi: 10.1186/s13638-021-01984-6]

15 Shi W, Chen L, Zhu X. Task offloading decision-making algorithm for vehicular edge computing: A deep-reinforcement-learning-based approach. Sensors, 2023, 23(17): 7595. [doi: 10.3390/s23177595]

16 Huang ZD, Wu XF, Dong SB. Multi-objective task offloading for highly dynamic heterogeneous vehicular edge computing: An efficient reinforcement learning approach. Computer Communications, 2024, 225: 27–43. [doi: 10.1016/j.comcom.2024.06.018]

17 Materwala H, Ismail L, Shubair RM, *et al*. Energy-SLA-aware genetic algorithm for edge-cloud integrated computation offloading in vehicular networks. Future Generation Computer Systems, 2022, 135: 205–222. [doi: 10.1016/j.future.2022.04.009]

18 刘建华, 罗荣鑫, 刘佳嘉, 等. 基于 NSGA-II 的车联网边缘计算任务卸载方案. 西安理工大学学报, 2023, 39(4): 557–566. [doi: 10.19322/j.cnki.issn.1006-4710.2023.04.012]

19 Ali A, Aadil F, Khan MF, *et al*. Harris Hawks Optimization-based clustering algorithm for vehicular Ad-Hoc networks. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(6): 5822–5841. [doi: 10.1109/TITS.2023.3257484]

20 Priya JS, Bhagyalakshmi A, Muthulakshmi K, *et al*. DBAHHO: Deep belief network-based adaptive Harris Hawks optimization for adaptive offloading strategy in mobile edge computing. The Journal of Supercomputing, 2022, 78(15): 16745–16769. [doi: 10.1007/s11227-022-04501-8]

21 Yi LZ, Gao XY, Li ZP, *et al*. Task offloading of intelligent building based on CO-HHO algorithm in edge computing. Journal of Electrical Engineering & Technology, 2022, 17(6): 3525–3539. [doi: 10.1007/s42835-022-01108-0]

22 Padmakala S, Khond S, Al-Farouni M, *et al*. Opposition strategy with Harris hawk algorithm for task scheduling in edge computing. Proceedings of the 2024 International Conference on Distributed Systems, Computer Networks and Cybersecurity (ICDSCNC). Bengaluru: IEEE, 2024. 1–5. [doi: 10.1109/ICDSCNC62492.2024.10939606]

23 Min H, Rahmani AM, Ghaderkourehpaz P, *et al*. A joint optimization of resource allocation management and multi-task offloading in high-mobility vehicular multi-access edge computing networks. Ad Hoc Networks, 2025, 166: 103656. [doi: 10.1016/j.adhoc.2024.103656]

24 Du WY, Ma J, Yin WJ. Orderly charging strategy of electric vehicle based on improved PSO algorithm. Energy, 2023, 271: 127088. [doi: 10.1016/j.energy.2023.127088]

25 Yang FQ, Yu SS, Meng C, *et al*. A three-stage optimization of charging scheduling of electric vehicles considering electricity price and user selection. Electrical Engineering, 2024, 106(4): 4737–4746. [doi: 10.1007/s00202-024-02251-9]

26 You GP, Hu YD, Lian C, *et al*. Mixed-strategy Harris hawk optimization algorithm for UAV path planning and engineering applications. Applied Sciences, 2024, 14(22): 10581. [doi: 10.3390/app142210581]

(校对责编: 李慧鑫)