# 边云协作下时延和能耗约束的启发式任务卸载方法

苏命峰[1,2]，王国军[3]，周 聪[1]，王 田[4]

（1. 湖南第一师范学院 计算机学院，长沙 410205；2. 中南大学 计算机学院，长沙 410083；3. 广州大学 计算机科学与网络工程学院，广州 510006；4. 北京师范大学 人工智能与未来网络研究院，广东 珠海 519087）

**摘 要**：为解决移动边缘计算中设备资源受限叠加任务复杂变化，引起负载失衡、任务时延和能耗增大等问题，仿生麻雀共生合作觅食搜索提出一种时延和能耗约束的边云协作计算任务卸载方法。首先，适应移动边云协作，设计飞行者改进发现者更新、正余弦扰动跟随者更新和自适应调整预警者更新，提出一种多策略改进麻雀搜索算法（MSSA）优化任务卸载位置。然后，考虑任务最大完成期限与时延松弛变量，融入超时惩罚能耗，提出一种基于MSSA的启发式任务卸载算法（HTMA），贪心比较不同时延约束下预卸载位置集的总任务时延和总任务能耗，进一步优化任务卸载。仿真实验表明：相比同类算法，本文搜索算法能有效提升寻优精度、收敛速度和鲁棒性，并且本文任务卸载算法适应网络变化的任务平均时延、总任务能耗和节点负载均衡度性能更优。

**关键词**：边缘计算；任务卸载；边云协作；启发式算法；云计算

**中图分类号**：TP301.6 **文献标志码**：A **文章编号**：1671-5497(2025)05-1648-16

**DOI**：10.13229/j.cnki.jdxbgxb.20230849

# A heuristic task offloading approach with delay and energy constraints for edge-cloud collaboration

SU Ming-feng[1,2], WANG Guo-jun[3], ZHOU Cong[1], WANG Tian[4]

(1. *School of Computer Science, Hunan First Normal University, Changsha* 410205, *China*; 2. *School of Computer Science and Engineering, Central South University, Changsha* 410083, *China*; 3. *School of Computer Science and Cyber Engineering, Guangzhou University, Guangzhou* 510006, *China*; 4. *Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai* 519087, *China*)

**Abstract**: To address the problems of load imbalance, task delay, and increased energy consumption caused by limited device resources and complex task variations in mobile edge computing, a computing task offloading approach with delay and energy constraints for edge-cloud collaboration is proposed, inspired by the cooperative foraging search of sparrow populations. Firstly, adapting to mobile edge cloud collaboration, designing the flyer improved producer update, sine-cosine perturbed follower update, and

**收稿日期**：2023-08-11.

**基金项目**：国家自然科学基金项目（62372121）；国家重点研发计划项目（2020YFB1005804）；湖南省自然科学基金项目（2024JJ5105，2023JJ40081）；中南大学中央高校基本科研业务费专项项目（2018zzts180）.

**作者简介**：苏命峰（1980-），男，教授，博士. 研究方向：边缘计算与协同计算. E-mail：mfsu@hnfnu.edu.cn

**通信作者**：王国军（1970-），男，教授，博士. 研究方向：网络与信息安全. E-mail：csgjwang@gzhu.edu.cn

adaptively adjusted alerter update, a multi-strategy improved sparrow search algorithm (MSSA) is proposed to optimize task offloading location. Then, considering the task maximum completion deadline and delay relaxation variables, incorporating the timeout penalty energy consumption, a heuristic task offloading with MSSA algorithm (HTMA) is proposed, which greedily compares the total task delay and total task energy consumption of pre-offloading location sets under different delay constraints to further optimize task offloading. Experimental results show that compared with similar algorithms, MSSA can improve the optimization accuracy, convergence speed, and robustness of location search. Moreover, HTMA adapts to network changes with better performance of average task completion delay, total task energy consumption, and node load balancing degree.

**Key words**: edge computing; task offloading; edge-cloud collaboration; heuristic algorithm; cloud computing

## 0　引　言

随着无线通信技术快速发展,移动智能终端也大量普及应用,涌现出动态内容交互、虚拟/增强现实、模式识别、自动/辅助驾驶等新兴应用。这些新兴应用具有计算密集和时延敏感特点[1],需要大量计算、存储、网络资源。资源受限的移动智能终端无法满足新兴应用需求。为了解决上述问题,移动边缘计算(Mobile edge computing, MEC)在靠近用户区域部署拥有一定计算、存储、网络资源的边缘设备,将用户计算任务(以下简称"任务")就近处理[2]。任务就近处理可以缩短数据传输距离,降低处理时延[3]和执行能耗。Wang等[4]采用强化学习优化MEC微服务部署,以较低成本降低总服务时延。Laskaridis等[5]使用深度神经网络根据节点不同负载生成卸载策略,提高任务执行效率和降低能耗开销。刘伟等[6]采用化学反应优化算法降低任务时延和移动终端能耗。然而,边缘设备的资源有限,当接收任务激增时,容易出现负载过重,导致完成任务的时延增大、能耗增加。同时,任务请求随时空变化,容易出现负载失衡,使一部分边缘设备负载过重引起资源紧缺,而另一部分边缘设备负载过轻出现资源空置,导致系统能效不高。

为了解决这些问题,最新研究考虑边云协作执行任务。针对边云协作的时延优化,Wang等[7]提出了基于边缘智能的分层定价,加快用户、边缘和云的动态博弈以提高用户服务质量。Zhao等[8]评估卸载到边缘或云所需的车辆最佳计算资源,优化任务卸载和计算资源分配以最小化任务时延。针对边云协作的能耗优化,Gupta等[9]提出了能耗最低化、效率最大化策略优化计算卸载,降低应用完成能耗。Jin等[10]优化边缘节点和链路的利用率,将部署问题转化为混合整数线性规划以减少总资源消耗。针对时延和能耗的优化,Zou等[11]使用深度强化学习算法,结合马尔可夫决策优化卸载决策。Ning等[12]将多用户计算卸载转化为混合整数线性规划问题,考虑资源竞争设计了一种启发式资源分配算法提高卸载能效。Chen等[13]利用Lyapunov技术开发了一种在线对等卸载框架,通过对等卸载博弈降低计算时延和能量成本。

边云协作下考虑多重因素优化任务卸载,其卸载难度随任务增加呈现指数级增长变化,具有复杂性、约束性、非线性特点。当前研究存在两个方面的问题。①某些研究在边缘设备资源不足时将接收的部分或全部任务卸载到云中心,没有充分考虑多个边缘设备之间的协作[8,9],使节点负载容易失衡、整体能效降低,任务时延得不到保障,适应网络变化的可扩展性差。②某些研究基于深度学习优化任务卸载决策[11,14],需要协同云和边缘层,网络参数过多,当任务量剧增时算法自身的计算和训练开销过大,容易引起时空维度爆炸,影响任务处理的时效性。为此,本文面向多边-云协作提出一种轻量级的启发式任务卸载方法。在边缘设备考虑时延和能耗约束,利用群智进化优化任务卸载,通过边缘设备和云中心协作执行任务,适应网络变化提高用户服务体验和系统整体能效。本文主要贡献如下:

(1) 提出了一种多策略改进麻雀搜索算法(Multi-strategy improved sparrow search algorithm, MSSA)。通过飞行者动量改进发现者的

任务卸载位置更新,提升预卸载位置集的多样性,提高算法全局搜索能力;融入非线性步长搜索因子和感知负载度的正余弦扰动量子改进跟随者的任务卸载位置更新,提升算法的局部搜索和全局寻优;融合位置搜索态偏离熵和非线性预警系数自适应调整预警者规模,叠加改进预警者的任务卸载位置更新,增强算法跳出局部最优,同时加快算法收敛。

(2)提出了一种基于MSSA的启发式任务卸载算法(Heuristic task offloading with MSSA algorithm, HTMA)。通过综合任务最大完成期限与时间松弛变量的不同时延约束,以融入超时惩罚能耗的总任务能耗为适应度函数,面向多边-云协作基于多策略改进麻雀搜索算法递进寻优不同时延约束下的预卸载位置集,并贪心比较不同位置集下均衡赋权的总任务时延和总任务能耗,进一步优化任务卸载。

(3)搭建了移动边云协作计算仿真实验环境,并验证了本文算法性能。实验结果表明:在MSSA算法提升任务卸载位置的寻优精度、收敛速度和鲁棒性的基础上,HTMA算法在不同任务规模和不同边缘设备下的任务平均时延、总任务能耗、节点负载均衡度均优于5种基准任务卸载算法,能综合优化任务时延和能耗,实现边缘设备负载均衡。

## 1 系统建模与问题分析

### 1.1 移动边云协作计算模型

如图1所示,云中心、边缘设备和移动用户通过有线、无线链路互联互通。云中心即云计算数据中心,拥有充足的硬件资源,可以同时运行大量计算任务。在靠近用户侧的不同地理位置部署若干边缘设备,包括基站、边缘服务器等。边缘设备拥有受限的计算、存储、网络资源,同时运行的任务有限。边缘设备通过有线链路与其他边缘设备和云中心相连。边缘设备接收任务,根据卸载策略将任务在本地执行,或派遣到其他边缘设备、云中心协作执行。移动用户拥有具备一定计算和数据处理能力的终端设备,例如智能手机,能够进行数据预处理或任务前置处理。移动用户根据需要产生若干时延敏感的任务,通过无线链路发送到边缘设备,移动边云协作计算模型为一个五元组模型$(c,S,U,I,Z)$。其中:$c$为云中心;$S$为边缘设备集,由$n(s)$个边缘设备组成,$S=\{s_1,s_2,\cdots,s_{n(s)}\}$。边缘设备拥有5个维度信息,$s=\{f_s,\varphi_s,v_s^r,B_s,R_s\}$。$f_s$为边缘设备的最大算力,量化为设备CPU最大时钟频率。$\varphi_s$和$v_s^r$分别为边缘设备的有线传输功率和任务执行功率系数,与设备CPU硬件架构相关[2,3]。$B_s$和$R_s$分别为边缘设备的最大无线链路带宽和有线链路带宽。边缘设备的角色包括本地边缘设备和协作边缘设备。对用户来说,直接接收任务的边缘设备为本地边缘设备,其余为协作边缘设备。针对不同用户,边缘设备的角色可以互换和叠加。$U$为移动用户集,由$n(u)$个移动用户组成,$U=\{u_1,u_2,\cdots,u_{n(u)}\}$。$I$为任务集,由$n(i)$个任务组成,$I=\{i_1,i_2,\cdots,i_{n(i)}\}$。任务拥有3个维度信息,$i=\{l_i,\chi_i,t_i\}$。$l_i$为任务数据量。$\chi_i$为任务计算量,量化为执行每单位任务数据需要占用的CPU时钟周期[1,3]。$t_i$为任务最大完成期限。$Z$为任务卸载位置集,$Z=\{z_1,z_2,\cdots,z_{n(i)}\}$。任务卸载位置分为3种情况:①卸载到本地边缘设备执行,表示为$z_i^s$;②由本地边缘设备卸载到其他协作边缘设备执行,表示为$z_i^{s,s}$;③由本地边缘设备卸载到云中心执行,表示为$z_i^{s,c}$。本文设定每个任务仅可选择一个卸载位置。

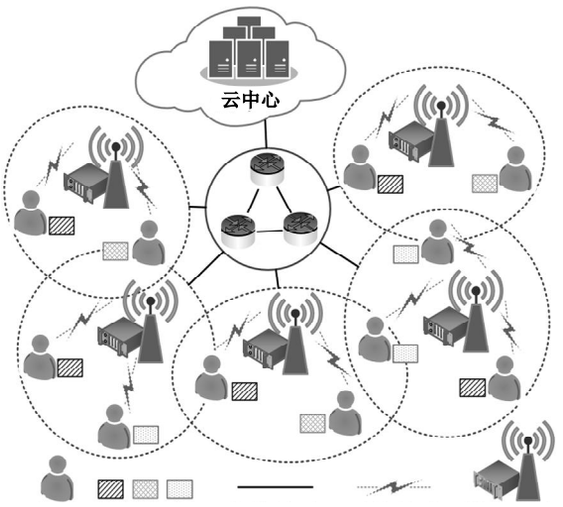



图1 移动边云协作计算模型

Fig. 1 Mobile edge-cloud collaborative computing model

### 1.2 任务时延模型

任务时延包括任务传输时延和任务执行时延。

1.2.1　任务传输时延

边缘设备通过无线链路关联附近的移动用户。移动用户与本地边缘设备之间的上行数据传输速率为$r_{u,s}$,表示为:

$$r_{u,s}=B_{u,s}\log_2\left(1+\frac{\varphi_{u,s}h_{u,s}}{\varepsilon+\sum_{u'\in U}\varphi_{u'}h_{u',s}}\right) \quad (1)$$

式中:$B_{u,s}$为边缘设备分配给移动用户的无线链路带宽;$\varphi_{u,s}$为移动用户与边缘设备之间的无线传输功率;$h_{u,s}$为信道增益系数;$\varepsilon$为背景噪声功率;$\sum_{u'\in U}\varphi_{u'}h_{u',s}$为关联到相同边缘设备的其他移动用户的无线干扰信号功率[11,12],两者合计为噪声功率。边缘设备和云中心通过有线链路接入核心网络。边缘设备的有线链路带宽为$R_s$,存在$r_{u,s}\ll R_s$。

移动用户的任务传输时延包括请求发送时延和结果返回时延。请求发送时延表示为:

$$T_i^t=\begin{cases}\ell_s^i\dfrac{l_i}{r_{u,s}}, z_i\in\{z_i^s\}\\ \dfrac{l_i}{r_{u,s}}+\ell_{s,s}^i\dfrac{l_i}{r_{s,s}^i}+\ell_{s,c}^i\dfrac{l_i}{r_{s,c}^i}, z_i\in\{z_i^{s,s},z_i^{s,c}\}\end{cases} \quad (2)$$

式中:$l_i/r_{u,s}$为任务请求从移动用户发送到本地边缘设备的传输时延;$r_{s,s}^i$和$r_{s,c}^i$分别为任务由本地边缘设备派遣到协作边缘设备和云中心的数据传输速率。由于边缘设备与云中心的传输距离较长,存在$r_{s,s}^i$大于$r_{s,c}^i$。当任务卸载在本地边缘设备执行时$z_i\in\{z_i^s\}$,其$\ell_s^i=1,\ell_s^i\in\{0,1\}$。当任务卸载到其他边缘设备协作执行时$z_i\in\{z_i^{s,s}\}$,其$\ell_{s,s}^i=1,\ell_{s,s}^i\in\{0,1\}$。当任务卸载到云中心协作执行时$z_i\in\{z_i^{s,c}\}$,其$\ell_{s,c}^i=1,\ell_{s,c}^i\in\{0,1\}$。考虑到任务结果相比请求发送的数据量少很多[7,15],并且移动用户的下行速率远大于上行速率,本文不考虑结果返回时延。

1.2.2　任务执行时延

边缘设备接收移动用户的任务,可根据卸载策略将任务在边缘设备本地执行,或者卸载到相邻边缘设备、云中心协作执行。任务执行时延$T_i^r$为:

$$T_i^r=\begin{cases}\dfrac{l_i\chi_i}{f_s^i}, z_i\in\{z_i^s,z_i^{s,s}\}\\ T_i^c, z_i\in\{z_i^{s,c}\}\end{cases} \quad (3)$$

式中:$l_i\chi_i/f_s^i$为任务在本地或相邻边缘设备的执行时延;$f_s^i$为边缘设备分配给任务的算力;$T_i^c$为任务在云中心的执行时延,由于云中心拥有较强计算能力,其值要小于任务在边缘设备的执行时延。

### 1.3　任务能耗模型

任务能耗包括任务传输能耗和任务执行能耗。

1.3.1　任务传输能耗

任务传输能耗与任务传输时延、设备的无线传输功率和有线传输功率相关。表示为:

$$E_i^t=\begin{cases}\ell_s^i\varphi_{u,s}\dfrac{l_i}{r_{u,s}}, z_i\in\{z_i^s\}\\ \varphi_{u,s}\dfrac{l_i}{r_{u,s}}+\ell_{s,s}^i\varphi_s\dfrac{l_i}{r_{s,s}^i}+\ell_{s,c}^i\varphi_s\dfrac{l_i}{r_{s,c}^i}, z_i\in\{z_i^{s,s},z_i^{s,c}\}\end{cases} \quad (4)$$

当任务在本地边缘设备执行时$z_i\in\{z_i^s\}$,其任务传输能耗取决于设备无线传输功率和任务从移动用户发送到本地边缘设备的传输时延,为$\varphi_{u,s}l_i/r_{u,s}$。当任务由本地边缘设备卸载到远端协作执行时$z_i\in\{z_i^{s,s},z_i^{s,c}\}$,在考虑$\varphi_{u,s}l_i/r_{u,s}$的基础上,还需计入本地边缘设备将任务发送至协作边缘设备或云中心的任务传输能耗,分别为$\varphi_sl_i/r_{s,s}^i$和$\varphi_sl_i/r_{s,c}^i$。考虑结果返回的数据量相比请求发送的数据量要少很多,且移动用户的下行速率远大于其上行速率,结果返回至用户的传输时延小[7],本文忽略结果返回的任务传输能耗。

1.3.2　任务执行能耗

任务执行能耗根据任务在边缘设备或云中心执行分别考虑。任务在边缘设备上运行的能耗可由CPU的动态功耗来测算[3,11]。当任务在本地或协作边缘设备上执行时$z_i\in\{z_i^s,z_i^{s,s}\}$,其任务执行能耗与执行完成的任务数据量$l_i$、任务计算量$\chi_i$、边缘设备分配给任务的算力$\chi_i$和边缘设备任务执行功率系数$v_s^r$相关。当任务在云中心执行时$z_i\in\{z_i^{s,c}\}$,其任务执行能耗由云中心任务执行功率$v_c^r$与任务执行时延$T_i^c$共同决定。由此得到任务执行能耗为:

$$E_i^r=\begin{cases}v_s^rl_i\chi_i(f_s^i)^2, z_i\in\{z_i^s,z_i^{s,s}\}\\ v_c^rT_i^c, z_i\in\{z_i^{s,c}\}\end{cases} \quad (5)$$

### 1.4　问题描述

边缘设备需要感知时延和能耗,结合任务负载和资源占用等制定卸载策略,将接收任务决定在本地执行,或者派遣到其他边缘设备、云中心协作执行。由于任务请求随时空变化,边缘设备负载容易失衡。另外,对卸载到边缘设备的任务,根

据式(3)增大分配给任务的边缘设备计算量可以降低任务执行时延,但是根据式(5)会增加任务执行能耗。由此需要考虑任务时延和能耗开销,兼顾边缘设备负载均衡,合理优化任务卸载。

考虑计算任务时延敏感,需要在任务最大完成期限的约束时间内完成任务,以保障用户服务质量。任务完成时延为考虑任务的排队和传输因素,从任务发起到执行完成所用的时间。表示为:

$$T_i^o=\begin{cases}T_i^r+T_{i-1}^o, T_i^t<T_{i-1}^o\\ T_i^t+T_i^r, T_i^t>T_{i-1}^o\end{cases} \tag{6}$$

式中:$T_{i-1}^o$为在节点上执行完前$i-1$个任务的等待时延。特别地,当任务$i$为1时,其$T_{i-1}^o=0$。总任务时延为移动用户所有任务完成时间之和,如式(7)所示。$T^o$越小,用户的应用服务体验越好。

$$T^o=\sum_{\{S\cup c\}}\sum_{\ell_i\in\{\ell_s^i,\ell_{s,s}^i,\ell_{s,c}^i\}}T_i^o \tag{7}$$

为了兼顾服务提供商利益,还需要考虑总任务能耗。总任务能耗为完成所有任务直接产生的能耗,包括任务传输能耗和任务执行能耗,如式(8)所示。$E^o$越低,服务提供商的系统能效越高。

$$E^o=\sum_{\{S\cup c\}}\sum_{\ell_i\in\{\ell_s^i,\ell_{s,s}^i,\ell_{s,c}^i\}}(E_i^r+E_i^t) \tag{8}$$

本文研究在移动边云协作下优化计算任务卸载,兼顾用户和服务提供商利益最小化总任务时延和总任务能耗。问题建模为:

$$\begin{aligned}&\min\zeta=\min(\omega_{\mathrm{T}}\overrightarrow{T^o}+\omega_{\mathrm{E}}\overrightarrow{E^o})\\ &\mathrm{st.}C_1:\sum_{\omega\in\{\omega_{\mathrm{T}},\omega_{\mathrm{E}}\}}\omega=1,\forall\omega_{\mathrm{T}}\in[0,1],\omega_{\mathrm{E}}\in[0,1]\\ &C_2:T_i^o\leqslant t_i,\forall i\in I\\ &C_3:0\leqslant B_{u,s}\leqslant B_s,\forall u\in U,s\in S\\ &C_4:0\leqslant f_s^i\leqslant f_s,\forall i\in I,s\in S\\ &C_5:0\leqslant r_{s,s}^i\leqslant R_s,\forall i\in I,s\in S\\ &C_6:0\leqslant r_{s,c}^i\leqslant R_s,\forall i\in I,s\in S\\ &C_7:\sum_{\ell_i\in\{\ell_s^i,\ell_{s,s}^i,\ell_{s,c}^i\}}\ell_i=1,\forall\{\ell_s^i,\ell_{s,s}^i,\ell_{s,c}^i\}\in\{0,1\},i\in I\end{aligned} \tag{9}$$

式中:$\overrightarrow{T^o}$和$\overrightarrow{E^o}$为$Z$-scores归一化的总任务时延和总任务能耗。$C_1$表示总任务时延权重系数$\omega_{\mathrm{T}}$和总任务能耗权重系数$\omega_{\mathrm{E}}$取值;$C_2$表示任务完成时延限制在任务最大完成期限之内;$C_3$表示移动用户与边缘设备之间的无线链路带宽不超过边缘设备的最大无线链路带宽;$C_4$表示边缘设备分配给任务的计算量不超过边缘设备的最大算力;$C_5$和$C_6$分别表示任务由本地边缘设备派遣到协作边缘设备和云中心的数据传输速率不超过边缘设备的有线链路带宽;$C_7$表示任务仅可在本地边缘设备、协作边缘设备和云中心选择一个卸载位置执行。

## 2 预备知识

麻雀搜索算法(Sparrow search algorithm,SSA)为启发式算法[16],是一种群体智能优化算法。相较于其他群体智能优化算法,SSA具有搜索精度高、收敛速度快、鲁棒性强等特点[17]。SSA受自然界中的生物群体行为启发,仿生麻雀种群觅食活动,将麻雀种群分为共生合作的发现者、跟随者和预警者3类。通过联合3类麻雀种群觅食位置的迭代更新,递进寻优适应度值最佳的食物位置,进而得到表征问题的最优解,借此优化移动边云协作计算的任务卸载位置。

麻雀种群集合为矩阵$\boldsymbol{X}$:

$$\begin{cases}\boldsymbol{X}=[x_1,\cdots,x_{n(j)}]^{\mathrm{T}}\\ x_j=[x_{j,1},x_{j,2},\cdots,x_{j,n(d)}]\end{cases} \tag{10}$$

式中:$n(j)$为麻雀种群数,即预卸载位置集数;$n(d)$为麻雀个体的搜索维数,与任务数$n(i)$一致。针对$n(j)$个麻雀$n(d)$个搜索维数,有$n(j)\times n(d)$个适应度值。

在麻雀觅食过程中,首先选择适应度值占优的麻雀为发现者。发现者能率先取得食物并带领其他麻雀靠近此食物源,其位置更新公式为:

$$x_{j,d}^{\tau+1}=\begin{cases}x_{j,d}^{\tau}\cdot\exp\left(\dfrac{-\tau}{\Delta\cdot\tau_{\max}}\right), r_{\mathrm{rv}}<r_{\mathrm{wv}}\\ x_{j,d}^{\tau}+Q\cdot L, r_{\mathrm{rv}}\geqslant r_{\mathrm{wv}}\end{cases} \tag{11}$$

式中:$x_{j,d}^{\tau+1}$为第$j$个麻雀$d$搜索维数在$\tau$次迭代的觅食位置,类比在预卸载位置集$j$中任务$d$在$\tau$次迭代的卸载位置;$\Delta$为(0,1)区间的随机数;$\tau_{\max}$为最大迭代数;$Q$为在[0,1]区间正态分布的随机数;$\boldsymbol{L}$为全1的$1\times n(d)$矩阵;$r_{\mathrm{rv}}$为在(0,1)区间均匀分布的随机数;$r_{\mathrm{wv}}$为警戒阈值,位于[0.5,1]区间。当$r_{\mathrm{rv}}<r_{\mathrm{wv}}$时,表示在麻雀当前位置没有天敌,将进行广泛搜索以探寻适应度值更优的位置。当$r_{\mathrm{rv}}\geqslant r_{\mathrm{wv}}$时,表示麻雀已经察觉到天敌,将向安全区域靠拢,按正态分布随机移动到当前位置附近。

然后,在麻雀种群中将发现者之外的其他麻

雀作为跟随者。跟随者利用当前发现者的位置信息继续寻找食物,其位置更新公式为:

$$x_{j,d}^{\tau+1}=\begin{cases}x_{\text{best}}^{\tau}+|x_{j,d}^{\tau}-x_{\text{best}}^{\tau}|A^{*}L, j\leqslant n(j)/2\\ Q\cdot\exp\left(\dfrac{x_{\text{worst}}^{\tau}-x_{j,d}^{\tau}}{j^{2}}\right), j>n(j)/2\end{cases}\tag{12}$$

当$j\leqslant n(j)/2$时,跟随者围绕在当前最佳位置的发现者周围觅食,有可能发生食物的争夺,让其成为新的发现者。$x_{\text{best}}^{\tau}$和$x_{\text{worst}}^{\tau}$分别表示截止$\tau$次迭代时适应度值最优和最差的麻雀种群觅食位置。$A^{*}=A^{\text{T}}(\boldsymbol{A}\cdot A^{\text{T}})^{-1}$,$\boldsymbol{A}$表示每个元素值随机为$\{-1,1\}$的$1\times n(d)$矩阵。当$j>n(j)/2$时,表示跟随者处于十分饥饿的状态,需要到别的区域寻找食物。

此外,在当前麻雀种群中随机选择部分麻雀作为预警者进行警戒。当发现危险来临时预警者会放弃当前位置,并快速移动到一个新的位置。预警者有助于跳出算法局部最优,其位置更新公式为:

$$x_{j,d}^{\tau+1}=\begin{cases}x_{\text{best}}^{\tau}+\beta\cdot|x_{j,d}^{\tau}-x_{\text{best}}^{\tau}|, y_{j}>y_{\text{best}}\\ x_{j,d}^{\tau}+k\dfrac{|x_{j,d}^{\tau}-x_{\text{worst}}^{\tau}|}{|y_{j}-y_{\text{worst}}|+\upsilon}, y_{j}=y_{\text{best}}\end{cases}\tag{13}$$

式中:$y_{\text{best}}$和$y_{\text{worst}}$分别为当前麻雀种群最佳和最差觅食位置的适应度值。当$y_{j}>y_{\text{best}}$时,表示麻雀处于种群外围区域,将向最佳位置靠拢。$\beta$为步长调整参数,表示$[1,n(d)]$区间正态分布的随机数。当$y_{j}=y_{\text{best}}$时,表示麻雀处于种群中心位置,将随机行走至自身附近位置。行走方位由$k$确定,$k$为一个$[-1,1]$区间均匀分布的随机数。行走步长与当前位置、最差位置、当前适应度、最差适应度相关。$\upsilon$为最小常数,避免分式分母取值为0。

SSA首先随机初始化麻雀种群位置,从麻雀种群中选取$\mu_{\text{sd}}$个发现者,根据式(11)进行位置更新;然后将没被选取的麻雀作为跟随者,根据式(12)进行位置更新;再从麻雀种群中随机选取$\mu_{\text{sw}}$个麻雀作为预警者,根据式(13)进行位置更新。位置更新的同时调整最优和最差适应度值。达到最大迭代数后输出最佳位置集。SSA算法的时间复杂度为$O(\text{SSA})=O(\tau_{\max}\times(n(j)+\mu_{\text{sw}})\times n(d))$。$n(j)\times n(d)$为发现者和跟随者单次搜索的时间复杂度,$\mu_{\text{sw}}\times n(d)$为预警者单次搜索的时间复杂度。

## 3　任务卸载

移动边云协作任务卸载为NP难问题[3,14]。针对任务卸载难题,传统方法需要遍历整个搜索空间,面对网络规模增长叠加多重资源动态变化,容易产生搜索的“组合爆炸”,难以在短时间内完成搜索。为此,仿生自然界的生物群智进化来解决任务卸载难题。将任务卸载方案类比麻雀种群,将时延和能耗约束的适应度函数作为问题表征,综合3类种群的卸载位置迭代更新,递进优化移动边云协作的任务卸载位置。首先,适应移动边云协作计算任务卸载对麻雀搜索算法进行改进。然后,考虑时延和能耗约束,引入时延松弛变量和超时惩罚能耗等,基于改进麻雀搜索算法进一步优化计算任务卸载。

### 3.1　多策略改进麻雀搜索算法

SSA在全局搜索能力、寻优精度、收敛速度方面存在提升空间[17-19],本文适应移动边云协作计算对SSA改进如下。①用飞行者动量改进发现者的任务卸载位置更新,提升预卸载位置集的多样性,提高算法的全局搜索能力。②引入正余弦策略扰动跟随者位置更新,增强预卸载位置集的多样性,提升算法的局部开发能力和全局寻优精度。③自适应调整预警者规模并优化其位置更新,保障预卸载位置集的多样性,帮助算法跳出局部最优并加快收敛。

#### 3.1.1　飞行者改进发现者更新策略

在任务卸载位置搜索过程中,首先由发现者带领麻雀种群(即预卸载位置集)发现适用度值优的任务卸载位置。在无危险时,发现者需要广泛探索,保证预卸载位置集的多样性,以不断发现适应度值更优的任务卸载位置。但根据式(11),当$r_{\text{rv}}<r_{\text{wv}}$时,发现者的位置搜索范围,即$\exp\left(\dfrac{-\tau}{\Delta\cdot\tau_{\max}}\right)$,随迭代增加而逐渐收窄,如图2(a)所示。这降低了预卸载位置集的多样性,弱化了算法的全局寻优。由此,受鸟群飞行启发,采用飞行者动量改进发现者的任务卸载位置更新,表示为:

$$x_{j,d}^{\tau+1}=\begin{cases}x_{j,d}^{\tau}\cdot(1+\nabla), r_{\text{rv}}<r_{\text{wv}}\\ x_{j,d}^{\tau}+Q\cdot L, r_{\text{rv}}\geqslant r_{\text{wv}}\end{cases}\tag{14}$$

当$r_{\text{rv}}<r_{\text{wv}}$时,采用飞行者动量$(1+\nabla)$影响

发现者的任务卸载位置更新。∇表示标准差为1,均值为0的高斯分布随机数。图2(b)显示改进后发现者的任务卸载位置搜索空间。与图2(a)不同,在整个位置搜索过程中,发现者一直都保持较广的搜索范围,不受迭代增加的影响,从而提升预卸载位置集的多样性,提高算法的全局搜索能力。

3.1.2 正余弦扰动跟随者更新策略

跟随者在3类麻雀种群中占比最大,改进其任务卸载位置更新能有效提升算法的寻优能力。在任务卸载位置搜索过程中,部分跟随者围绕在当前最佳发现者周围搜索位置,其他跟随者基于自身位置漫无目的搜索位置。前者在算法初期弱化预卸载位置集的多样性,使位置搜索容易停滞,增加算法陷入局部最优的可能。后者在算法后期不利于任务卸载位置的局部搜索,影响算法的寻优精度。由此,在跟随者搜索阶段,引入正余弦扰动策略改进跟随者的任务卸载位置更新,表示为:

$$x_{j,d}^{\tau+1}=\begin{cases}\theta_\tau x_{\text{best}}^{\tau}+\rho_\tau\cos(\partial)|x_{j,d}^{\tau}-x_{\text{best}}^{\tau}|A^{*}L, j\leqslant\dfrac{n(j)}{2}\\ x_{j,d}^{\tau}+\rho_\tau\sin(\partial)|x_{j,d}^{\tau}-x_{\text{worst}}^{\tau}|, j>\dfrac{n(j)}{2}\end{cases}$$

$$\theta_\tau=\left(\frac{\exp(\tau/\tau_{\max})-1}{\exp-1}\right)^{2},\ \rho_\tau=\left(1-\left(\frac{\tau}{\tau_{\max}}\right)^{\eta}\right)^{1/\eta} \tag{15}$$

当$j\leqslant n(j)/2$时,采用余弦扰动机制更新任务卸载位置。当$j>n(j)/2$时,采用正弦扰动机制更新任务卸载位置。$\theta_\tau$为非线性步长搜索因子,用于调整任务卸载位置更新的依赖度。在算法初期,$\theta_\tau$值偏小可以降低当前全局最佳位置对任务卸载位置更新的影响,有助于增强预卸载位置集的多样性,提升算法的全局搜索能力。在算法后期,$\theta_\tau$值偏大有助于利用当前全局最佳位置信息微调任务卸载位置,提高算法的局部开发能力。$\rho_\tau$为正余弦扰动量子,用于步长幅度调整。在任务卸载寻优初期,$\rho_\tau$减少幅度趋小,有利于跳出局部最优,增强卸载位置优化的全局搜索。在任务卸载寻优后期,$\rho_\tau$减小幅度趋大,有助于卸载位置的局部寻优细化,提升最优解精度。$\eta$为负载度,$\eta=1+0.05\times n(i)/n(s)$,与任务数和边缘设备数相关。两者比值越大,说明边缘设备负载越重,$\eta$越大,将增大正余弦扰动。$\theta_\tau$和$\rho_\tau$的变化曲线如图3所示,根据不同任务负载列出3种负载度下正余弦扰动量子$\rho_\tau$的变化曲线。$\partial$为正余弦参量,其值为$(-\pi,\pi)$区间的随机数,影响跟随者的行走方位。

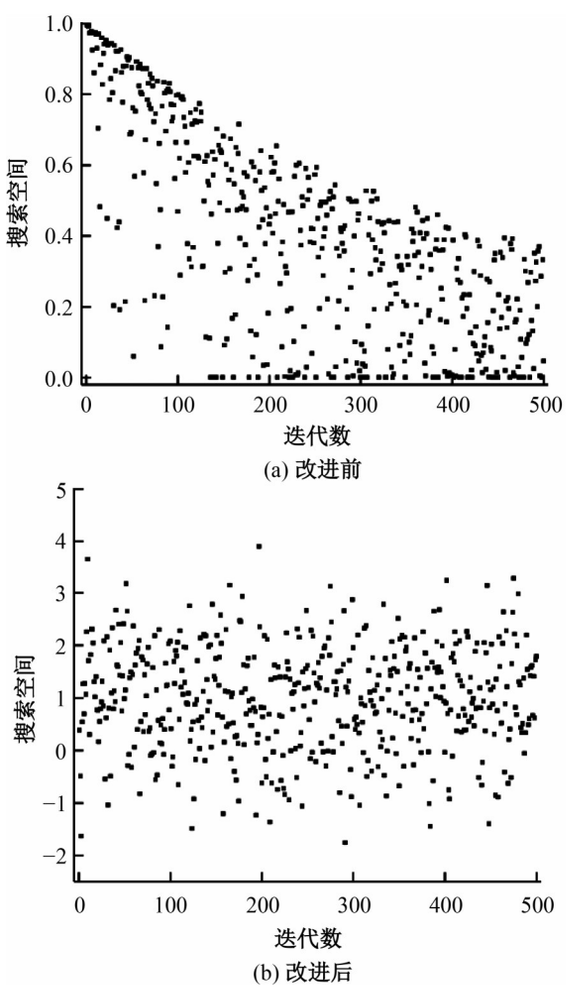



图2 发现者的任务卸载位置搜索空间($r_{\text{rv}}<r_{\text{wv}}$)

**Fig. 2 Producer's task offloading location search space ($r_{\text{rv}}<r_{\text{wv}}$)**

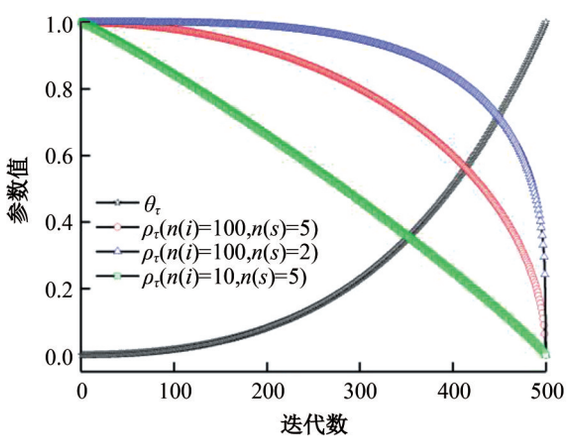



图3 $\theta_\tau$和$\rho_\tau$的变化曲线($\tau_{\max}=500$)

**Fig. 3 Change curve of $\theta_\tau$ and $\rho_\tau$ ($\tau_{\max}=500$)**

3.1.3　自适应调整预警者更新策略

预警者及数量多少直接影响任务卸载位置的全局寻优和搜索速度。一般来说,预警者数量多,可以强化算法跳出局部最优,但是会增加任务卸载位置的搜索时间,影响算法收敛。预警者数量少,可以减少任务卸载位置的搜索时间,但是影响任务卸载优化的全局搜索,弱化算法跳出局部最优。因此,采用自适应调整预警者更新策略动态调整预警者规模。在预警者搜索阶段,融合非线性预警系数和位置搜索态偏离熵得到自适应调整预警者数$\mu_{sw}^{\tau}$,表示为:

$$\mu_{sw}^{\tau}=\mu_{sw}^{max}-\Gamma_{\tau}(\mu_{sw}^{max}-\Phi\mu_{sw}^{min}) \tag{16}$$

$$\Gamma_{\tau}=(2\frac{\tau}{\tau_{max}}-(\frac{\tau}{\tau_{max}}))^2 \tag{17}$$

$$\Phi=\exp|\frac{y_{best}-y_{worst}}{y_{best}+y_{worst}}| \tag{18}$$

式中:$\mu_{sw}^{max}$、$\mu_{sw}^{min}$分别为最大和最小预警者数。合理设置$\mu_{sw}^{max}$和$\mu_{sw}^{min}$值,在任务卸载搜索前期保持较多预警者,保证预卸载位置集的搜索广度。随着搜索递进自适应减小预警者数,加快算法收敛。$\Gamma_{\tau}$为非线性预警系数,与当前和最大的迭代数相关。$\Gamma_{\tau}$值随迭代递进呈现非线性微分递增,使预警者数逐步减少。$\Phi$为位置搜索态偏离熵,$\Phi\in[1,e]$,与当前预卸载位置集的最佳和最差适应度值相关。可以看出,$\Phi$值小,则预警者数大,可提高警戒,增强算法跳出局部最优的能力。反之,$\Phi$值大,则预警者数小,可减少搜索时间,加快算法收敛。

在自适应调整预警者数的同时,改进预警者的任务卸载位置更新,表示为:

$$x_{j,d}^{\tau+1}=\begin{cases}x_{j,d}^{\tau}+\beta(x_{j,d}^{\tau}-x_{best}^{\tau}), y_j>y_{best}\\ x_{j,d}^{\tau}+\beta(x_{worst}^{\tau}-x_{best}^{\tau}), y_j=y_{best}\end{cases} \tag{19}$$

当预警者的任务卸载位置在当前最佳任务卸载位置的外围时$y_j>y_{best}$,将其移步到自己当前位置与最佳位置之间的随机位置。当预警者的任务卸载位置在当前最佳任务卸载位置时$y_j=y_{best}$,将跳跃到当前最差与最佳位置之间的随机位置。改进预警者的任务卸载位置更新相比SSA算法式(13),在保障预卸载位置集的多样性、提升算法跳出局部最优的基础上,简化了预警者的任务卸载位置更新计算。

3.1.4　MSSA算法描述

多策略改进麻雀搜索算法(MSSA)综合飞行者改进发现者更新策略、正余弦扰动跟随者更新策略和自适应调整预警者更新策略改进麻雀搜索算法,如算法1所示。

算法1　MSSA

输入:麻雀种群数n(j),任务数n(i),最大迭代数$\tau_{max}$,发现者数$\mu_{sd}$,警戒阈值$\tau_{wv}$,最大预警者数$\mu_{sw}^{max}$,最小预警者数$\mu_{sw}^{min}$,适应度函数

输出:最佳位置集

1. 初始化任务预卸载位置
2. while $\tau\in\tau_{max}$ do
3. for i=1 to $\mu_{sd}$
4. 运用飞行者改进发现者更新策略,根据式(14)更新发现者的任务卸载位置
5. end for
6. for i=($\mu_{sd}$+1) to n(j)
7. 运用正余弦扰动跟随者更新策略,根据式(15)更新跟随者的任务卸载位置
8. end for
9. 运用自适应调整预警者更新策略,根据式(16)自适应调整预警者数$\mu_{sw}^{\tau}$
10. for i=1 to $\mu_{sw}^{\tau}$
11. 根据式(17)更新预警者的任务卸载位置
12. end for
13. end while
14. return 最佳位置集

第1步根据任务数确定搜索维数,根据边缘设备集和云中心确定搜索空间的边界,随机初始化任务预卸载位置。第2~13步进行$\tau_{max}$次迭代搜索,依次更新发现者(第3~5步)、跟随者(第6~8步)、预警者(第9~12步)3类预卸载位置集的任务卸载位置,位置更新的同时计算适应度值。达到$\tau_{max}$迭代后,第14步返回任务卸载的最佳位置集。MSSA算法的时间复杂度为$O(\text{MSSA})=O((\tau_{max}\times n(j)+n_{sw})\times n(i))$。与$O(\text{SSA})$不同,$n_{sw}=\sum_{\tau=1}^{\tau_{max}}\mu_{sw}^{\tau}$,为$\tau_{max}$次迭代搜索过程中自适应调整的预警者数累加。由于在迭代搜索过程中$\mu_{sw}^{\tau}$逐次递减,存在$n_{sw}$小于$\tau_{max}\times\mu_{sw}$(SSA算法$\tau_{max}$次迭代预警者数累加)。MSSA算法的时间复杂度小于SSA。

### 3.2　基于MSSA的启发式任务卸载算法

为均衡时延和能耗,进一步优化任务卸载,考虑任务最大完成期限,引入时延松弛变量和超时惩罚能耗,提出一种基于MSSA的启发式任务卸载算法(HTMA)。通过设计融合时间和能耗的

适应度函数,运用改进麻雀搜索算法MSSA递进寻优任务卸载位置,通过本地边缘设备、协作边缘设备或云中心协作执行任务,兼顾用户和服务提供商利益取得总任务时延和总任务能耗最小。HTMA考虑边缘计算的分布式特性,在各边缘设备独立进行任务卸载决策,无须云中心和边缘层设备联动,为轻量级任务卸载算法,如算法2所示。

算法2 HTMA

输入:任务集I,边缘设备集S,云中心c,预卸载位置集数n(j),最大迭代数$\tau_{max}$,发现者数$\mu_{sd}$,警戒阈值$\tau_{wv}$,最大预警者数$\mu_{sw}^{max}$,最小预警者数$\mu_{sw}^{min}$,时延松弛变量$\xi$,时延-能耗惩罚算子$\gamma$

输出:任务卸载位置集Z

1. while $t_\xi \in ((t_i-\xi),t_i)$ do

2. 基于式(8),引入时延-能耗惩罚项$E_i^*=\begin{cases}\gamma(T_i^o-t_\xi)^2, T_i^o>t_\xi\\ 0, T_i^o\leqslant t_\xi\end{cases}$,得到融入超时惩罚能耗的总任务能耗$E_*^o=\sum_{\{S\cup c\}}\sum_{i\in\{I_r,I_e\}}(E_i^r+E_i^t+E_i^*)$

3. $Z(t_\xi)=\text{fitness}(E_*^o)/\llbracket \text{MSSA} \rrbracket$

4. end while

5. 根据式(7)计算不同时延约束下预卸载位置集$\mathbb{Z}$的总任务时延$T^o$

6. 根据式(8)计算不同时延约束下预卸载位置集$\mathbb{Z}$的总任务能耗$E^o$

7. $Z\xleftarrow{\min(\omega_T\overline{T^o}+\omega_E\overline{E^o})}\mathbb{Z}=\{Z(t_\xi)|t_\xi=((t_i-\xi),t_i)\}$

8. return 任务卸载位置集Z

在边缘设备运行HTMA算法进行计算任务卸载。第1步基于任务最大完成期限$t_i$,引入时延松弛变量$\xi$,考虑不同时延$t_\xi\in((t_i-\xi),t_i)$约束下的任务预卸载。第2~3步基于MSSA算法,以融入超时惩罚能耗的总任务能耗$E_*^o$为适应度函数,得到时延$t_\xi$约束下的任务预卸载位置$Z(t_\xi)$。当$T_i^o>t_\xi$时,$E_*^o$计入超时惩罚能耗$E_i^*=\gamma(T_i^o-t_\xi)^2$。$\gamma$为时延-能耗惩罚算子。第4步得到不同时延约束下预卸载位置集$\mathbb{Z}=\{Z(t_\xi)|t_\xi=((t_i-\xi),t_i)\}$。第5步、第6步根据式(7)和式(8)分别计算预卸载位置集$\mathbb{Z}$的总任务时延$T^o$和总任务能耗$E^o$。第7步根据式(9)贪心比较经过Z-scores归一化和均衡赋权的预卸载位置集$\mathbb{Z}$的$T^o$和$E^o$,得到最优任务卸载位置集$Z$。第8步返回任务卸载位置集$Z$。HTMA算法的时间复杂度为$O(\text{HTMA})=O(\xi\times(\tau_{max}\times n(j)+n_{sw})\times n(i))$。

## 4 仿真分析

移动边云协作计算实验采用Matlab软件仿真,依托硬件为一台4×3.5 GHz CPU、64 GB RAM、3×2 TB SATA的x64工作站。在仿真环境部署云中心和若干边缘设备为移动用户提供计算服务[11,20]。在实验中模拟网络变化动态改变不同任务数、边缘设备数等参数,主要参数及变化范围如表1所示。

**表1 实验环境参数**

**Table 1 Parameters of experimental environment**

| 参数 | 单位 | 值范围 |
|---|---|---|
| 任务数据量 | kB | [600,1 000] |
| 任务计算量 | cycle•kbit$^{-1}$ | [20,100] |
| 任务最大完成期限 | ms | 50 |
| 无线传输功率 | W | [2,5] |
| 边缘设备的最大无线链路带宽 | Mbit/s | 100 |
| 信道增益系数[11] | | $8.2^{-8}$~$1.5^{-6}$ |
| 噪声功率 | W | [1,2]×$10^{-8}$ |
| 边缘设备的最大算力 | GHz | 2×[2,4] |
| 边缘设备的有线链路带宽 | Mbit/s | 1 000 |
| 边缘设备的有线传输功率 | W | [1,3] |
| 边缘设备任务执行功率系数 | W•Hz$^{-3}$ | [1.2,1.5]×$10^{-26}$ |
| 云中心任务执行功率 | W | [2.1,2.5]×$10^{-25}$ |

### 4.1 多策略改进麻雀搜索算法性能分析

为了评估分析MSSA算法性能,选用SSA[16]、RWSSA[19](采用随机行走策略扰动最优麻雀搜索位置的一种改进SSA算法)、AWPSO[21](采用自适应加权的一种改进粒子群优化算法)等基准算法进行比较。4种算法的参数设置如表2所示。

在实验中设置边缘设备数$n(s)$为5,任务数$n(i)$为100,优化目标为总任务能耗,即适应度函数为式(8),得到算法的适应度值变化曲线,如图4所示。4种算法的适应度值随迭代数递增呈现不同程度递减。综合分析寻优精度和收敛速度,MSSA性能最好。从迭代数10开始,MSSA的适应度值明显低于其他算法,随着迭代递进逐步拉

**表2 算法参数设置**

**Table 2 Setting of algorithm parameters**

| 算法 | 参数 |
|---|---|
| MSSA | $\mu_{sd}=20$, $\tau_{wv}=0.8$, $\mu_{sw}^{max}=30$, $\mu_{sw}^{min}=3$, $n(j)=100$ |
| SSA | $\mu_{sd}=20$, $\tau_{wv}=0.8$, $\mu_{sw}=20$, $n(j)=100$ |
| RWSSA | $\mu_{sd}=20$, $\tau_{wv}=0.8$, $\mu_{sw}=20$, $n(j)=100$ |
| AWPSO | $C_1=1.5$, $C_2=1.5$, $W=0.8$, $n(j)=100$ |

开差距，最终取得全局最小值，为 314.38，并且其算法收敛速度快，在[180，190]迭代区间收敛。RWSSA 取得的适应度值次之，为 395.12，在[230，240]迭代区间收敛。SSA 取得的适应度值偏大，为 423.91，在[420，430]迭代区间收敛，收敛速度相对最慢。AWPSO 取得的适应度值达到 443.78，寻优精度相对最差，在[220，230]迭代区间收敛。

考虑网络变化进一步分析算法的寻优精度和收敛速度。图 5 显示了不同任务下算法的最优适应度值。在负载比较小时（任务数≤30），4 种算法的寻优精度相当，取得的最优适应度值相差不大。随着任务数增加，MSSA 取得最优适应度值相比其他算法的优势拉大，位于[13.99，314.38]区间，其寻优精度最好。寻优精度其次为 RWSSA、SSA 和 AWPSO 算法，它们的最优适应度值位于[14.00，395.12][14.00，423.91][13.99，433.78]区间。

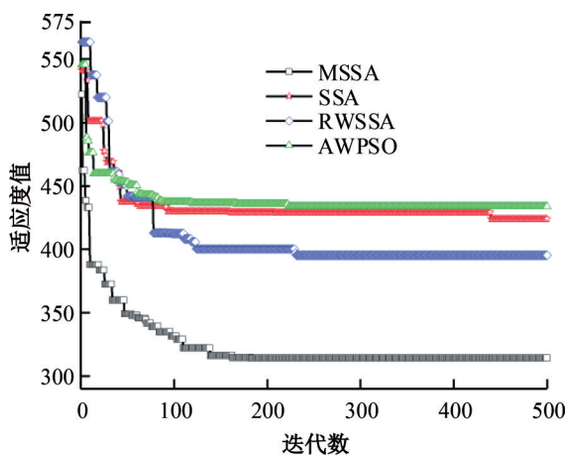



**图 4　算法性能（$n(i)$=100，$n(s)$=5）**

**Fig. 4　Performance of algorithms ($n(i)$=100, $n(s)$=5)**

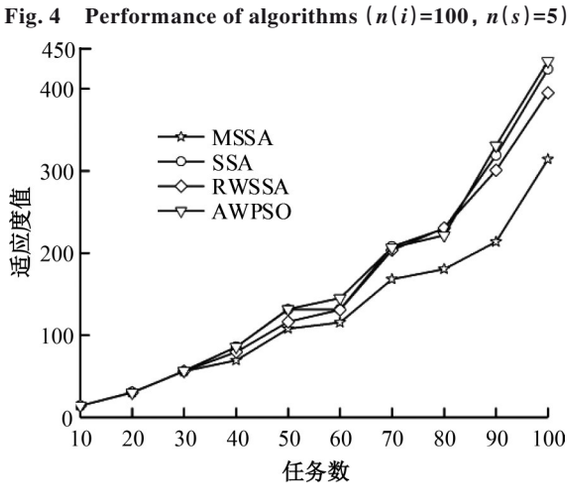



**图 5　不同任务下算法最优适应度值**

**Fig. 5　Fitness value of algorithm for different tasks**

图 6 显示了不同任务下算法的收敛速度。当任务数为 10 时，4 种算法均在[20，30]迭代区间完成收敛，且取得的最优适应值相近。随着任务数增加，4 种算法收敛取得各自最优适应值所需的迭代数增加，它们取得最优适应度值的差值增大。从算法收敛迭代数分析，AWPSO 最低，位于[37，364]区间，均值为 202，但其适应度值偏高，全局寻优精度明显低于 MSSA。MSSA 位于[65，383]区间，均值为 206，在取得最好寻优精度的同时也有不错的收敛性能。RWSSA 位于[31，397]区间，均值为 235，收敛速度优于 SSA。SSA 位于[25，435]区间，均值达到 262，相对收敛最慢。综合分析不同任务数下算法的寻优精度和收敛速度，MSSA 的鲁棒性好。

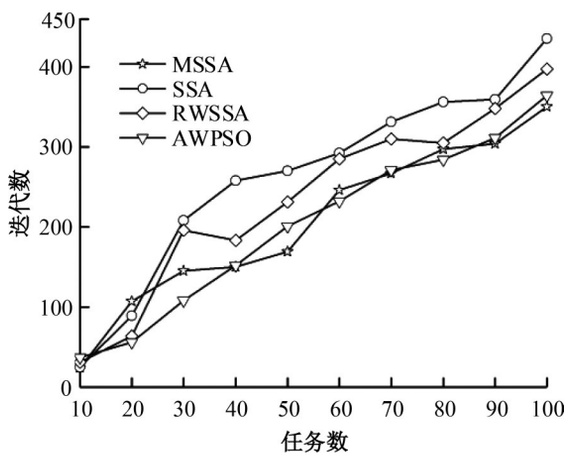



**图 6　不同任务下算法收敛**

**Fig. 6　Convergence of algorithm for different tasks**

### 4.2　任务卸载算法性能分析

从任务平均时延、总任务能耗、节点负载均衡度评估 HTMA 算法性能。为了均衡时延和能耗，在实验中均等考虑总任务时延和总任务能耗权重，设置 $\omega_T=\omega_E=0.5$。并且根据实验数据，将时延松弛变量 $\xi$ 设置为 20，将时延-能耗惩罚算子 $\gamma$ 设置为 $2\times10^6$，能合理量化时延-能耗惩罚，以融合超时惩罚能耗的总任务能耗为目标得到不同时延约束下的预卸载位置集，取得任务时延和能耗的综合优化。

为了验证 HTMA 算法的有效性，采用以下基准算法对比实验。这些算法（方法）包括经过改进的性能优良的启发式算法、经典任务卸载算法和传统任务卸载方法。

（1）IHRA（Iterative heuristic MEC resource allocation algorithm），为性能优良的启发式算法，基于混合整数线性规划迭代更新资源分配，卸载

任务到边缘或云中心执行[12]。

(2) TP (Task offloading with PSO algorithm),为性能优良的启发式算法,基于shrinkage因子改进PSO算法,综合时延约束和系统开销,优化任务到边缘或云中心执行[22]。

(3) TS (Task offloading with SSA algorithm),为性能优良的启发式算法,与HTMA算法类似,不同点在于第2步基于SSA算法得到不同时延约束下的预卸载位置集。

(4) TR (Task offloading with random algorithm),为经典任务卸载算法,边缘设备将任务随机卸载到本地边缘设备、其他边缘设备或云中心协作执行[3]。

(5) TC (Task offloading with cloud algorithm),为传统任务卸载方法,边缘设备接收任务将其卸载到云中心执行[7]。

4.2.1 任务平均时延

任务平均时延(Average task completion delay, ATCD)为总任务时延与总任务数的比值。ATCD值小,说明任务从发出请求到执行完成的平均时间短,用户服务质量好。

图7显示了不同任务下算法的任务平均时延。从ATCD均值总体分析算法性能,HTMA>IHRA>TS>TP>TR>TC。具体分析来看,TC将任务卸载到云中心执行,传输时延大,任务负载集中,其ATCD最高,在[69.00, 69.82]ms区间。TR随机卸载任务到边缘或云中心执行,没有优化卸载,其ATCD较高,在[30.40, 59.15]ms区间。TP和TS分别利用改进PSO和SSA算法优化任务卸载,它们的ATCD有降低,分别在[20.85, 52.21]ms和[20.85, 53.92]ms区间。IHRA迭代更新MEC资源分配优化任务卸载,其算法寻优性能有提升,ATCD在[20.48, 50.76]ms区间。HTMA融入超时惩罚能耗,基于MSSA优化不同时延约束下的预卸载位置集,全局寻优精度高,其ATCD低于其他算法,为[20.71, 48.19]ms,且随着任务数增加,ATCD相比其他算法增长趋缓。HTMA的ATCD值相比TS、TP、TC、TR、IHRA算法平均总体分别降低46.59%、20.05%、7.55%、6.07%、4.64%。

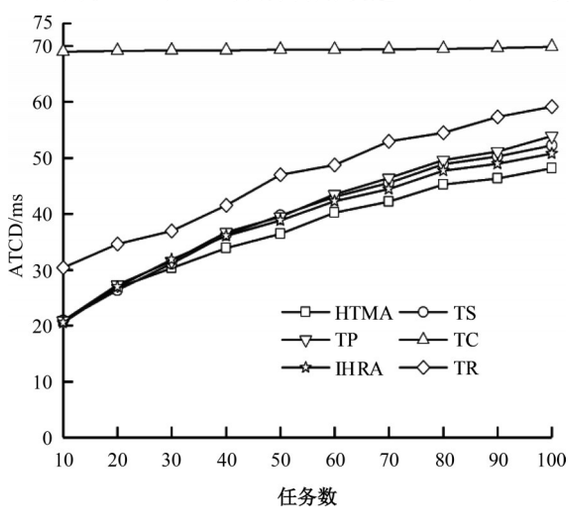



图7 不同任务下任务平均时延($n(s)$=5)

Fig. 7 ATCD for different tasks ($n(s)$=5)

在任务数不变情况下,改变边缘设备数进一步评测算法的任务平均时延。如图8所示,除TC外,其他算法的ATCD随边缘设备数增加而有不同程度减小。TC将任务卸载到云中心执行,其ATCD不会因边缘设备数改变而有明显变化,位于均值72.66 ms附近。TR将任务随机卸载到边缘设备或者云中心执行,容易出现节点负载失衡,任务平均时延偏高,其ATCD均值达到60.18 ms。TP和TS分别利用改进PSO和SSA算法优化任务卸载位置,能有效降低任务平均时延,其ATCD均值分别为52.66、50.87 ms。IHRA基于启发式资源分配优化任务卸载位置,寻优能力相比TP和TS有提升,其ATCD均值为50.25 ms。HTMA基于MSSA融入超时惩罚能耗优化不同时延约束下任务卸载位置,全局寻优能力强,其ATCD值均低于其他算法,为48.56 ms,相比TC、TR、TP、TS、IHRA算法平均总体分别降低33.17%、19.32%、7.80%、4.55%、3.37%。

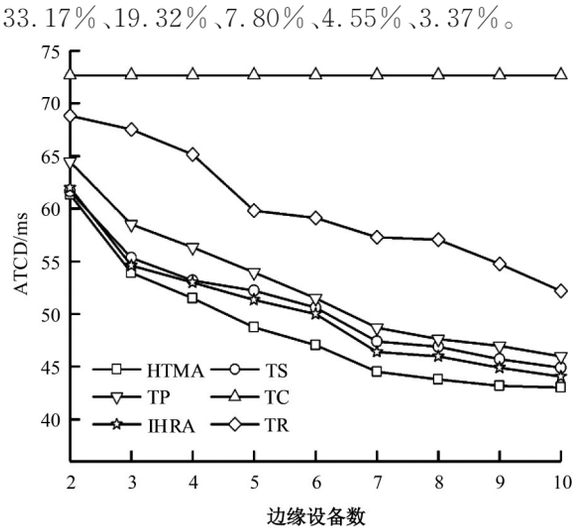



图8 不同边缘设备下任务平均时延($n(i)$=100)

Fig. 8 ATCD for different edge devices ($n(i)$=100)

4.2.2　总任务能耗

总任务能耗(Total task energy consumption, TTEC)为完成所有任务所产生的能耗。TTEC值小,说明系统整体能效高,有利于保障服务提供商利益。

图9显示了不同任务下算法的总任务能耗,6种算法的TTEC随着任务数增加呈现线性增长。按TTEC均值比较分析,TC最大,其次分别为TR、TP、TS、IHRA算法,HTMA最小。TC将任务卸载到云中心集中执行,任务的传输和执行能耗高,其TTEC偏高,在[20.60,612.33]J区间。TR将任务随机卸载,没有优化任务卸载,其TTEC也较高,在[15.70,470.46]J区间。TP、TS分别利用改进PSO和SSA算法优化任务卸载,它们的TTEC相比TR和TC有所减少,分别在[12.11,299.06]J和[12.11,285.02]J区间。IHRA基于启发式资源分配优化任务卸载,其TTEC较低,在[12.11,274.87]J区间。HTMA基于MSSA改进位置搜索,叠加时延能耗惩罚优化任务卸载,在任务数>50时优势明显,其TTEC位于[12.11,262.01]J区间,相比TC、TR、TS、TP、IHRA算法平均总体分别降低61.54%、48.51%、13.05%、9.03%、5.06%。

在任务数不变情况下,改变边缘设备数进一步评估算法的总任务能耗,如图10所示。与评估ATCD类似,除TC外,其他算法的TTEC随边缘设备增加有不同程度降低。TC将任务卸载到云

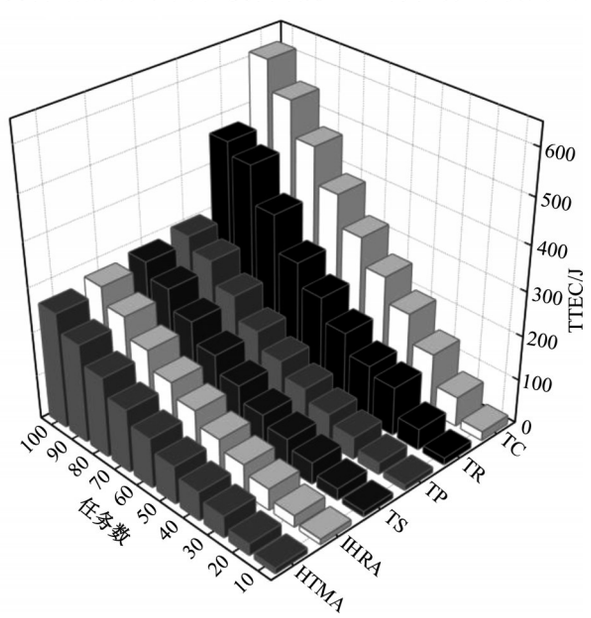



**图9　不同任务下总任务能耗($n(s)$=5)**

**Fig. 9　TTEC for different tasks ($n(s)$=5)**

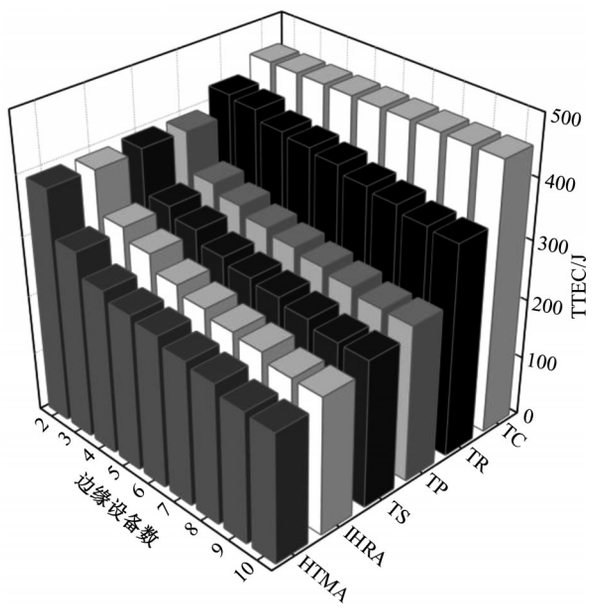



**图10　不同边缘设备下总任务能耗($n(i)$=100)**

**Fig. 10　TTEC for different edge devices ($n(i)$=100)**

中心,在任务数固定时,其TTEC不会受到边缘设备变化影响,均值在452.35 J附近。TR随机卸载任务到边缘设备或云中心,随着边缘设备数增加,节点负载有改善,其TTEC有降低,均值为389.20 J。TP和TS分别采用改进PSO和SSA算法优化任务卸载位置,随着边缘设备数增加能迅速降低TTEC,均值分别为300.45 J和289.40 J。IHRA结合混合整数线性规划优化任务卸载位置,能有效降低TTEC,均值为278.38 J。HTMA改进发现者、跟随者和预警者3类种群的卸载位置更新,全局寻优精度高,取得的TTEC最低,均值为268.46 J,相比TC、TR、TP、TS、IHRA算法平均总体分别降低40.65%、31.02%、10.65%、7.24%、3.56%。

4.2.3　节点负载均衡度

节点负载均衡度(Node load balance degree, NLBD)为各边缘设备的节点负载率标准差,记为$\psi$:

$$\psi=\sqrt{\frac{1}{n(s)}\sum_{s=1}^{n(s)}(L_s-\overline{L})^2} \tag{20}$$

$$L_s=(\sum_{s}l_i\chi_i)/f_s \tag{21}$$

式中:$L_s$为边缘设备的节点负载率,是边缘设备上执行任务所需计算量与其最大算力的比值;$\overline{L}$为边缘设备的平均节点负载率。$\psi$值越小,说明边缘设备负载越均衡,有助于减少总任务时延和总任务能耗。

由于TC将任务派遣到云中心执行,其边缘设备节点负载率为0,在实验中主要考虑其他5种算法的节点负载均衡度。如图11所示,随着任务数增加,节点负载也有所增加,各算法的NLBD也有不同程度增大。比较分析不同任务下的NLBD均值,TR最大,其次分别为TP、TS、IHRA算法,HTMA最小。TR随机卸载任务到边缘或云中心,节点负载率相差较大,其NLBD偏大,在[0.225 5,0.469 8]区间。TP、TS分别采用改进PSO和SSA算法优化任务卸载,可缩小节点负载率差值,降低NLBD,分别在[0.211 9,0.283 2]和[0.210 8,0.279 6]区间。IHRA基于启发式MEC资源分配优化任务卸载,降低节点负载率差值,从而减小NLBD,位于[0.210 2,0.235 7]区间。HTMA算法结合时延-能耗惩罚,基于MSSA优化任务卸载位置,节点负载率相对均衡,取得的NLBD最小,位于[0.193 1,0.230 2]区间,相比TR、TP、TS、IHRA算法,平均总体分别降低36.54%、15.20%、14.15%、9.01%。

在任务数不变的情况下,改变边缘设备数进一步评估算法的节点负载均衡度。如图12所示,随着边缘设备数增加,其负载有所减轻,算法的NLBD有不同幅度降低。其中,TR的NLBD从0.550 5降为0.307 2,但没有优化任务卸载,其NLBD均值偏高,达到0.433 4。其他4种算法对任务卸载有不同程度优化,其节点负载相对平衡,NLBD均值偏低。TP从0.297 9降低到0.220 1,均值为0.265 5;TS从0.297 5降低到0.218 2,均值为0.262 7;IHRA从0.296 2降低到0.215 3,均值为0.265 5;HTMA从0.272 1降低到0.200 8,

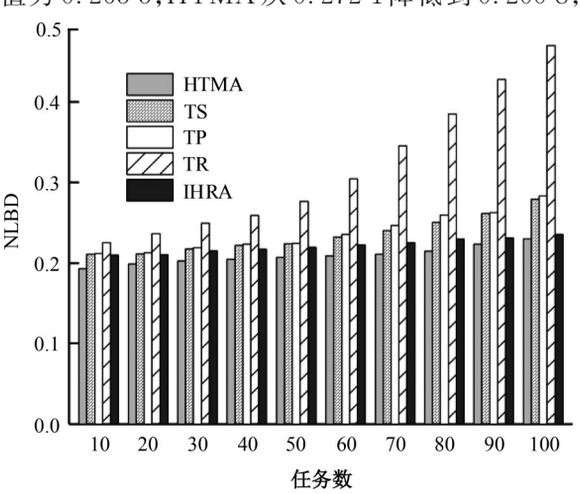



**图11 不同任务下节点负载均衡度($n(s)$=5)**

**Fig. 11 NLBD for different tasks ($n(s)$=5)**

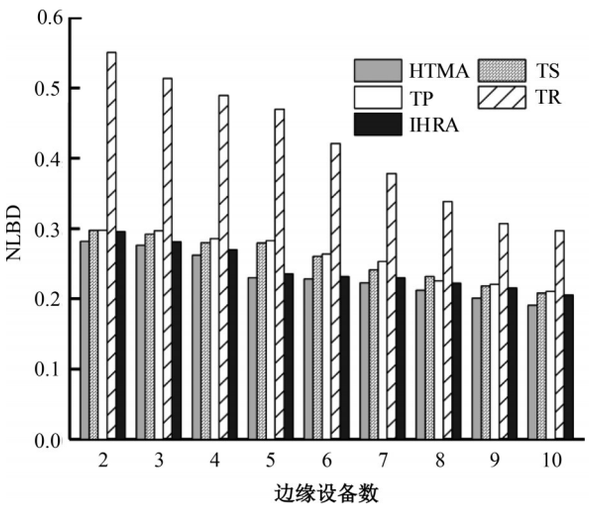



**图12 不同边缘设备下节点负载均衡度($n(i)$=100)**

**Fig. 12 NLBD for different edge devices ($n(i)$=100)**

均值为0.235 6,其NLBD均值最低,相比TR、TP、TS、IHRA算法平均总体分别降低44.85%、11.07%、10.12%、4.73%。

### 4.3 任务卸载相关参数讨论

#### 4.3.1 时延-能耗惩罚算子$\gamma$

$\gamma$能明显影响任务卸载的时延和能耗。不同$\gamma$值下HTMA取得的总任务时延(Total task completion delay, TTCD)和总任务能耗(TTEC)如图13(a)所示。在$\gamma$为1~1 000时,TTCD达到11 415.86~7 383 ms,而TTEC为206.59~224.03 J。说明$\gamma$偏小时,针对不同时延约束下任务的超时惩罚能耗低,感知任务时延的能耗惩罚对任务卸载影响小,取得的TTCD过大,TTEC偏小。在$\gamma$增大到$5\times10^4$时,其TTCD降低到4 930.19 ms,为最高值的40%以下,但是TTEC有所增加,为262.17 J。说明随着$\gamma$增大,感知任务时延的能耗惩罚对任务卸载影响增大,能有效降低TTCD,而TTEC的变化相对较小。在$\gamma$为$2\times10^6$时,取得的TTCD最小,为4 819.32 ms。当$\gamma$继续增大,达到$10\times10^6$时,TTCD反而有所增加,为5 398.04 ms。如图13(b)所示,当$\gamma$在合理区间$5\times10^5$~$10\times10^6$时,取得的TTEC在241.52~262.15 J范围,均值为250.17 J,取得的TTCD在4 819.32~5 398.04 ms范围,均值为5 166.43 ms。

#### 4.3.2 时延约束与时延松弛变量

时延约束$t_{\xi}$与任务最大时延$t_i$和时延松弛变量$\xi$有关,直接影响超时惩罚能耗的阈值。不同$t_{\xi}$下的总任务时延和总任务能耗如图14所示。$t_i$为50 ms,$\xi$为40。随着$t_{\xi}$增大,即时延惩罚阈值增大

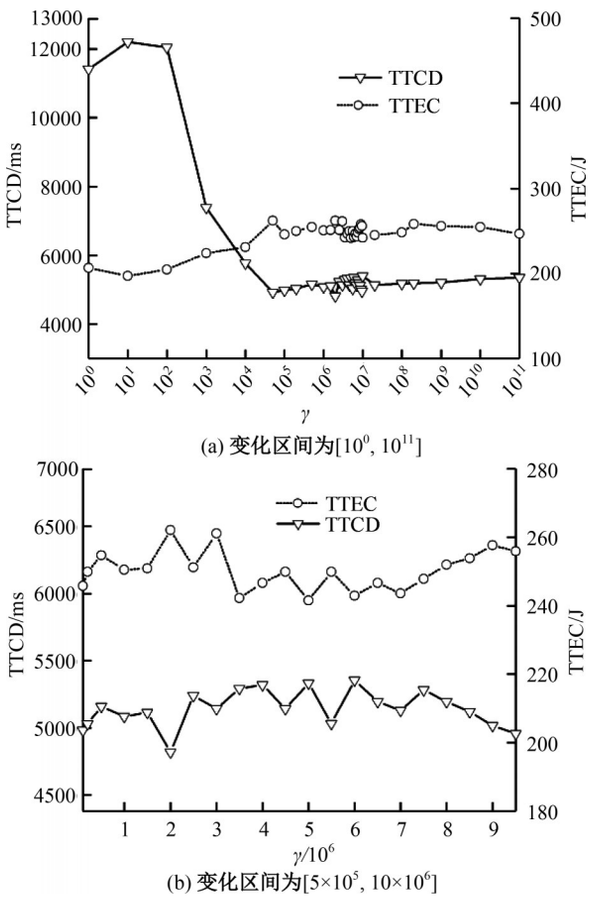



**图 13　不同 $\gamma$ 的总任务时延和总任务能耗 ($n(i)$=100, $n(s)$=5)**

**Fig. 13　TTCD and TTEC for different $\gamma$ ($n(i)$=100, $n(s)$=5)**

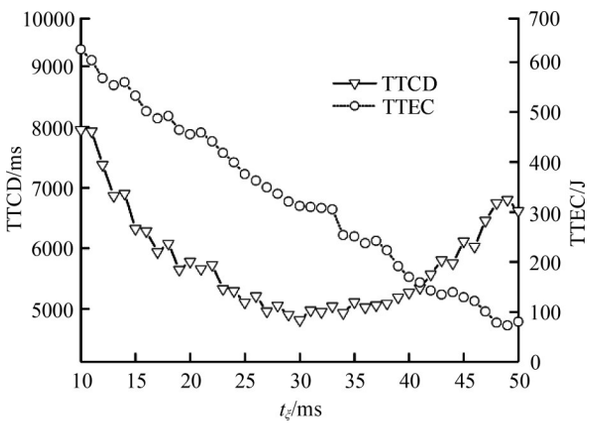



**图 14　不同 $t_\xi$ 的总任务时延和总任务能耗 ($n(i)$=100, $n(s)$=5)**

**Fig. 14　TTCD and TTEC for different $t_\xi$ ($n(i)$=100, $n(s)$=5)**

时,TTCD 和 TTEC 随之呈现不同变化。其中,TTCD 在 $t_\xi \in [10, 30]$ ms 时逐步降低,在 $t_\xi$=30 ms 时得到全局最低值,为 4 819.32 ms,在 $t_\xi \in [31, 50]$ ms 时,反而有所提高,增加到 6 616.37 ms。TTEC 总体呈现降低变化,从 624.90 J($t_\xi$=10 ms)减小到 73.36 J($t_\xi$=49 ms),再略提升到 80.18 J($t_\xi$=50 ms)。统计发现,在时延约束 $t_\xi$ 偏小时,时延惩罚阈值小,得到的 TTCD 和 TTEC 偏大。说明时延约束过小,经过任务卸载的任务时延多数大于时延惩罚阈值,任务的超时惩罚能耗过大,任务卸载优化的性能不高。

时延松弛变量 $\xi$ 决定算法的迭代搜索轮次,对预卸载任务集规模和算法收敛速度有影响。一般来说,$\xi$ 越大,预卸载任务集的规模越大,有利于扩大算法的搜索广度,但是会增加算法的时间复杂度,影响收敛速度。图 15 显示了不同 $\xi$ 下综合总任务完成延迟和总任务能耗的总系统服务能耗。当 $\xi$ 偏小时,预卸载任务集的规模小,通过任务卸载优化得到的总系统服务能耗偏高。随着 $\xi$ 增大,扩大了预卸载任务集的规模,任务卸载优化的性能明显提升,取得的总系统服务能耗随之减少。当 $\xi$ 大于 15 时,即可取得综合 TTCD 和 TTEC 的总系统服务能耗最低。

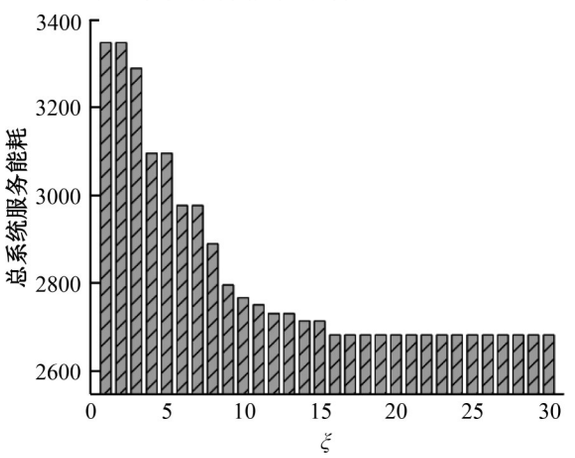


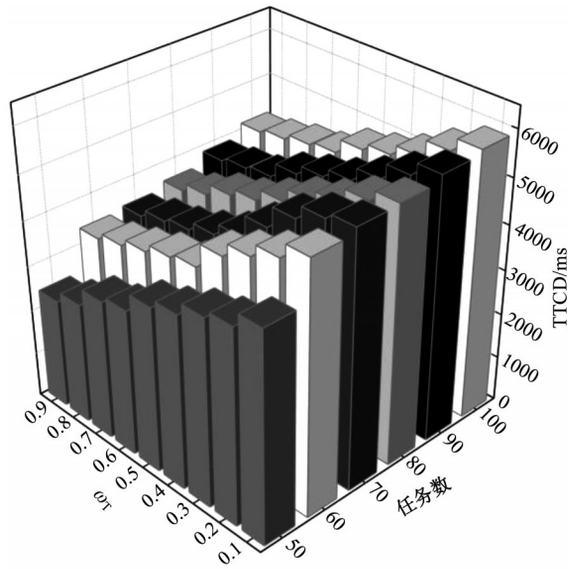

**图 15　不同 $\xi$ 的总系统服务能耗($n(i)$=100, $n(s)$=5)**

**Fig. 15　Overall system service efficiency for different $\xi$ ($n(i)$=100, $n(s)$=5)**

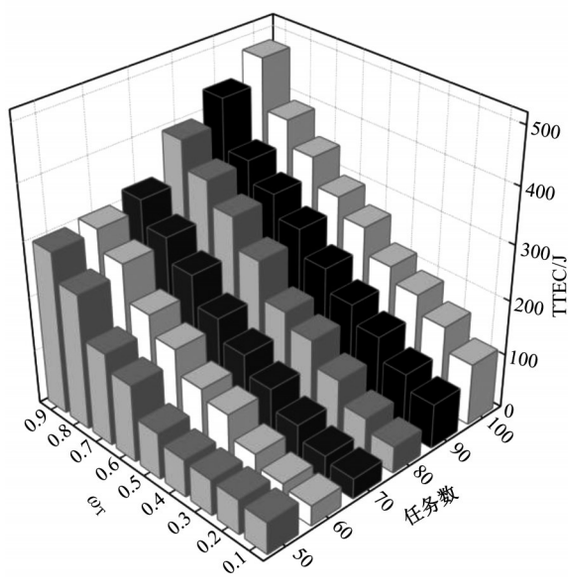
4.3.3　总任务能耗权重系数和总任务时延权重系数

由式(9)的 $C_1$ 约束可知,总任务时延权重系数 $\omega_T$ 和总任务能耗权重系数 $\omega_E$ 存在 $\omega_T+\omega_E=1$,若 $\omega_T$ 增大,则 $\omega_E$ 减小。图 16 表示不同 $\omega_T$(亦改变 $\omega_E$)下总任务时延和总任务能耗的变化趋势。

在任务数分别为 50、60、70、80、90、100 的情况下,$\omega_T$ 从 0.1 增加到 0.9,如图 16(a)所示,HTMA 取得的 TTCD 分别从 4 609.43、5 482.65、5 610.89、5 640.61、5 787.07、5 884.98 ms,降低为 2 427.97、3 384.22、3 456.67、3 616.79、3




(a) 总任务时延




(b) 总任务能耗

**图16 不同$\omega_T$的总任务时延和总任务能耗**
**($n(i)$=100, $n(s)$=5)**

**Fig. 16 TTCD and TTEC for different $\omega_T$**
**($n(i)$=100, $n(s)$=5)**

896.07、4 170.05 ms。说明随着$\omega_T$不断增加,任务卸载优化考虑任务时延的比重增大,使取得的TTCD值总体呈现递减变化。同时可以看到,如图16(b)所示,HTMA取得的TTEC分别从58.29、37.04、37.28、48.55、81.79、112.98 J,增加到297.63、303.07、320.89、394.39、437.82、480.29 J。说明在$\omega_T$不断增加的同时,$\omega_E$不断减少,任务卸载优化考虑任务能耗的比重不断减少,使得取得的TTEC值总体呈现递增变化。

## 5 结束语

为满足新兴应用需求,提供高效快速的计算服务,本文兼顾用户和服务提供商利益,利用改进群体智能算法优化时延和能耗约束的移动边云协作任务卸载。首先,考虑任务时延和能耗,建模移动边云协作计算。然后,使用移动边云协作计算改进麻雀搜索算法,通过飞行者动量改进发现者的任务卸载位置更新,通过非线性步长搜索因子和感知负载度的正余弦扰动量子改进跟随者的任务卸载位置更新,通过融合位置搜索态偏离熵和非线性预警系数自适应调整预警者数并改进预警者的任务卸载位置更新,综合提升任务卸载的全局搜索能力和速度。最后,结合超时惩罚能耗和不同时延松弛变量,基于改进麻雀搜索算法迭代更新不同时延约束下的预卸载位置集,贪心比较均衡赋权的总任务时延和总任务能耗,进一步优化任务卸载。通过仿真实验验证了本文算法的有效性。MSSA算法相比SSA、RWSSA、AWPSO对比算法,取得了最高的寻优精度和不错的收敛速度,且算法的鲁棒性好。HTMA算法在不同任务和边缘设备下,其ATCD、TTEC、NLBD均优于TR、TC、TS、TP、IHRA这5种基准任务卸载算法,能有效降低任务时延和能耗,均衡边缘设备任务负载。下一步将结合深度强化学习、联邦学习进一步改进现有计算任务卸载方法,并将改进的方法应用于端-边-云协作下的计算任务卸载优化。

**参考文献:**

[1] Kim T, Sathyanarayana S D, Chen S Q, et al. MODEMS: optimizing edge computing migrations for user mobility[J] IEEE Journal on Selected Areas in Communications, 2023, 41(3):675-689.

[2] 张依林, 梁玉珠, 尹沐君, 等. 移动边缘计算中计算卸载方案研究综述[J]. 计算机学报, 2021, 44(12): 2408-2432.
Zhang Yi-lin, Liang Yu-zhu, Yin Mu-jun, et al. Survey on the methods of computation offloading in mobile edge computing[J]. Chinese Journal of Computers, 2021,44(12):2408-2432.

[3] 苏命峰, 王国军, 李仁发. 基于利益相关视角的多维QoS云资源调度方法[J]. 通信学报, 2019, 40(6): 102-115.
Su Ming-feng, Wang Guo-jun, Li Ren-fa. Multidi-

mensional QoS cloud computing resource scheduling method based on stakeholder perspective[J]. Journal on Communications, 2019, 40(6): 102-115.

[4] Wang S G, Guo Y, Zhang N, et al. Delay-aware microservice coordination in mobile edge computing: a reinforcement learning approach[J]. IEEE Transactions on Mobile Computing, 2021, 20(3): 939-951.

[5] Laskaridis S, Venieris S I, Almeida M, et al. SPINN: synergistic progressive inference of neural networks over device and cloud[C]//The 26th ACM/IEEE International Conference on Mobile Computing and Networking, New York, USA, 2020: 1-15.

[6] 刘伟, 黄宇成, 杜薇, 等. 移动边缘计算中资源受限的串行任务卸载策略[J]. 软件学报, 2020, 31(6): 1889-1908.
Liu Wei, Huang Yu-cheng, Du Wei, et al. Resource-constrained serial task offload strategy in mobile edge computing[J]. Journal of Software, 2020, 31(6): 1889-1908.

[7] Wang T, Lu Y C, Wang J H, et al. EIHDP: edge-intelligent hierarchical dynamic pricing based on cloud-edge-client collaboration for iot systems[J]. IEEE Transactions on Computers, 2021, 70(8): 1285-1298.

[8] Zhao J H, Li Q P, Gong Y, et al. Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks[J]. IEEE Transactions on Vehicular Technology, 2019, 68(8): 7944-7956.

[9] Gupta S, Chakareski J. Lifetime maximization in mobile edge computing networks[J]. IEEE Transactions on Vehicular Technology, 2020, 69(3): 3310-3321.

[10] Jin P P, Fei X C, Zhang Q X, et al. Latency-aware VNF chain deployment with efficient resource reuse at network edge[C]//The 39th IEEE Conference on Computer Communications, New Jersey, USA, 2020: 267-276.

[11] Zou J F, Hao T B, Yu C, et al. A3C-DO: a regional resource scheduling framework based on deep reinforcement learning in edge scenario[J]. IEEE Transactions on Computers, 2021, 70(2): 228-239.

[12] Ning Z L, Dong P R, Kong X J, et al. A cooperative partial computation offloading scheme for mobile edge computing enabled internet of things[J]. IEEE Internet of Things Journal, 2019, 6(3): 4804-4814.

[13] Chen L X, Zhou S, Xu J. Computation peer offloading for energy-constrained mobile edge computing in small-cell networks[J]. IEEE/ACM Transactions on Networking, 2018, 26(4): 1619-1632.

[14] Lai F, Zhu X F, Madhyastha H V, et al. Oort: efficient federated learning via guided participant selection[C]//The 15th USENIX Symposium on Operating Systems Design and Implementation, Berkeley, USA, 2021: 19-35.

[15] Biswas N, Wang Z J, Vandendorpe L, et al. On joint cooperative relaying, resource allocation, and scheduling for mobile edge computing networks[J]. IEEE Transactions on Computers, 2022, 70(9): 5882-5897.

[16] Xue J K, Shen B. A novel swarm intelligence optimization approach: sparrow search algorithm[J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.

[17] 国强, 朱国会, 李万臣. 基于混沌麻雀搜索算法的TDOA/FDOA定位[J]. 吉林大学学报:工学版, 2023, 53(2): 593-600.
Guo Qiang, Zhu Guo-hui, Li Wan-chen. TDOA/FDOA localization based on chaotic sparrow search algorithm[J]. Journal of Jilin University (Engineering and Technology Edition), 2023, 53(2): 593-600.

[18] Cheng B P, Fang Y W, Peng W S. Improved sparrow search algorithm based on normal cloud model and niche recombination strategy[J]. IEEE Transactions on Cloud Computing, 2023, 11(3): 2529-2545.

[19] Chang Z Z, Gu Q H, Lu C W, et al. 5G private network deployment optimization based on RWSSA in open-pit mine[J]. IEEE Transactions on Industrial Informatics, 2022, 18(8): 5466-5476.

[20] 苏命峰, 王国军, 李仁发. 边云协同计算中基于预测的资源部署与任务调度优化[J]. 计算机研究与发展, 2021, 58(11): 2558-2570.
Su Ming-feng, Wang Guo-jun, Li Ren-fa. Resource deployment with prediction and task scheduling optimization in edge cloud collaborative computing[J]. Journal of Computer Research and Development, 2021, 58(11): 2558-2570.

[21] Liu W B, Wang Z D, Yuan Y, et al. A novel sigmoid-function-based adaptive weighted particle swarm optimizer[J]. IEEE Transactions on Cybernetics, 2021, 51(2): 1085-1093.

[22] Wang T, Zhang Y L, Xiong N A, et al. An effective edge-intelligent service placement technology for 5G-and-beyond industrial IoT[J]. IEEE Transactions on Industrial Informatics, 2022, 18(6):4148-4157.