DOI:10.16356/j.2097-6771.2026.02.016

面向多边缘制造场景的动态协作粒子群优化任务卸载方法

唐敦兵,董浩然,李东东,张泽群,袁星高
(南京航空航天大学机电学院,南京 210016)

摘要: 针对制造车间多边缘任务卸载过程中存在的资源分配不均、计算效率低下等问题,提出了一种基于动态协作粒子群优化(Dynamic collaborative particle swarm optimization, DCPSO)的多边缘任务卸载优化方法。首先,为提高初始解的质量以提升整体优化效率,设计了一种结合随机采样与适应度引导的贪心机制的混合初始化策略,实现解的多样性与质量的平衡。然后,为了增强算法在复杂空间中的探索能力,构建了一种动态子群协作更新机制,通过动态子群划分与自适应粒子更新,显著提升了收敛速度与子代解的质量。最后,进一步引入变异机制增强算法的局部搜索能力,提升算法跳出局部最优的能力。实验结果表明,与5种基线算法相比,DCPSO算法在收敛性、稳定性和敏感性方面均表现出显著优势。

关键词: 边缘计算;智能制造;任务卸载;多目标优化;粒子群算法
中图分类号: TP393.0 **文献标志码:** A **文章编号:** 1005-2615(2026)02-0400-12

Dynamic Collaborative Particle Swarm Optimization Task Offloading Method for Multi-edge Manufacturing Scenarios

TANG Dunbing, *DONG Haoran*, *LI Dongdong*, *ZHANG Zequn*, *YUAN Xinggao*
(College of Mechanical and Electrical Engineering, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China)

Abstract: This paper addresses the issues of uneven resource allocation and low computational efficiency in the multi-edge task offloading process within manufacturing workshops. A multi-edge task offloading optimization method based on dynamic collaborative particle swarm optimization (DCPSO) is proposed. First, to enhance the quality of initial solutions and thereby improve overall optimization efficiency, a hybrid initialization strategy integrating random sampling and fitness-guided greedy mechanisms is designed to achieve a balance between solution diversity and quality. Second, to strengthen the algorithm's exploration capability in complex spaces, a dynamic subgroup collaboration update mechanism is developed, employing dynamic subgroup partitioning and adaptive particle updates to significantly enhance convergence speed and the quality of offspring solutions. Finally, a mutation mechanism is introduced to augment the algorithm's local search capability and improve its ability to escape local optima. Experimental results demonstrate that, compared to the five baseline algorithms, the DCPSO algorithm exhibits significant advantages in terms of convergence, robustness, and sensitivity.

Key words: edge computing; smart manufacturing; task offloading; multi-objective optimization; particle swarm optimization (PSO)

基金项目: 国家自然科学基金青年基金(52305539);国家自然科学基金重大培育项目(92267109)。
收稿日期: 2025-07-22;**修订日期:** 2025-09-29
通信作者: 李东东,男,博士研究生,E-mail:lddya1996@nuaa.edu.cn。

引用格式: 唐敦兵,董浩然,李东东,等. 面向多边缘制造场景的动态协作粒子群优化任务卸载方法[J]. 南京航空航天大学学报(自然科学版),2026,58(2):400-411. TANG Dunbing, DONG Haoran, LI Dongdong, et al. Dynamic collaborative particle swarm optimization task offloading method for multi-edge manufacturing scenarios[J]. Journal of Nanjing University of Aeronautics & Astronautics(Natural Science Edition),2026, 58(2):400-411.

随着智能制造技术的快速发展，制造车间内各类传感器、工业机器人和智能终端的数量呈指数级增长，由此产生的计算密集型任务（如实时质量检测、设备预测性维护和数字孪生等）对传统集中式计算架构提出了严峻挑战[1-2]。原有的集中式处理模式在应对海量数据并发处理时，暴露出响应延迟高、带宽压力大且能耗激增等固有缺陷，已难以满足现代智能制造系统对实时性、可靠性和能效的严苛要求[3]。

此外，5G通信技术的规模化部署正成为智能制造发展的关键驱动力，其低时延特性推动制造业加速向分布式边缘计算范式转型，以应对集中式计算架构的固有局限。通过将计算任务就近上传至车间网络边缘节点进行处理，并将处理结果实时回传至设备，边缘计算能够有效满足工业物联网设备的实时计算需求，同时显著降低系统延迟和能耗[4]。

边缘计算问题通常可以归结为计算任务在工业终端设备与边缘服务器之间的最优分配问题，需要统筹节点计算能力、网络通信时延和系统能耗特性等关键因素[5]。针对这一复杂优化问题，学界已提出从传统基础调度优化方法（如先到先服务[6]和轮询调度[7]）到启发式优化方法（包括蚁群优化[8]和遗传算法[9]）等多种解决方案。其中，粒子群优化（Particle swarm optimization，PSO）算法凭借其简单性、有效性等优势，在分布式优化问题领域展现出突出的应用潜力[10]。例如，Peng等[11]在移动边缘云计算中应用PSO实现连续不间断服务，李金等[12]依托容器技术与粒子群算法的结合优化分布式边缘节点资源利用率，Li等[13]则基于PSO实现云边协同框架下的电网数字化资源多目标优化分配。然而，传统PSO算法在解决制造环境边缘计算问题时依旧存在一些局限性。在初始化方面，传统PSO的初始化阶段通常依赖随机采样，可能导致复杂解空间中收敛缓慢[14]。在粒子更新方面，大多数算法采用静态参数设置，难以适应制造环境中异构且动态的工作负载特性[15]。此外，传统PSO算法易陷入局部最优，特别是在高维搜索空间中，粒子更新机制可能导致过早收敛，限制了对全局最优解的探索能力[16]。

鉴于此，一些研究对传统PSO算法进行了改进。Liu等[17]以最小化时延为优化目标，通过引入任务优先级并调整粒子速度更新概率设计了多元粒子群算法。Zhang等[18]将混沌理论引入量子PSO进行时延能耗联合优化。申秀雨等[19]进一步考虑边缘卸载场景中的安全需求，提出双层改进PSO，通过优化初始化策略与迭代机制，有效降低系统总代价。Velrajan等[20]聚焦于用户服务质量，提出引入“共振”参数的闭环自适应PSO，动态计算服务迁移的性能阈值。吴波等[21]设计了基于反向学习与精英保留策略的粒子群算法，优化多基站多任务卸载环境的系统性能。尽管现有研究在PSO的改进方面取得了显著成效，但这些改进措施尚未从根本上解决算法固有的早熟收敛问题，粒子群算法在收敛速度与全局搜索能力之间的平衡仍有待进一步探索。

针对上述问题，本文提出一种动态协作粒子群优化（Dynamic collaborative particle swarm optimization，DCPSO）算法用于解决制造环境下的多边缘任务卸载问题。首先，采用一种混合初始化策略生成初代种群，通过融合随机采样与适应度引导的贪心机制，显著提升了初始解质量。其次，设计了一种动态子群协作更新机制，实现动态子群划分和自适应粒子更新，提高了收敛速度与子代解的质量。最后，引入受遗传算法启发的变异机制[22]，有效维持了种群多样性，增强了算法的局部搜索能力。

本文的创新贡献主要体现在：

（1）建立面向制造车间的多边缘任务卸载问题的系统模型，通过加权代价函数综合优化计算时延与能耗指标；

（2）设计混合初始化策略，在保证粒子多样性的同时大幅优化初始解质量，有效提高算法收敛速度；

（3）构建动态子群协作更新机制，实现全局动态协同更新，减小群体早熟收敛的风险，提升对复杂搜索空间的探索能力；

（4）引入变异机制，有效避免算法陷入局部最优解，维持了种群多样性，避免了算法陷入局部最优问题。

实验结果表明，相较于基线算法，DCPSO算法在收敛性分析中表现出更快的收敛速度和更好的求解质量，稳定性分析验证了其鲁棒性优势，进一步的敏感性分析则证明该算法在不同规模的制造场景中均具有更好的适应性。

1　多边缘任务卸载问题模型

1.1　系统场景

目前，制造车间的网络拓扑结构多采用传统的集中式或部分分布式架构，通常以中央服务器为核心，各类自动化设备通过有线或无线网络连接至中央服务器[23]。尽管该拓扑结构具有设计简单和成本低廉的优势，但在实际应用中存在着一些缺陷。

首先,集中式架构对中央服务器的依赖性强,一旦服务器发生故障或过载,可能导致整个系统瘫痪[24]。其次,传统架构在应对动态变化的计算需求时,灵活性和扩展性较差,难以满足智能制造车间对低延迟、高可靠性的要求。针对这些问题,本文提出了一种基于多边缘任务卸载决策的系统模型,通过设备层与边缘层的分布式协同架构,充分利用边缘服务器的计算能力和设备的本地处理能力,实现任务的高效分配与执行。

本文所构建的制造车间多边缘任务卸载决策系统模型如图1所示。在整体,系统分为设备层、网络层与边缘层。设备层由制造车间内的各类自动化机械设备组成,包括但不限于数控机床、自动导向车、工业机器人和自动化立体仓库等,是产生计算任务的主体,其本身也具有一定的计算能力。同时,每个设备配备一台5G客户前端设备(Customer premise equipment, CPE),并通过5G物联网卡接入运营商5G网络。网络层由运营商5G网络和车间网络组成,负责设备层与边缘层之间的无线通信与数据传输。其中,运营商5G网络下的5G核心网的用户面功能(User plane function, UPF)与车间路由器之间通过通用路由封装(Generic routing encapsulation, GRE)隧道搭建端到端局域网。边缘层由多台部署在制造车间内部的边缘服务器组成,是执行计算任务的主体,每台服务器具有较高存储能力与计算能力,其中一台作为边缘调度器,负责收集全局信息,执行算法并做出卸载决策。各边缘服务器通过无线方式接入车间路由器,路由器通过5G专线接入运营商5G网络。

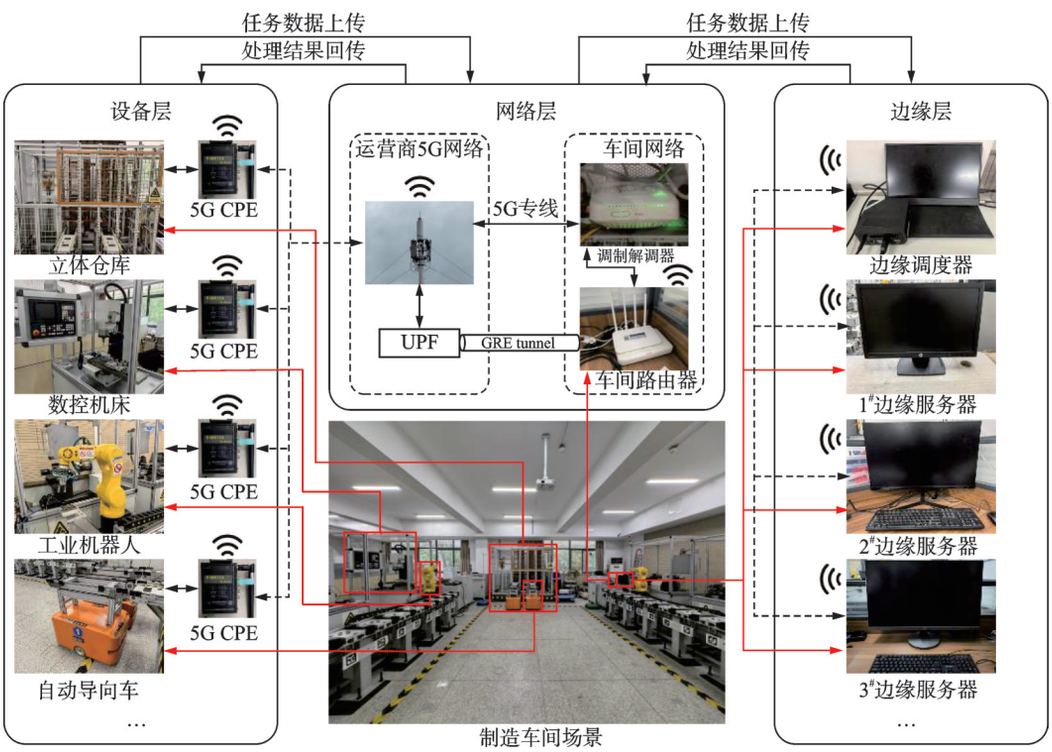

图1 制造车间多边缘任务卸载决策系统模型

Fig.1 Multi-edge task offloading decision system model for manufacturing workshops

为了便于分析,本文对制造车间多边缘任务卸载决策系统进行数学建模。假设制造车间场景中包含N台机械设备,其序号为$n\in\{1,2,3,\cdots,N\}$。每台机械设备存在L个计算任务,其序号为$l\in\{1,2,3,\cdots,L\}$。车间内共有M台边缘服务器,其序号为$m\in\{1,2,3,\cdots,M\}$。

每台机械设备的模型用$R_n=\{f_n,P_n^{\text{trans}},P_n^{\text{comp}},L_n,H_n^m,Q_n\}$表示。其中:$f_n$表示机械设备$R_n$的CPU频率,$P_n^{\text{trans}}$表示机械设备$R_n$的发射功率,$P_n^{\text{comp}}$表示机械设备$R_n$处理任务时的功率,$L_n$表示机械设备$R_n$产生的任务数量,$H_n^m$表示机械设备$R_n$向边缘服务器$S_m$传输数据时的上行链路信道增益,$Q_n$表示机械设备$R_n$的CPU以正常频率运行时所能承载的最大任务数。

每台边缘服务器的模型用$S_m=\{f_m,P_m^{\text{comp}},Q_m\}$表示。其中:$f_m$表示边缘服务器$S_m$的CPU频率,$P_m^{\text{comp}}$表示边缘服务器$S_m$处理任务时的功率,$Q_m$表示边缘服务器$S_m$的CPU以正常频率运行时所能承载的最大任务数。

每个任务用1个二元组$D_{n,l}=\{a_{n,l},b_{n,l}\}$表示。其中:$D_{n,l}$表示第$n$台机械设备上的第$l$个任务,$a_{n,l}$

表示该任务的计算位置，$a_{n,l}\in\{0,1,2,\cdots,m,\cdots,M\}$；$b_{n,l}$表示该任务的数据量大小，$b_{n,l}\geqslant 0$。由所有任务的计算位置构成的卸载决策矩阵$\boldsymbol{A}$为

$$\boldsymbol{A}=\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,l} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,l} \\ \vdots & \vdots & & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,l} \end{bmatrix}_{N\times L} \tag{1}$$

式中：当$a_{n,l}=0$时表示任务在本地进行计算，$a_{n,l}=m$时表示任务卸载至第m台边缘服务器进行计算。由所有任务的数据量构成的数据矩阵$\boldsymbol{B}$为

$$\boldsymbol{B}=\begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,l} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,l} \\ \vdots & \vdots & & \vdots \\ b_{n,1} & b_{n,2} & \cdots & b_{n,l} \end{bmatrix}_{N\times L} \tag{2}$$

式中：当$b_{n,l}=0$时表示该任务不存在，$b_{n,l}>0$时表示该任务存在并具有相应数据量，且$Pr(b_{n,l}=0)=\theta$，$Pr(b_{n,l}>0)=1-\theta$。通过这种方式，每台机械设备可能拥有不同数量的有效任务，模拟了制造车间中设备间任务量存在差异的情形。

本文进行如下假设：

(1) 边缘服务器地理位置固定，与各机械设备之间的网络连接稳定。

(2) 在某一瞬时，设备层的每台设备任务数量为L，边缘层的每台服务器可以为任意一台设备提供计算服务，且能够同时进行多个任务的计算。

(3) 任务卸载至边缘服务器进行计算后，设备待机能耗忽略不计。

(4) 任务在边缘服务器计算完毕后，由于计算结果数据量很小，其回传至设备的时延与能耗均忽略不计。

1.2　时延模型

1.2.1　本地计算时延

若任务$D_{n,l}$在机械设备R_n上本地处理($a_{n,l}=0$)，本地计算时延$T_{l,n}^{\text{local}}$的定义如式(3)所示。其中，c表示CPU处理1 bit数据所需要消耗的CPU周期数。由于机械设备的CPU计算能力有限，不能承载过多的计算任务。因此，本文设置每台机械设备的CPU以正常频率运行时所能承载的最大任务数为Q_n，当设备当前任务总数Q_n^{now}大于Q_n时，CPU出现性能退化效应，调整后的机械设备CPU频率f_n'的计算公式如式(4)所示。

$$T_{l,n}^{\text{local}}=\frac{b_{n,l}\cdot c}{f_n'} \tag{3}$$

$$f_n'=\begin{cases} f_n & Q_n^{\text{now}}\leqslant Q_n \\ f_n\cdot\dfrac{Q_n}{Q_n^{\text{now}}} & Q_n^{\text{now}}>Q_n \end{cases} \tag{4}$$

1.2.2　边缘计算时延

若任务$D_{n,l}$卸载至边缘服务器S_m处理($a_{n,l}=m$)，其过程由传输和计算两个阶段组成。在传输阶段中，机械设备R_n需要通过上行链路将计算任务数据上传给边缘服务器S_m。系统的上行链路采用正交频分多址(Orthogonal frequency division multiple access，OFDMA)技术，根据香农公式的定义可以得出任务$D_{n,l}$从机械设备R_n卸载到边缘服务器S_m的传输速率V_n^m为

$$V_n^m=W\times\log_2\left(1+\frac{P_n^{\text{trans}}\times H_n^m}{W\times N_0}\right) \tag{5}$$

式中：W表示上行传输带宽，N_0表示噪声功率频谱密度。

任务$D_{n,l}$卸载至边缘服务器S_m时，传输时延$T_{n,l}^m$计算公式如式(6)所示，计算时延$T_{l,m}$计算公式如式(7)所示。本文考虑到边缘服务器的计算能力有限，不能无限制地向一台边缘服务器进行任务卸载。因此，本文设置每台边缘服务器的CPU以正常频率运行时所能承载的最大任务数Q_m，当卸载至某一台边缘服务器的任务总数Q_m^{now}大于Q_m时，CPU出现性能退化效应，调整后的边缘服务器CPU频率f_m'的计算公式如式(8)所示。

$$T_{n,l}^m=\frac{b_{n,l}}{V_n^m} \tag{6}$$

$$T_{l,m}=\frac{b_{n,l}\cdot c}{f_m'} \tag{7}$$

$$f_m'=\begin{cases} f_m & Q_m^{\text{now}}\leqslant Q_m \\ f_m\cdot\dfrac{Q_m}{Q_m^{\text{now}}} & Q_m^{\text{now}}>Q_m \end{cases} \tag{8}$$

综上，任务$D_{n,l}$卸载至边缘服务器S_m处理的总时延$T_{n,l}^{\text{edge}}$为

$$T_{n,l}^{\text{edge}}=T_{n,l}^m+T_{l,m} \tag{9}$$

1.3　能耗模型

(1) 本地计算能耗

任务$D_{n,l}$在机械设备R_n上本地计算时，其能耗$E_{l,n}^{\text{local}}$为

$$E_{l,n}^{\text{local}}=P_n^{\text{comp}}T_{l,n}^{\text{local}} \tag{10}$$

(2) 边缘计算能耗

任务$D_{n,l}$卸载至边缘服务器S_m计算时，其能耗分为传输能耗与计算能耗两部分。传输能耗$E_{n,l}^m$和计算能耗$E_{l,m}$计算公式分别为

$$E_{n,l}^m=P_n^{\text{trans}}T_{n,l}^m \tag{11}$$

$$E_{l,m}=P_m^{\text{comp}}T_{l,m} \tag{12}$$

综上，任务$D_{n,l}$卸载至边缘服务器S_m处理的总能耗$E_{n,l}^{\text{edge}}$计算公式为

$$E_{n,l}^{\text{edge}}=E_{n,l}^m+E_{l,m} \tag{13}$$

1.4 卸载问题模型

根据上述模型,对于N台机械设备中的$N\times L$个计算任务,它们的总时延T^{total}和总能耗E^{total}分别为

$$T^{\text{total}}=\sum_{n=1}^{N}\sum_{l=1}^{L}(1-\min(1,a_{n,l}))T_{l,n}^{\text{local}}+\min(1,a_{n,l})T_{n,l}^{\text{edge}} \tag{14}$$

$$E^{\text{total}}=\sum_{n=1}^{N}\sum_{l=1}^{L}(1-\min(1,a_{n,l}))E_{l,n}^{\text{local}}+\min(1,a_{n,l})E_{n,l}^{\text{edge}} \tag{15}$$

将系统的总代价F定义为任务计算时延与能耗的加权和,计算公式如式(16)所示。

$$F=\lambda T^{\text{total}}+(1-\lambda)E^{\text{total}} \tag{16}$$

式中:λ为时延权重因子,$(1-\lambda)$为能耗权重因子,$\lambda\in[0,1]$。λ越大,系统对时延的敏感程度越高,对能耗的敏感程度越低。反之,系统对时延的敏感程度越低,对能耗的敏感程度越高。λ可根据实际情况调整。

综上,卸载问题可描述为何种任务分配决策可使得系统的时延与能耗加权和最小,即以最小化系统总代价为优化目标。优化目标函数和约束条件分别为

$$\begin{aligned}&\text{P1: }\min_{A}\{F\}\\&\text{s.t. C1: }a_{n,l}\in\{0,1,2,\cdots,m,\cdots,M\}\\&\quad\text{C2: }\sum_{k=0}^{M}\delta(a_{n,l},k)=1\\&\quad\forall n\in\{1,2,\cdots,N\},\quad\forall l\in\{1,2,\cdots,L\}\end{aligned} \tag{17}$$

式中:C1表示每个任务的卸载决策$a_{n,l}$在取值范围内取整数,C2表示每个任务必须且只能在本地或某一台边缘服务器上执行,不可分割。

2 DCPSO算法

PSO算法是一种经典的启发式群体智能优化算法,广泛应用于任务分配、调度优化等领域。然而,传统PSO算法在复杂高维问题中容易陷入局部最优且收敛速度较慢。针对这些问题,本文提出DCPSO算法,通过改进初始化方法、优化粒子更新机制和引入变异操作,显著提升算法性能。DCPSO算法流程如图2所示。

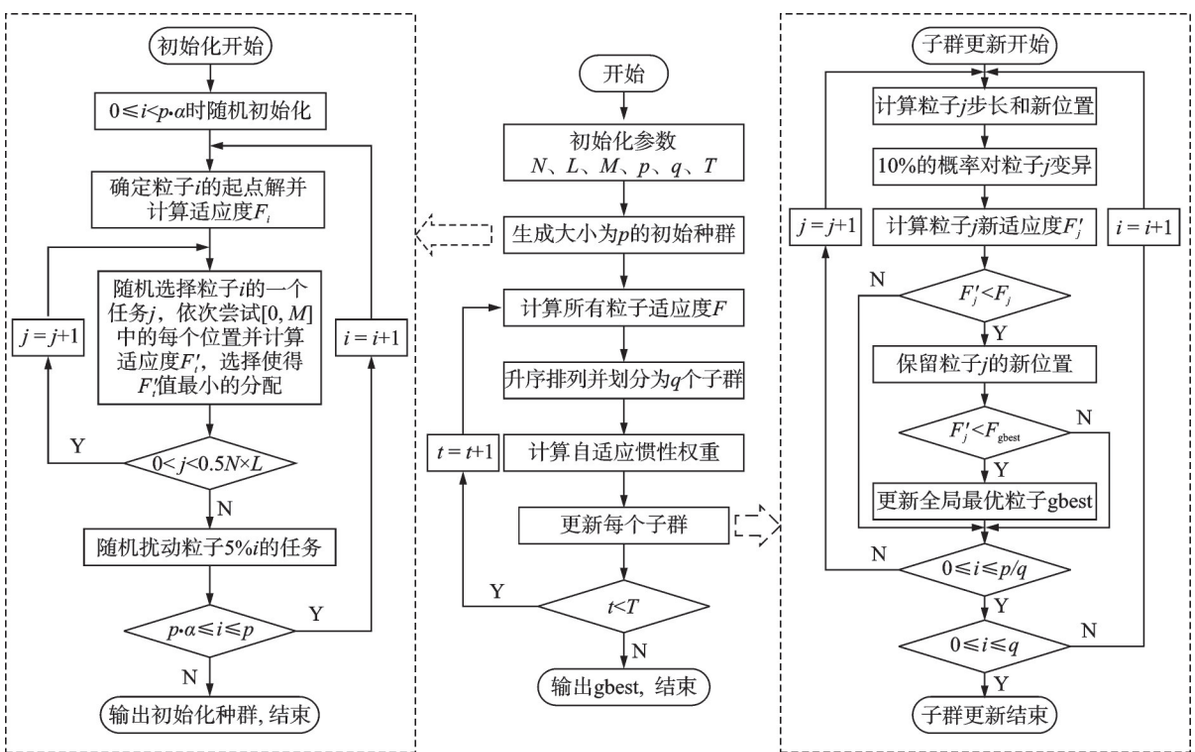

图2 DCPSO算法流程图

Fig.2 Flowchart of DCPSO algorithm

2.1 粒子编码

由于多边缘计算卸载问题是一种离散优化问题,粒子位置矩阵中的值代表任务的卸载位置,因此本文对粒子采用整数编码。粒子位置矩阵$\boldsymbol{A}_i$中的每个元素$a_{n,l}\in\{0,1,2,\cdots,m,\cdots,M\}$。在DCPSO算法的每次迭代中,粒子的位置都会进行更新,由于随机数的存在,会出现非整数位置值和超出合法范围位置值,因此采用式(18)进行离散数值转

换，并限制数值的合法范围。

$$A_i(n,l)=\begin{cases}\lfloor A_i(n,l)\rfloor & 0<A_i(n,l)\leqslant M\\ \max(0,\min(A_i(n,l),M)) & \text{其他}\end{cases} \tag{18}$$

2.2　混合初始化策略

对于启发式算法，初始化种群的质量直接影响着算法的迭代速度与最终解质量[25]。经典PSO算法通常采用随机初始化，初始解分布均匀但质量通常较差，难以贴近优化目标。针对这一问题，DCP-SO算法提出一种结合随机采样和基于适应度引导的贪心机制的混合初始化策略，在保证粒子多样性的同时，有效降低初始种群适应度。

在进行种群初始化时，对其中比例为α的粒子保留随机采样，确保解空间的广泛覆盖。对于另外比例为$(1-\alpha)$的粒子，采用基于适应度引导的贪心机制进行初始化。在确定起点解时，模拟负载均衡场景，将粒子$\boldsymbol{A}_i$的全部任务按$[0,M]$的顺序进行分配。$\boldsymbol{A}_i(n,l)$为

$$\boldsymbol{A}_i(n,l)=\begin{cases}U[0,M] & 1\leqslant i\leqslant p\cdot\alpha\\ (n\cdot L+l)\bmod(M+1) & p\cdot\alpha<i\leqslant p\end{cases} \tag{19}$$

式中：$\boldsymbol{A}_i(n,l)$表示第i个粒子的第n台设备中的第l个任务，p为种群大小，U为均匀随机整数。

确定粒子$\boldsymbol{A}_i$的起点解后进行局部搜索。随机选择$\boldsymbol{A}_i$中的$N\times L/2$个任务，将每个任务的分配位置依次修改为$[0,M]$，选择使得粒子$\boldsymbol{A}_i$适应度最低的分配位置作为该任务的候选解，计算公式为

$$\boldsymbol{A}_i(n,l)=\arg\min_s F(\boldsymbol{A}_i^{n,l=s})\quad s\in[0,M] \tag{20}$$

式中：$\boldsymbol{A}_i^{n,l=s}$表示$\boldsymbol{A}_i$在(n,l)处的分配位置为s。

局部搜索完成后，对粒子$\boldsymbol{A}_i$中5%的粒子进行随机扰动，从而得到粒子$\boldsymbol{A}_i$的最终解为

$$\boldsymbol{A}_i(n,l)=U[0,M]\quad i\in(p\cdot\alpha,p] \tag{21}$$

为了探究随机采样比例α对种群最优适应度和粒子多样性的影响，从而确定α合适的取值，本文在种群大小p、机械设备量N、单机任务量L和边缘服务器数量M分别为50、30、5和5的测试环境下进行仿真实验，并设置α取值范围为$[0,1]$，步长为0.1。实验结果统计种群最优适应度和粒子多样性随α增长的变化趋势。其中，粒子多样性采用平均汉明距离(Hamming distance，HD)进行评价，在本文中汉明距离$H(\boldsymbol{A}_i,\boldsymbol{A}_j)$指粒子$\boldsymbol{A}_i$与粒子$\boldsymbol{A}_j$位置矩阵的每个对应元素值不同的次数之和，计算公式如式(22)所示。包含p个粒子的粒子群的平均汉明距离H_{avg}计算公式如式(23)所示。实验结果如图3所示。

$$\begin{cases}H(\boldsymbol{A}_i,\boldsymbol{A}_j)=\sum\limits_{n=1}^{N}\sum\limits_{l=1}^{L}\delta(a_{n,l}^i,a_{n,l}^j)\\ \delta(a_{n,l}^i,a_{n,l}^j)=\begin{cases}1 & a_{n,l}^i\neq a_{n,l}^j\\ 0 & a_{n,l}^i=a_{n,l}^j\end{cases}\end{cases} \tag{22}$$

$$H_{\text{avg}}=\frac{2}{p(p-1)}\sum_{i=1}^{p-1}\sum_{j=i+1}^{p}H(\boldsymbol{A}_i,\boldsymbol{A}_j) \tag{23}$$

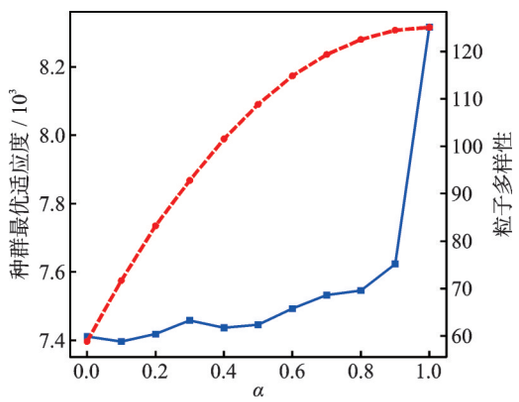

图3　随机采样比例对种群最优适应度和粒子多样性的影响

Fig.3　Effect of random sampling ratio on population optimal fitness and particle diversity

由图3中数据可知，在随机采样比例较低($0\leqslant\alpha<0.6$)的情况下，种群最优适应度维持在较低水平且上升缓慢，同时伴随着粒子多样性的快速增加。随着随机采样比例的提高($0.6\leqslant\alpha\leqslant1$)，种群最优适应度的增长速率逐渐提高，并在$\alpha=1$时达到最大值，而此时粒子多样性的增长幅度相对有限。因此，为了兼顾较低的种群最优适应度和较优的粒子多样性，本文确定随机采样比例α的值为0.5，即种群中50%的粒子采用随机采样策略，另50%的粒子采用基于适应度引导的贪心机制。

混合初始化策略的伪代码如算法1所示。

算法1　混合初始化策略

输入：种群大小p，设备量N，单机任务量L，边缘服务器量M，随机采样比例α。

输出：初始种群particles。

Begin:

(1)　for $i=1$ to $p\cdot\alpha$ do

(2)　　根据式(19)为粒子particles[i]的每个任务随机赋值

(3)　　fitness[i]$=F$(particles[i])

(4)　end for

(5)　for $i=p\cdot\alpha+1$ to p do

(6)　　根据式(19)确定粒子particles[i]的起点解

(7)　　for $j=1$ to $N\times L/2$ do

(8)　　　for $m=0$ to M do

(9) 根据式(20)对粒子particles[i]的起点解进行局部搜索得到候选解

(10) end for

(11) end for

(12) 根据式(21)随机扰动粒子particles[i]的候选解中5%的任务得到最终解

(13) fitness[i]=F(particles[i])

(14) end for

(15) best_particle=particles[argmin(fitness)]

End

2.3 动态子群协作更新机制

经典PSO算法中,所有粒子在全局搜索空间内搜索时会逐渐趋向于聚集在某个区域,种群多样性迅速降低,导致群体过早收敛到局部最优解。为了解决这一问题,本文在所提DCPSO算法中引入一种动态子群划分策略。设计将粒子群划分为多个子群,每个子群独立进行局部优化,并在满足某种条件时,子群之间进行信息融合。与此同时,在每个子群内部使用自适应粒子更新策略,通过自适应惯性权重与改进的粒子步长更新公式进行粒子位置更新。

DCPSO算法在每一次迭代前,根据每个粒子的适应度值$F(\boldsymbol{A}_i)$,将种群中的p个粒子进行升序排序,得到排序后的索引序列,如式(24)所示。然后,将排序后的粒子分配到q个子群,每个子群包含p/q个粒子,按式(25)所示的方式进行分组。

$$O=\arg\mathrm{sort}(F(\boldsymbol{A}_1),F(\boldsymbol{A}_2),\cdots,F(\boldsymbol{A}_p)) \tag{24}$$

$$G_i=\left\{O[i-1],O[i-1+q],\cdots,O\left[i-1+\left(\frac{p}{q}-1\right)q\right]\right\}\quad i=1,2,\cdots,q \tag{25}$$

式中G_i表示第i个子群。

通过上述升序排序和间隔分配的方式,使得每个子群均包含从最优到较差的粒子,确保子群内适应度分布的多样性,避免了单一子群内粒子性能过于集中,增强搜索能力。

完成子群划分后,在子群G_i内,对每一个粒子进行位置更新。当前粒子$\boldsymbol{A}_i$的位置更新步长$\boldsymbol{S}_i$综合考虑子群最优粒子$\boldsymbol{A}_{\mathrm{lbest}}$和全局最优粒子$\boldsymbol{A}_{\mathrm{gbest}}$,计算公式为

$$\boldsymbol{S}_i=\omega[r_1(\boldsymbol{A}_{\mathrm{lbest}}-\boldsymbol{A}_i)+r_2(\boldsymbol{A}_{\mathrm{gbest}}-\boldsymbol{A}_i)] \tag{26}$$

式中:$r_1,r_2\in[0,1]$为随机数,ω为自适应惯性权重。为了在迭代初期促进全局搜索而在迭代后期加强局部搜索能力,ω通过当前迭代次数进行自适应调整,计算公式为

$$\omega=1-\left(\frac{t}{T}\right)^5 \tag{27}$$

式中:t为当前迭代次数,T为最大迭代次数。

粒子更新步长$\boldsymbol{S}_i$确定后,第$t+1$代粒子位置$\boldsymbol{A}_i^{t+1}$通过在第t代粒子位置$\boldsymbol{A}_i^t$的基础上叠加步长$\boldsymbol{S}_i$得到,计算公式为

$$\boldsymbol{A}_i^{t+1}=\boldsymbol{A}_i^t+\boldsymbol{S}_i \tag{28}$$

动态子群协作更新机制伪代码如算法2所示。

算法2 动态子群协作更新机制

输入:初始种群particles,最大迭代次数T,种群大小p,子群数q。

输出:全局最优粒子gbest_particle。

Begin:

(1) for t=1 to T do

(2) 根据式(24)对种群进行排序

(3) 根据式(25)对种群进行子群划分得到q个子群,每个粒子记为sgparticles[i][j]

(4) 根据式(27)计算当前代的惯性权重ω

(5) for i=1 to q do

(6) for j=1 to p/q do

(7) 根据式(26)计算粒子sgparticles[i][j]的更新步长S_j

(8) 根据式(28)更新粒子sgparticles[i][j]的位置并记录为new_position

(9) if Random<0.1 then

(10) 根据算法3对new_position进行变异

(11) new_fitness=F(new_position)

(12) if new_fitness<sgfitness[i][j]then

(13) sgparticles[i][j]=new_position

(14) if new_fitness<gbest_fitness then

(15) gbest_particle=new_position

(16) end for

(17) end for

(18) end for

End

2.4 变异机制

在粒子完成位置更新后,为维持种群多样性,本文所提DCPSO算法引入变异机制对粒子的新位置$\boldsymbol{A}_i^{t+1}$进行扰动。对于第$t+1$代粒子的位置$\boldsymbol{A}_i^{t+1}$,设定变异概率σ为0.1,通过生成随机数r_3与σ进行比较以决定是否触发变异操作。若$r_3<\sigma$,

则在该粒子的位置矩阵中随机选取1～5个任务，对其卸载决策进行随机调整，从而生成一个新解$A_i^{t+1}{}'$，计算公式为

$$A_i^{t+1}{}'=\begin{cases}A_i^{t+1} & r_3\geqslant\sigma\\ A_i^{t+1}(n,l)=U[0,M] & k\in\mathbf{N},1\leqslant k\leqslant K,r_3<\sigma\end{cases}\tag{29}$$

式中：$K=U[1,5]$，$r_3\in[0,1]$为随机数，σ为变异概率。

变异机制伪代码如算法3所示。

算法3　变异机制

输入：粒子particle。

输出：新粒子new_particle。

Begin:

(1)　$K=$RandomInteger(1,5)

(2)　for $k=1$ to K do

(3)　$n=$RandomInteger(1,N)，$l=$RandomInteger(1,L)

(4)　particle[n,l]=RandomInteger(0,M)

(5)　end for

(6)　new_particle=particle

End

3　实验仿真与分析

为验证本文所提出的DCPSO算法的有效性，采用多维度评估方法和递进式实验设计，系统分析DCPSO算法的性能特征。首先，通过收敛性实验构建算法性能评估基准，与PSO、王泽等[26]提出的遗传-二进制粒子群(Genetic algorithm-binary particle swarm optimization，GA-BPSO)算法对比初始解质量、收敛速度与最终解质量，确立算法在单次优化中的理论优势。然后，通过稳定性实验的重复性测试，与上述两种算法对比鲁棒性特征，验证DCPSO算法优势的稳定性。最后，开展敏感性实验，通过构建不同规模的系统测试场景，将DCPSO算法与5种基线算法进行对比，揭示算法性能随问题复杂度变化的规律及优势的普适性。

在本节实验中，使用Python 3.11.3进行算法实现，系统运行环境为win10 64位操作系统，硬件配置为13th Gen Intel(R) Core(TM) i9-13900KF 3.00 GHz处理器，32 GB内存。

3.1　参数设置

实验参数的设置对算法性能和实验结果的影响很大，对于一些常用基本参数，本文参考相关文献进行设置[27]。实验参数如表1所示。

表1　实验参数

Table 1　Experimental parameters

参数	数值	参数	数值
f_n/GHz	0.5～2.5	W/MHz	2
P_n^{trans}/W	0.1～0.5	$W\times N_0$/W	1×10^{-9}
P_n^{comp}/W	10	λ	0.5
H_n^m	2×10^{-10}～2×10^{-6}	N	10～50
Q_n	5	L	2～10
f_m/GHz	5～10	M	5
P_m^{comp}/W	30	p	50
Q_m	20	q	10
$b_{n,t}$/Mbit	0或1～50	T	200
θ	0～0.1	α	0.5
c/(cycle•bit^{-1})	1 000	σ	0.1

3.2　基线算法

本文采用控制变量法研究DCPSO算法的性能表现，重点考察算法自身特性及系统规模变化对优化效果的影响机制。本文选择5种算法作为对比基线，包括3种常见规则策略[28]、经典粒子群算法以及一种近期的改进算法，具体算法详述如下。

(1) 全本地(LOCAL)：各机械设备的计算任务由其自身完成，不进行卸载。

(2) 全卸载(EDGE)：所有机械设备的计算任务均卸载至边缘服务器。

(3) 全随机(RANDOM)：每个任务的计算位置随机生成。

(4) PSO。

(5) GA-BPSO。

3.3　仿真实验及分析

3.3.1　算法收敛性验证

为了研究DCPSO算法的收敛特性，定量评估算法的性能优势，本实验在30台机械设备(单机任务量为5个)和5台边缘服务器的测试环境中，对PSO、GA-BPSO和DCPSO算法进行仿真实验，统计实验结果如图4及表2所示。

由实验结果可知，在初始解质量方面，PSO、GA-BPSO和DCPSO算法系统总代价分别为9 248.81、8 545.87和7 463.48，DCPSO相较于PSO

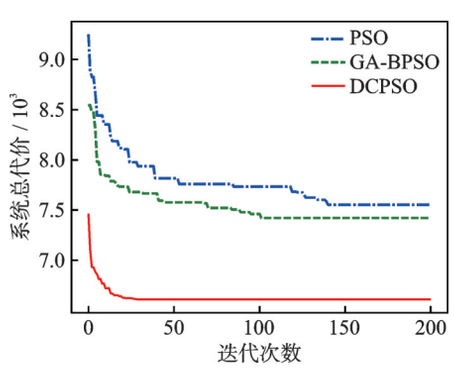

图4　算法收敛性分析

Fig.4　Analysis of algorithm convergence

表2 算法收敛性数据

Table 2 Data of algorithm convergence

指标	LOCAL	EDGE	RANDOM	PSO	GA-BPSO	DCPSO
初始解	—	—	—	9 248.81	8 545.87	7 463.48
收敛代数	—	—	—	142	103	29
最终解	12 459.04	9 613.36	8 621.88	7 554.31	7 422.76	6 609.67

与GA-BPSO分别降低了19.3%和12.7%,证实了混合初始化策略的有效性。在收敛速度方面,PSO算法于142代收敛,收敛速度较慢,且在迭代过程中多次出现停滞现象,印证了经典PSO算法局部搜索能力不足的理论缺陷。GA-BPSO算法收敛代数减少至103代,但仍存在类似的局部优化瓶颈。相比之下,DCPSO算法收敛于29代,收敛速度明显提升,且未出现前两种算法中的停滞现象。在最终解质量方面,DCPSO算法系统总代价为6 609.67,较PSO和GA-BPSO分别降低了12.5%和11.0%,其中PSO算法和GA-BPSO算法对应的系统总代价分别为7 554.31和7 422.76。此外,全本地模式、全卸载模式及全随机模式的系统总代价分别为12 459.04、9 613.36和8 621.88,均显著高于上述3种算法。上述实验结果充分验证了DCPSO算法在收敛性和优化能力方面的优越性。

3.3.2 算法稳定性验证

为进一步验证DCPSO算法性能优势的稳定性并系统评估其鲁棒性特征,本实验在30台机械设备(单机任务量为5个)和5台边缘服务器的测试环境中,分别对DCPSO算法及5种基线算法进行50次独立运行。统计实验结果如图5及表3所示。

由实验结果可知,在全本地模式下,由于所有任务均在设备本地执行,不存在资源竞争,系统总代价恒为12 459.04。全随机模式的系统总代价平均值为9 144.94,相较于全卸载模式的9 932.90更低,这是因为全随机模式在一定程度上利用了本地计算资源,减轻了边缘服务器的计算负载,从而降低了系统总代价。PSO、GA-BPSO和DCPSO算法系统总代价平均值分别为7 545.73、7 373.98和6 599.07,DCPSO相较于PSO与GA-BPSO分别降低了12.5%和10.5%,表明DCPSO算法的优化能力优势具备良好的可复现性。更重要的是,DCPSO算法系统总代价标准差仅为55.59,是PSO算法的41.7%,GA-BPSO算法的49.7%,全随机模式的14.8%,全边缘模式的18.4%,充分体现了DCPSO算法受随机因素的影响更小,鲁棒性更优。上述实验结果表明DCPSO算法在优化结果的稳定性方面具备显著优势。

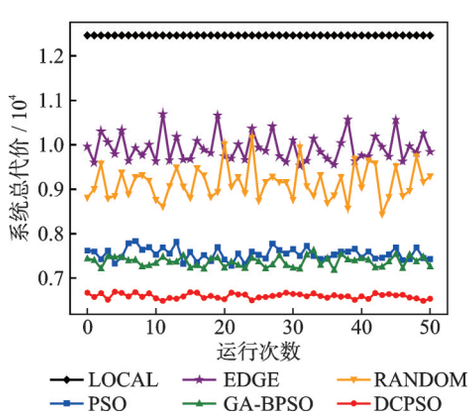

图5 算法稳定性分析

Fig.5 Analysis of algorithm stability

表3 算法稳定性数据

Table 3 Data of algorithm stability

指标	LOCAL	EDGE	RANDOM	PSO	GA-BPSO	DCPSO
最大值	—	10 691.27	10 152.96	7 834.10	7 629.61	6 695.50
最小值	—	9 532.96	8 435.41	7 279.80	7 181.29	6 490.99
平均值	12 459.04	9 932.90	9 144.94	7 545.73	7 373.98	6 599.07
标准差	—	301.69	376.48	133.37	111.79	55.59

3.3.3 算法敏感性分析

尽管上述实验表明了DCPSO算法具有良好的收敛性与稳定性,但为进一步地探索DCPSO算法在不同系统规模场景下的普适性,本节分别在不同设备量(10~50台)、单机任务量(2~10个)和数据量(10~50 Mbit)3个维度开展对比实验,每个实验仅改变单一维度参数,以揭示算法性能随问题复杂度的变化规律。

(1) 设备量的影响

为了测试设备量对于系统总代价的影响,本实验采用渐进式测试方案,将机械设备规模从10台逐步扩展至50台(单机任务量为5个),边缘服务器数量保持5台不变。实验结果如图6(a)所示。

分析实验结果可知,全本地计算模式由于设备

间无资源竞争且单机任务量恒定，系统总代价呈现理想的线性增长趋势。相比之下，其他5种卸载模式则因设备间存在资源竞争而表现出显著的非线性特征。此外，当总任务量不超过边缘服务器总最大正常负载（$M \cdot Q_m$）时，全卸载模式凭借边缘服务器的计算优势表现更优；而超过该阈值后，边缘服务器性能退化效应使得系统时延和能耗呈现指数级恶化，全随机模式因能有效利用本地计算资源而实现对全卸载模式的反超。

在所有测试场景中，3种启发式算法均表现出显著优势，尤其是DCPSO算法，其系统总代价相比PSO和GA-BPSO算法分别降低了6.7%～25.3%和4.8%～20.4%。此外，DCPSO的曲线增长速率相较次优算法平均下降了18.0%。这些结果不仅体现了启发式方法在多边缘卸载优化任务中的有效性，更突出展示了DCPSO在时延-能耗联合优化方面优越的规模适应性。尤为值得一提的是，在设备数量突破临界值后，DCPSO的性能衰减程度远低于其他算法，这一特性为实际工业场景中的规模扩展提供了技术保障。

（2）单机任务量的影响

为了测试单机任务量对于系统总代价的影响，本实验在固定30台机械设备和5台边缘服务器的测试环境下，通过逐步增加单机任务量（2～10个）构建了总任务量60～300个的渐进式测试场景。实验结果如图6（b）所示。

分析实验结果可知，随着系统总任务量的持续增长，6种算法均呈现出显著的非线性代价增长特征，这种变化趋势源于计算资源过载引发的性能退化效应。值得注意的是，当单机任务量超过4个/台（即总任务量超过边缘服务器总最大正常负载100个）时，系统性能呈现加速恶化态势，这一现象产生的主要原因是边缘服务器负载的饱和。

从全部测试结果来看，3种启发式算法均取得了较为显著的优化成效。其中，DCPSO在系统总代价方面较PSO和GA-BPSO分别降低了10.0%～30.5%与6.5%～22.3%。同时，DCPSO算法曲线的平均增长速率也较次优算法降低了21.5%。这些表现充分说明DCPSO不仅能在不同任务密度下保持较强的时延-能耗平衡能力，而且在高负载场景（单机任务量＞8个/台）下展现出更稳健的性能衰减曲线，为突发任务激增等实际场景的资源调度提供了可靠的解决方案。

（3）数据量的影响

为了测试每个任务的数据量对于系统总代价的影响，本实验在固定30台机械设备（单机任务量为5个）和5台边缘服务器的测试环境下，通过调控任务数据量（10～50 Mbit）构建了线性变化的测试场景。实验结果如图6（c）所示。

分析实验结果可知，系统总代价随数据量增长呈现线性上升趋势，这一现象源于两种不同的作用机制。对于本地计算任务而言，数据量增加直接导致低性能设备计算时延的显著提升。对于卸载至边缘的计算任务而言，数据量增加导致传输时延上升，而边缘服务器的高计算性能使计算时延影响相对较小，故传输时延是系统总代价增长的主导

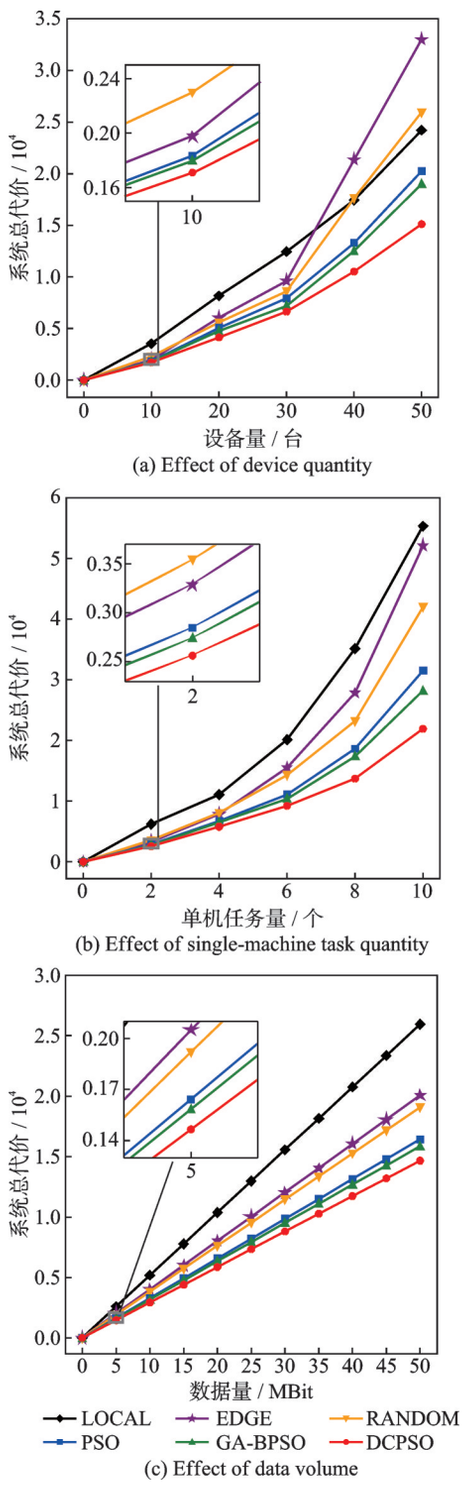

图6　算法敏感性分析

Fig.6　Analysis of algorithm sensitivity

因素。

综合各种数据量的实验结果,3种启发式算法均显示出较好的优化能力,DCPSO算法尤为突出,其系统总代价相较PSO和GA-BPSO分别降低10.7%与7.5%,且其系统总代价增长速率比次优算法降低7.5%。上述实验数据表明,DCPSO算法能够在不同的数据规模场景下有效实现边缘层与设备层资源的协同利用,进而达到更优的时延-能耗整体优化效果。

4 结 论

本文针对制造车间环境下的多边缘任务卸载优化问题,创新性地提出了DCPSO。在问题模型层面,通过构建基于多目标加权代价函数的系统优化模型,实现了计算任务时延与设备能耗的协同优化。在算法设计层面,DCPSO的核心创新体现在3个方面。首先,混合初始化策略创造性地将随机采样与基于适应度引导的贪心机制相结合,在维持种群多样性的同时显著提升了初始解的质量。其次,动态子群协作更新机制通过动态子群划分策略和自适应粒子更新规则,有效克服了传统PSO算法在复杂搜索空间中易出现的早熟收敛问题。最后,变异操作增强了算法跳出局部最优解的能力。实验验证表明,相较于现有基线算法,DCPSO在收敛速度、求解质量和鲁棒性等方面均展现出显著优势,并且在不同规模的系统场景中均有更好的性能表现。在未来的研究中,将针对DCPSO算法参数敏感性、场景普适性和混合初始化策略的计算复杂度以及理想假设与实际环境差异等局限性展开进一步优化。

参考文献:

[1] LIN Zhiwen, LIU Zhifeng, ZHANG Yueze, et al. Edge-fog-cloud hybrid collaborative computing solution with an improved parallel evolutionary strategy for enhancing tasks offloading efficiency in intelligent manufacturing workshops[J]. Journal of Intelligent Manufacturing, 2025, 36(7): 4635-4662.

[2] 郭永安, 王宇翱, 周沂, 等. 边缘网络下多无人机协同计算和资源分配联合优化策略[J]. 南京航空航天大学学报, 2023, 55(5): 757-767.
GUO Yongan, WANG Yuao, ZHOU Yi, et al. Multi-UAV collaborative computing and resource allocation joint optimization strategy in edge networks[J]. Journal of Nanjing University of Aeronautics & Astronautics, 2023, 55(5): 757-767.

[3] HE Yi, ZHANG Yanzhong, WU Che, et al. Architecture design and application of IIoT platform in automobile manufacturing based on microservices and deep learning techniques[J]. IEEE Access, 2024, 12: 166834-166842.

[4] LIN Chuncheng, DENG Derjiunn, HSIEH Litsung, et al. Optimal deployment of private 5G multi-access edge computing systems at smart factories: Using hybrid crow search algorithm[J]. Journal of Network and Computer Applications, 2024, 227: 103906.

[5] ASAAD R R, HANI A A, SALLOW A B, et al. A development of edge computing method in integration with IOT system for optimizing and to produce energy efficiency system[C]//Proceedings of the 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE). Greater Noida, India: IEEE, 2024: 835-840.

[6] ALMULIFI A, KURDI H. The role of fog device density in IoT-fog-cloud systems[J]. Procedia Computer Science, 2024, 241: 242-247.

[7] 张明, 付乐, 王海峰. 面向边缘计算的并发数据流接转控制模型[J]. 计算机应用, 2024, 44(12): 3876-3883.
ZHANG Ming, FU Le, WANG Haifeng. Relay control model for concurrent data flow in edge computing [J]. Journal of Computer Applications, 2024, 44(12): 3876-3883.

[8] GE Haibo, GENG JiaJun, AN Yu, et al. Research on collaborative computational offload strategy based on improved ant colony algorithm in edge computing [C]//Proceedings of the 5th International Conference on Natural Language Processing (ICNLP). Guangzhou, China: IEEE, 2023: 486-490.

[9] 尼俊红, 臧云. 异构边缘云架构下的多任务卸载算法[J]. 哈尔滨工程大学学报, 2024, 45(4): 800-807.
NI Junhong, ZANG Yun. Multitask offloading algorithm under heterogeneous edge cloud architecture[J]. Journal of Harbin Engineering University, 2024, 45(4): 800-807.

[10] YOU Qian, TANG Bing. Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things[J]. Journal of Cloud Computing, 2021, 10(1): 41.

[11] PENG Qixin, CHEN Xinde, HUANG Yujing, et al. Particle swarm optimization-based task migration in mobile-edge cloud computing[C]//Proceedings of 2023 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics). Danzhou, China: IEEE, 2023: 616-623.

[12] 李金，樊腾飞，高红亮，等. 边缘计算网络中多核任务卸载调度和资源适配研究[J]. 兵工自动化，2025，44(3)：29-34.
LI Jin，FAN Tengfei，GAO Hongliang，et al. Research on multi-core task offload scheduling and resource adaptation in edge computing networks[J]. Ordnance Industry Automation，2025，44(3)：29-34.

[13] LI Longmei，NIU Jun，YANG Xiaotong. Resource optimization allocation method of power grid digital transformation based on cloud edge collaboration[C]//Proceedings of the 4th International Conference on Smart Grid and Energy Engineering (SGEE). Zhengzhou，China：IEEE，2023：62-65.

[14] BEY M，KUILA P，NAIK B B，et al. Quantum-inspired particle swarm optimization for efficient IoT service placement in edge computing systems[J]. Expert Systems with Applications，2024，236：121270.

[15] VAN ZYL J P ，ENGELBRECHT A P. Set-based particle swarm optimisation：A review[J]. Mathematics，2023，11(13)：2980.

[16] FREITAS D，LOPES L G，MORGADO-DIAS F. Particle swarm optimisation：A historical review up to the current developments[J]. Entropy，2020，22(3)：362.

[17] LIU Xuanyan，YAN Rui，KIM J Y，et al. MPSO：An optimization algorithm for task offloading in cloud-edge aggregated computing scenarios for autonomous driving[J]. Mobile Networks and Applications，2024，30：702-716.

[18] ZHANG Degan，SUN Guixiang，ZHANG Jie，et al. Offloading approach for mobile edge computing based on chaotic quantum particle swarm optimization strategy[J]. Journal of Ambient Intelligence and Humanized Computing，2023，14(10)：14333-14347.

[19] 申秀雨，姬伟峰. 考虑安全的边-云协同计算卸载成本优化[J]. 信息网络安全，2024，24(7)：1110-1121.
SHEN Xiuyu，JI Weifeng. Optimization of cost of edge-cloud collaborative computing offloading considering security[J]. Netinfo Security，2024，24(7)：1110-1121.

[20] VELRAJAN S，CERONMANI SHARMILA V. QoS-aware service migration in multi-access edge compute using closed-loop adaptive particle swarm optimization algorithm[J]. Journal of Network and Systems Management，2023，31(1)：17.

[21] 吴波，龙廷艳，万良，等. MEC中基于改进粒子群算法的任务卸载策略[J]. 计算机工程，2026，52(4)：327-338.
WU Bo，LONG Tingyan，WAN Liang，et al. Task offloading strategies based on improved particle swarm algorithms in MEC[J]. Computer Engineering，2026，52(4)：327-338.

[22] ZHENG Tao，YANG Bin. Multi-server cooperative offloading strategy for dependent tasks based on improved genetic algorithm[C]//Proceedings of Advanced Intelligent Computing Technology and Applications. Singapore：Springer，2024：3-14.

[23] BOCCELLA A R，CENTOBELLI P，CERCHIONE R，et al. Evaluating centralized and heterarchical control of smart manufacturing systems in the era of industry 4.0[J]. Applied Sciences，2020，10(3)：755.

[24] HÄCKEL B，HÄNSCH F，HERTEL M，et al. Assessing IT availability risks in smart factory networks[J]. Business Research，2019，12(2)：523-558.

[25] FÉ J，CORREIA S D，TOMIC S，et al. Swarm optimization for energy-based acoustic source localization：A comprehensive study[J]. Sensors，2022，22(5)：1894.

[26] 王泽，郭荣佐. 基于GA-BPSO算法的MEC卸载决策[J]. 计算机工程与设计，2023，44(7)：2054-2061.
WANG Ze，GUO Rongzuo. MEC offloading decision based on GA-BPSO algorithm[J]. Computer Engineering and Design，2023，44(7)：2054-2061.

[27] 周天清，曾新亮，胡海琴. 基于混合粒子群算法的计算卸载成本优化[J]. 电子与信息学报，2022，44(9)：3065-3074.
ZHOU Tianqing，ZENG Xinliang，HU Haiqin. Computation offloading cost optimization based on hybrid particle swarm optimization algorithm[J]. Journal of Electronics & Information Technology，2022，44(9)：3065-3074.

[28] 蒋鹏，富爽，丁晨阳. 多设备多任务场景下基于改进粒子群优化的计算卸载策略[J]. 黑龙江八一农垦大学学报，2024，36(1)：98-107.
JIANG Peng，FU Shuang，DING Chenyang. Computation offloading strategy based on improved particle swarm optimization in multi-user and multi-task scenarios[J]. Journal of Heilongjiang Bayi Agricultural University，2024，36(1)：98-107.

（编辑：陈珺）