计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
2025,34(2):264−271 [doi: 10.15888/j.cnki.csa.009748] [CSTR: 32024.14.csa.009748]

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

无人机辅助 MEC 中的依赖性任务卸载①

李贵勇, 廖福建, 田　旭

(重庆邮电大学 通信与信息工程学院, 重庆 400065)
通信作者: 廖福建, E-mail: 907685071@qq.com

摘　要: 在任务计算密集型和延迟敏感型的场景下, 无人机辅助的移动边缘计算由于其高机动性和放置成本低的特点而被广泛研究. 然而, 无人机的能耗限制导致其无法长时间工作并且卸载任务内的不同模块往往存在着依赖关系. 针对这种情况, 以有向无环图 (direct acyclic graph, DAG) 为基础对任务内部模块的依赖关系进行建模, 综合考虑系统时延和能耗的影响, 以最小化系统成本为优化目标得到最优的卸载策略. 为了解决这一优化问题, 提出了一种基于亚群、高斯变异和反向学习的二进制灰狼优化算法 (binary grey wolf optimization algorithm based on subpopulation, Gaussian mutation, and reverse learning, BGWOSGR). 仿真结果表明, 所提出算法计算出的系统成本比其他 4 种对比方法分别降低了约 19%、27%、16%、13%, 并且收敛速度更快.

关键词: 移动边缘计算; 无人机; 任务卸载; 依赖性任务; 二进制灰狼优化算法

引用格式: 李贵勇,廖福建,田旭.无人机辅助 MEC 中的依赖性任务卸载.计算机系统应用,2025,34(2):264–271. http://www.c-s-a.org.cn/1003-3254/9748.html

Dependent Task Offloading in Mobile Edge Computing Assisted by Unmanned Aerial Vehicle

LI Gui-Yong, LIAO Fu-Jian, TIAN Xu

(School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China)

Abstract: In computation-intensive and latency-sensitive tasks, unmanned aerial vehicle (UAV)-assisted mobile edge computing has been extensively studied due to its high mobility and low deployment costs. However, the energy consumption of UAVs limits their ability to work for extended periods, and there are often dependencies among different modules within offloading tasks. To address these issues, directed acyclic graph (DAG) is utilized to model the dependencies among internal modules of tasks. Considering the impacts of system latency and energy consumption, an optimal offloading strategy is derived to minimize system costs. To achieve optimization, a binary grey wolf optimization algorithm based on subpopulation, Gaussian mutation, and reverse learning (BGWOSGR) is proposed. Simulation results show that the proposed algorithm reduces system costs by around 19%, 27%, 16%, and 13% compared to four other methods, with a faster convergence speed.

Key words: mobile edge computing (MEC); unmanned aerial vehicle (UAV); task offloading; dependent task; binary grey wolf optimization algorithm

随着 5G 技术的不断发展, 用户终端上运行的计算密集型程序, 如线上游戏、虚拟现实、面部识别等, 对低延迟和高性能的需求日益增长. 为了满足这些应用的计算需求, 进一步提升用户的服务质量 (quality of service, QoS)[1]. 移动边缘计算 (mobile edge computing, MEC)[2]作为一种新型计算模式应运而生, 它通过在边

① 基金项目: 重庆市自然科学基金创新发展联合基金 (中国星网) (CSTB2023NSCQ-LZX0114); 重庆市自然科学基金面上项目 (cstc2021jcyj-msxmX0454)
收稿时间: 2024-07-05; 修改时间: 2024-07-25; 采用时间: 2024-08-01; csa 在线出版时间: 2024-12-06
CNKI 网络首发时间: 2024-12-06

缘服务器上处理用户终端任务的卸载请求, 实现高效的数据处理. 然而, 传统的基础设施型 MEC 存在部署成本高昂的问题, 而且在缺乏基础设施的农村地区、大型场所的高流量期间或发生自然灾害时的地区可能无法正常满足用户的计算需求. 近年来, 无人机 (unmanned aerial vehicle, UAV) 因其卓越的机动性和灵活部署能力而备受瞩目, 已广泛应用于多个领域[3]. 在沙漠、山区等偏远区域, 无人机可作为空中通信枢纽, 与卫星、地面网络融合, 实现全球空天地一体化覆盖. 在自然灾害发生, 地面基站受损的情况下, 无人机可作为应急通信设施支持救援. 在具有密集人群的热点场景, 无人机可增强或补充地面网络. 无人机不仅能提升 MEC 的计算能力, 还能扩展其覆盖范围, 为用户提供更加稳定可靠的服务[4].

无人机辅助的移动边缘计算系统相较于传统固定边缘服务器位置的 MEC 系统, 它提供了更为高效、迅速的计算服务. 然而, 实际的应用上, 这项技术仍面临着诸多挑战. 其中一个问题是, 用户任务往往由多个依赖模块组成, 这些模块之间存在明确的顺序依赖关系[5]. 这意味着, 任何一个模块都需要在其所有前置模块完成之后才能启动执行, 这就增加了任务的计算时延和求解最优卸载方案的难度. 另一个核心问题是支持 MEC 的无人机在能耗方面存在限制, 与云服务器和传统 MEC 系统相比, 其能源储备更为有限. 因此, 不合理的卸载决策不仅会导致任务完成时间的延长, 还会显著增加无人机的电池能量消耗[6]. 为了充分发挥无人机辅助 MEC 系统的优势, 本文计划探索出最佳的卸载策略, 以在有限的能源条件下, 实现任务延迟和能耗的最小化.

Yan 等[7]提出了一种多用户单 MEC 服务器的系统模型, 设计出一种二等分的搜索方法以降低系统的时延和能耗的加权和. Yuan 等[8]将计算卸载问题解耦为资源规划问题和卸载策略问题. 使用基于任务卸载率和资源分配率的博弈论方法设计了计算卸载方案, 减少了高能耗和不确定延迟. Lv 等[9]提出了一种启发式算法, 这种任务卸载算法能够预测卸载决策的影响. 该算法在减少时间和能耗方面具有显著效果. Li 等[10]针对单服务器场景提出了一种轻量级且高效的卸载决策. 此决策可以按顺序调度多用户任务, 从而使任务的平均计算延迟最小. Bai 等[11]为了解决任务分布不均匀, 无法满足实时性要求的问题, 提出了一种使用凸近似法简化原有问题并利用李雅普诺夫优化做出决策的协作卸载算法. 仿真表明该算法显著减少了任务的完成延迟. Fan 等[12]基于每个任务的最大容忍完成时间的约束下以最小化所有任务的总成本为目标, 提出了一种启发式算法来解决依赖性任务的卸载问题. Zhao 等[13]考虑了单个边缘服务器和多个用户的场景下依赖性任务的卸载问题, 提出了一种启发式算法来优化系统时延和用户设备的能耗. Xu 等[14]通过假设已知的无人机位置, 通过获得博弈模型的均衡解来解决服务定价和任务卸载子问题, 并且采用遗传算法求解无人机定位子问题. 仿真结果表明, 该算法可以带来更高的无人机卸载收益并降低用户延迟. Zhou 等[15]将无人机与用户之间的交互建模为 Stackelberg 博弈, 并且设计了一种基于梯度的动态迭代搜索方法来获取卸载方案的近似最优解. Wang 等[16]提出一种太赫兹波段的无人机辅助计算卸载架构, 综合考虑了无人机系统的服务质量和资源约束, 提出了一种双重深度 Q 学习算法以获得卸载方案的近似最优解.

综上所述, 由于依赖性任务的卸载问题较复杂, 在以往的研究中考虑任务间依赖约束的工作只有较少的一部分, 且大多只考虑单一用户或者单一 MEC 服务器的场景. 在此背景下, 本文对无人机辅助 MEC 场景下多用户多服务器中具有依赖关系的用户任务如何进行卸载的问题进行分析研究.

本文的主要贡献如下.

1) 在基于无人机辅助边缘服务器卸载的场景下, 构建了网络模型、任务依赖模型、计算模型, 采用加权法将时延、能耗的联合优化问题转化为对系统成本的单一优化问题, 通过尽可能地减少系统成本来获取最优的任务卸载方案.

2) 提出了一种 BGWOSGR 算法, 引入动态双亚群的策略, 使得两个亚群可以分别发挥局部搜索和全局搜索的优势, 在此基础上加入了高斯变异和反向学习策略, 保持了算法迭代过程中的种群多样性以尽可能地避免局部收敛问题.

3) 在 Matlab 环境上对本文提出的算法进行仿真, 仿真结果表明, 与参考的基准算法相比, 该算法收敛性更好, 并且可以有效地降低系统的时延和能耗.

1 系统模型

1.1 网络模型

无人机辅助 MEC 系统的网络模型如图 1 所示. 该

系统场景包括N个地面用户和M个配备 MEC 服务器在用户区域上方一定高度的无人机.

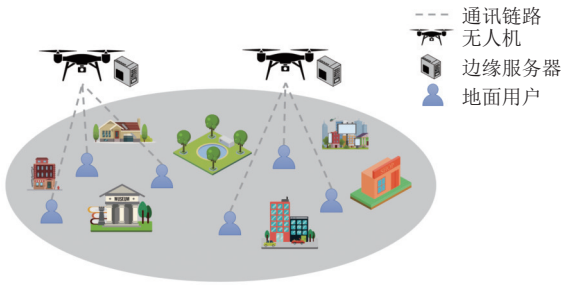

图 1 无人机辅助的 MEC 系统网络模型

假设无人机均飞行在某高度下的多个最优位置, 用户和无人机通过无线方式进行通信, 并且在无人机范围内的用户可以选择是否将计算任务卸载到指定的无人机上进行处理.

地面用户n的位置可以用三维坐标$(x_n,y_n,0)$表示, 无人机m的位置使用(x_m,y_m,H)进行表示. 本文考虑用户与无人机进行通信时的相对位置不变, 用户n和无人机m之间的信道增益$G_{n,m}$可以表示为:

$$G_{n,m}=\frac{\beta_0}{d_{n,m}^2}=\frac{\beta_0}{(x_n-x_m)^2+(y_n-y_m)^2+H^2} \tag{1}$$

其中, $d_{n,m}$表示用户n和无人机m之间的距离, β_0表示$d_{n,m}=1$时的信道功率的增益.

因此, 通过香农定理可知, 用户n将任务数据上传到无人机m的数据传输的通信速率可以表示为:

$$r_{n,m}=B\log_2\left(1+\frac{P_nG_{n,m}}{N_0B}\right) \tag{2}$$

其中, B为信道带宽, P_n为用户n终端的发射功率, N_0为无线信道中的噪声功率谱密度.

1.2 任务依赖模型

现有的计算密集型任务通常由多个相互依赖的模块组成, 例如处理虚拟现实相关的程序任务, 完成该任务主要包括环境建模、场景渲染、交互设计、用户输入捕获、实时反馈、系统优化这 6 个关键步骤, 而且各个步骤之间存在着数据依赖.

因此, 本文将同一任务的不同模块间的依赖关系建模为一个 DAG, 表示为$G=(P,E)$, 如图 2 所示.

$P=\{P_1,P_2,\cdots,P_K\}$表示组成任务所需要的所有模块集合[17], K表示为模块的个数, E表示边的集合. 每条边$e_{i,j}\ (1\leqslant i<j\leqslant K)$代表模块$i$和$j$之间的依赖关系, 即如果模块$j$需要模块$i$完成之后才能执行, 称模块$i$为$j$的前驱任务, 一个模块只有在其所有前驱模块完成后才能开始执行. 假设每个用户同一时隙有且仅有一个任务需要进行计算处理, 设R_n为用户n产生的计算任务, 将R_n中的第i个模块P_i^n相关信息定义为三元组$P_i^n=\{D_i^n,C_i^n,T_{i,\max}^n\}$. 其中, D_i^n代表模块数据量的大小, C_i^n代表处理D_i^n大小的数据所需要的 CPU 周期数, $T_{i,\max}^n$表示处理模块P_i^n可容忍的最大时延.

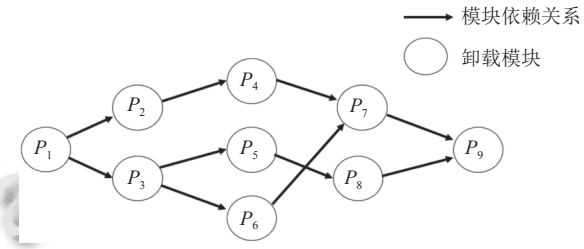

图 2 任务模块依赖图

定义 1. 若P_i^n是P_j^n的前驱模块, 则$e_{i,j}^n$为 1, 否则为 0, 即:

$$e_{i,j}^n=\begin{cases}1, & P_i^n\text{是}P_j^n\text{的前驱模块}\\ 0, & \text{其他情况}\end{cases} \tag{3}$$

定义 2. 卸载决策变量X_i^n表示模块P_i^m的卸载方式:

$$X_i^n=\begin{cases}1, & \text{卸载到无人机}\\ 0, & \text{数据本地计算}\end{cases} \tag{4}$$

模块的卸载方式本文考虑二进制卸载: $X_i^n=1$表示用户n产生的模块P_i^n卸载到无人机上; $X_i^n=0$表示模块P_i^n进行本地计算, 不进行卸载.

1.3 计算模型

1.3.1 本地计算

当用户对任务模块进行本地计算时, 假设本地的计算速率为f_n^{loc}, 则模块P_i^n的本地的计算时延表示为:

$$T_i^{n,\text{loc}}=\frac{D_i^nC_i^n}{f_n^{\text{loc}}} \tag{5}$$

模块P_i^n的本地计算的能耗为:

$$E_i^{n,\text{loc}}=\alpha_n\cdot\left(f_n^{\text{loc}}\right)^2D_i^nC_i^n \tag{6}$$

其中, α_n为用户n的本地设备的能耗系数, f_n^{loc}为用户n的本地设备的 CPU 频率.

1.3.2 边缘计算

无人机辅助的 MEC 网络中, 用户可以将计算任务卸载到无人机上进行计算. 边缘计算时延包含计算时延、数据上传时延和处理后数据的回传时延. 考虑通常情况下处理后数据的大小远小于上传数据的大小, 因此可以忽略数据的回传时延.

在边缘计算的场景中, 用户n将模块P_i^n卸载到无人机m的总时延为:

$$T_i^{n,m,\mathrm{up}} = \frac{C_i^n}{r_{n,m}} + \frac{C_i^n D_i^n}{f_m} \tag{7}$$

该场景下的能耗主要包含用户上传数据的传输能耗和无人机的计算能耗, 可以表示为:

$$E_i^{n,m,\mathrm{up}} = P_n \cdot \frac{C_i^n}{r_{n,m}} + \alpha_m \cdot (f_m)^2 D_i^n C_i^n \tag{8}$$

其中, α_m为无人机m的能耗系数, f_m为无人机m的 CPU 频率.

1.4 问题描述

由于每个任务可以选择在本地计算或卸载至无人机上进行计算, 则用户n完成模块P_i^n的计算时延为:

$$T_i^{n,\mathrm{com}} = (1 - X_i^n) \cdot T_i^{n,\mathrm{loc}} + X_i^n \cdot T_i^{n,m,\mathrm{up}} \tag{9}$$

由图 2 可知, 任务的总完成时间包含各个模块的执行时间和不同模块之间的依赖关系可能导致的等待时间. 因此, 用户n的任务总完成时间等价于最后一个模块完成的时间, 而且模块i的完成时间取决于其前驱模块的完成时间, 两者关系如下:

$$T_i^n = \begin{cases} \max\limits_{j \in pre(i)} \{e_{j,i}^n \cdot T_j^{n,\mathrm{com}}\} + T_i^{n,\mathrm{com}}, & pre(i) \neq \varnothing \\ T_i^{n,\mathrm{com}}, & pre(i) = \varnothing \end{cases} \tag{10}$$

其中, $pre(i)$为模块P_i^n的前驱模块集合. 用户n任务R_n的总时延可以表示为:

$$T_n = \max_i \{T_i^n\} \tag{11}$$

用户n的任务总能耗可以表示为:

$$E_n = \sum_{i=1}^{K} ((1 - X_n) \cdot E_i^{n,\mathrm{loc}} + X_n \cdot E_i^{n,m,\mathrm{up}}) \tag{12}$$

其中, $X_n = (X_1^n, X_2^n, \cdots, X_i^n)$为用户$n$任务中所有模块的卸载方案.

本文优化目标为任务执行时延和能耗的加权和[18], 用户n的任务执行成本可以表示为:

$$Y_n = \gamma \cdot T_n + (1 - \gamma) \cdot E_n \tag{13}$$

其中, γ为时延权重因子, $(1-\gamma)$为能耗权重因子, $\gamma \in [0,1]$. 可以根据不同的场景设定不同的权重因子, 例如在时延敏感场景, 可以将γ设为较大的值, 在能耗敏感场景, γ可以将设为较小的值.

除此之外, 还需要定义用户n的任务卸载到哪一个无人机上, 因此使用式 (14) 来表示用户n和卸载无人机m的关系:

$$Z_{n,m} = \begin{cases} 1, & \text{用户 } n \text{ 需要卸载到无人机 } m \\ 0, & \text{其他情况} \end{cases} \tag{14}$$

综上所述, 本文将系统中用户任务中所有模块的卸载决策作为优化问题, 以最小化任务的执行成本作为优化目标. 因此可以描述为如下的优化问题U:

$$\begin{cases} U: \min \sum\limits_{n=1}^{N} Y_n(X_n) \\ \text{s.t.} \begin{cases} C1: X_i^n \in \{0,1\} \\ C2: \sum\limits_{m=1}^{M} Z_{n,m} \leqslant 1 \\ C3: T_i^n \leqslant T_{i,\max}^n \end{cases} \end{cases} \tag{15}$$

其中, $C1$表示任务模块只能进行本地计算或者全部卸载至无人机上进行计算. $C2$表示用户n只能选择将任务卸载到最多一个无人机上执行. $C3$表示任务模块的计算时延不允许超过其最大容忍时延.

2 算法设计

2.1 灰狼优化算法

灰狼优化算法 (grey wolf optimization, GWO) 是 Mirjalili 等人于 2014 年提出的[19]. GWO 算法具有全局搜索能力强、原理简单、参数少等特点, 广泛用于求解特征选择、参数寻优等问题. 所以本文采用 GWO 算法用于寻找 MEC 卸载决策问题的最优近似解. 灰狼优化算法中的解是灰狼个体的位置, 适应度值前 3 的解分别被视作领导狼群的α狼、β狼和δ狼. 剩余适应度较低的解, 被视为ω狼. GWO 算法通过α狼、β狼和δ狼模拟狼群的狩猎行为来指导ω狼进行搜索, 能够在复杂的搜索空间中有效地寻找问题的最优解. 灰狼种群的狩猎过程通常包括包围猎物、追捕猎物和攻击猎物等步骤, 这些步骤在 GWO 算法中被转化为数学公式, 用于指导搜索过程, 公式大致如下:

$$D = \left| C \cdot X_p(t) - X(t) \right| \tag{16}$$

$$X(t+1) = X_p(t) - A \cdot D \tag{17}$$

$$A = 2 \cdot a \cdot r_1 - a \tag{18}$$

$$C = 2 \cdot r_2 \tag{19}$$

$$a = 2 - \frac{2 \cdot t}{T} \tag{20}$$

其中, t和T分别为算法当前迭代次数和最大迭代次数; $X(t)$表示灰狼位置; $X_p(t)$为猎物位置; A和C为系数; r_1,

r_2为[0, 1]之间的随机数; a为收敛因子.

灰狼个体的位置更新公式为:

$$\begin{cases} D_\alpha = |C_1 \cdot X_\alpha(t) - X(t)| \\ D_\beta = |C_2 \cdot X_\beta(t) - X(t)| \\ D_\delta = |C_3 \cdot X_\delta(t) - X(t)| \end{cases} \tag{21}$$

$$\begin{cases} X_1 = X_\alpha - A_1 \cdot D_\alpha \\ X_2 = X_\beta - A_2 \cdot D_\beta \\ X_3 = X_\delta - A_3 \cdot D_\delta \end{cases} \tag{22}$$

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \tag{23}$$

其中, 式 (21) 中的D_α、D_β、D_δ表示灰狼个体到α狼, β狼和δ狼之间的距离. 式 (22) 中的X_1、X_2、X_3表示灰狼个体向α狼、β狼和δ狼前进的步长和方向. 最终, 算法根据式 (23) 得到每只灰狼的更新后的位置.

2.2 改进二进制灰狼优化算法

本文求解的任务卸载决策问题本质上是一个二元选择问题, 而传统的灰狼优化算法的搜索空间是连续的, 所以需要对连续的灰狼位置做二进制的映射处理, 使用 *Sigmoid* 函数将灰狼位置映射到[0, 1]区间内:

$$Sigmoid[X(t+1)] = \frac{1}{1 + \mathrm{e}^{-10\left[\left(\frac{X_1+X_2+X_3}{3}\right)-0.5\right]}} \tag{24}$$

$$x_d(t+1) = \begin{cases} 1, & Sigmoid[X(t+1)] > rand \\ 0, & \text{其他} \end{cases} \tag{25}$$

其中, $rand$是[0,1]内均匀分布的随机数; $x_d(t+1)$是在d维中灰狼迭代后更新的位置.

2.2.1 动态双亚群策略

为了增强灰狼优化算法的全局搜索能力, 本文引入了广泛应用于算法优化的亚群策略[20], 它通过将种群分割成多个子群 (亚群), 每个亚群专注于各自区域内的搜索和开发, 从而大幅增强了算法的全局搜索能力. 然而, 根据“没有免费的午餐”定理[21], 种群的分割间接地削弱了算法的局部搜索能力. 为了弥补这一缺陷, 本文使用了一种动态双亚群策略. 在算法的每次迭代开始时, 该策略会根据灰狼个体适应度的大小将灰狼种群划分为两个亚群. 其中亚群 1 负责局部搜索, 亚群 2 专注于全局搜索. 在灰狼优化算法中, 系数A的大小影响着算法局部搜索和全局搜索的能力. 由式 (18) 可知, A的绝对值会在$[0,a]$范围变化. 当A的绝对值小于 1 的时候, 灰狼个体会进行精细的局部搜索以更准确地定位猎物, 而当A的绝对值大于 1 的时候, 灰狼个体会向远离猎物的方向进行移动, 扩大搜索范围进行全局搜索.

考虑到A的值会因为a的变化而变化, 本文通过改变亚群 1 和亚群 2 的a的公式来对不同亚群实现不同的搜索方式, 并将原先线性变化的a值改为非线性变化.

亚群 1 中a的公式为:

$$a = 1 - \frac{\mathrm{e}^{\frac{t}{T}} - 1}{\mathrm{e} - 1} \tag{26}$$

亚群 2 中a的公式为:

$$a = 2 - \left(\frac{t}{T}\right)^2 \tag{27}$$

在动态双亚群策略中, 每次迭代的开始阶段, 种群会动态地分为两个亚群. 亚群 1 的主要任务是局部搜索, 因此适应度最好的前1/4的灰狼个体被划分到亚群 1 中, 剩余3/4的灰狼个体被划分到亚群 2 中. 随着迭代的进行, 算法会逐渐从全局搜索转向局部搜索, 因此两个亚群内的个体数量也需要动态调整. 亚群的个体数量由式 (28)、式 (29) 来决定, 以确保算法在不同阶段都能保持最佳的性能.

$$G_1 = (1 - \left(\frac{3}{4} \cdot \sin\left(\frac{t\pi}{2T} + \frac{\pi}{2}\right)\right) \cdot N \tag{28}$$

$$G_2 = N - G_1 \tag{29}$$

其中, G_1为亚群 1 的数量, 向下取整且最大值为$N-3$, G_2为亚群 2 的数量, N为灰狼种群数量.

2.2.2 高斯变异策略

灰狼优化算法迭代过程中的寻优能力主要依靠α狼, β狼和δ狼的位置变化, 但灰狼优化算法本身不具有变异机制, 这就可能导致算法迭代过程中的种群多样性降低, 收敛速度下降. 为了解决这一问题, 本文在原有的位置更新公式的基础上加上了一个服从高斯分布的随机扰动项, 定义如式 (30) 所示.

$$X(t+1) = \frac{X_1 + X_2 + X_3}{3} \cdot (1 + Gauss(0,1)) \tag{30}$$

其中, $Gauss(0,1)$为均值为 0, 标准差为 1 具有高斯分布性质的随机变量. 因此, 灰狼的位置更新方法从式 (23) 替换成式 (30).

2.2.3 贪婪反向学习策略

由于算法的迭代过程中存在容易陷入局部最优解的问题, 本文引入了反向学习机制[22]. 反向学习的核心思想在于生成个体的反向解. 假设D维搜索空间中的灰狼个体$X = (x_1, x_2, \cdots, x_d)$, 则其反向解$\bar{X} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_d)$的定义如下:

$$\bar{x}_i = \begin{cases} 1 - x_i, & rand > pl \\ x_i, & \text{其他} \end{cases} \tag{31}$$

其中, $rand$为[0,1]之间的随机数; pl是反向学习系数,

表明有pl比例的x_i变异为二进制相反数.

反向学习既可以使得算法当前解迅速接近最优解, 也可能使得较优解迅速远离最优解, 为了避免后者这种情形, 本文采用了一种反向学习结合贪婪策略二次选择的方法. 该方法在当前灰狼个体的位置X和通过反向学习得到的新位置$\bar{X}$之间进行比较, 选择适应度更优的位置进行保留, 见式 (32). 该方法能够有效增强算法的全局搜索能力, 从而提高算法的收敛速度.

$$X^{\text{new}}=\begin{cases}\bar{X}, & f(\bar{X})\leqslant f(X)\\ X, & f(\bar{X})>f(X)\end{cases} \tag{32}$$

2.2.4 算法流程

结合上述改进策略, BGWOSGR 的流程如算法 1.

算法 1. BGWOSGR 算法

输入: 用户数量N, 无人机数量M, 不同用户任务的 DAG 矩阵, 用户和无人机的卸载关系矩阵Z, 适应度计算函数$\sum_{n=1}^{N}Y_n(X_n)$.

输出: 最优适应度值, 最优卸载策略.

初始化: 最大迭代次数T, 灰狼种群数量P, 灰狼位置矩阵 (卸载方案矩阵) $X=(X^1,X^2,\cdots,X^P)^{\mathrm{T}}$, 其中$X^P=(X_1,X_2,\cdots,X_n)$;

1. while $t\leqslant T$ do

2. 将灰狼种群自适应划分为灰狼亚群 1 和亚群 2, 亚群 1 为$P_1=(1-(\frac{3}{4}\cdot\sin(\frac{\pi t}{2T}+\frac{\pi}{2}))\cdot P$, 亚群 2 为$P_2=P-P_1$;

3. for i=1:P_1 //亚群 1 负责局部搜索

4. $fit\_1(i)=\sum_{n=1}^{N}Y_n(X_n)$ //计算亚群 1 的个体适应度

5. 更新α, β和δ狼; 根据式计算出a, 并根据式 (18) 和式 (19) 计算出A和C; 根据式 (30) 对灰狼位置进行更新, 并根据式 (24) 和式 (25) 获得新的灰狼位置X^i;

6. end for

7. for j=1:P_2 //亚群 2 负责全局搜索

8. $fit\_2(j)=\sum_{n=1}^{N}Y_n(X_n)$ //计算亚群 2 的个体适应度

9. 更新α, β和δ狼; 根据式 (27) 计算出a, 并根据式 (18) 和式 (19) 计算出A和C; 根据式 (23)、式 (31) 和式 (32) 对灰狼位置进行更新, 并根据式 (24) 和式 (25) 获得新的灰狼位置X^j;

10. end for

11. $fitbest(t)=\min(fit\_1,fit\_2)$ //记录适应度值的最小值

12. $Xbest(t)=\min(X^1,X^2,\cdots,X^P)$ //记录当前迭代最优卸载方案

13. 将亚群 1 和亚群 2 重组为数量为P的灰狼种群

14. 迭代次数++

15. end while

16. 输出最优的适应度值$\min(fitbest)$与其对应的最优卸载策略$Xbest$

3 仿真与评估

3.1 仿真参数设置

本文仿真实验基于 Matlab 平台, 该平台适合本文的复杂问题模型求解. 假设了如下场景: 某一地区发生了地质灾害, 地面通信基站损坏, 当地路况较差. 为了恢复通信, 有关部门安排了M个携带 MEC 的无人机飞往灾区, 无人机群在灾区上方悬停, 为当地的N个用户提供通信和计算服务. 假设每个用户同一时隙有一个任务需要卸载, 该任务可以拆分为若干个模块, 根据模块间的依赖关系确定每个模块的执行顺序. 针对不同用户, 按照距离最近原则生成为用户n分配可供任务卸载的无人机m. 针对不同用户的任务, 随机生成多个模块和代表模块依赖关系的 DAG 矩阵作为系统的输入. 仿真实验部分参数配置如表 1 所示.

表 1 仿真实验参数配置

参数名称	数值
用户数量N	[30, 80]
无人机数量M	8
任务模块的数据量D_i^n (MB)	[400, 900]
任务模块的计算量C_i^n (G)	[1, 3.5]
任务模块的容忍时延$T_{i,\max}^n$ (s)	[1,2]
无人机飞行高度H (m)	10
无人机MEC的CPU频率f_m (GHz)	10
用户的CPU频率f_n (GHz)	2
信道带宽B (GHz)	0.5
用户传输功率P_n (W)	5

3.2 仿真结果分析

为了评估本文所提出的算法的性能, 本文将与以下 4 种基准方案进行对比: 1) 二进制蜻蜓算法 (binary dragonfly algorithm, BDA); 2) 遗传算法 (genetic algorithm, GA)[23]; 3) 二进制粒子群优化算法 (binary particle swarm optimization, BPSO)[24]; 4) 二进制灰狼优化算法 (binary grey wolf optimization, BGWO).

3.2.1 算法收敛性分析

不同算法的收敛情况如图 3 所示. 由图 3 可知, BDA 算法和 BGWO 算法的收敛速度较慢, GA 算法收敛慢且系统成本较高, BPSO 收敛快但其容易陷入局部最优解, BGWOSGR 算法收敛速度和 BPSO 算法相近, 但 BGWOSGR 算法的系统成本比 BPSO 算法低 16% 左右, 比 BGWO 算法低 13% 左右. 综合各种算法的收敛速度和寻优效果来看, BGWOSGR 算法在求解本文设计的问题模型的前提下, 具有更优越的性能.

3.2.2 不同用户数量下的性能分析

为了验证不同用户数量对系统性能的影响, 本文考虑动态调整用户数量以模拟不同场景下算法对系统性能的影响. 由图 4 可知, 随着用户数量的增加, 每一个时隙系统处理的任务也随之增加, 从而使得系统的

时延和能耗也随之增加.

相比于其他 4 种基准算法, 本文提出的算法具有更低的系统成本, 这说明, BGWOSGR 算法可以实现时延和能耗之间的均衡, 并选择最优的卸载策略.

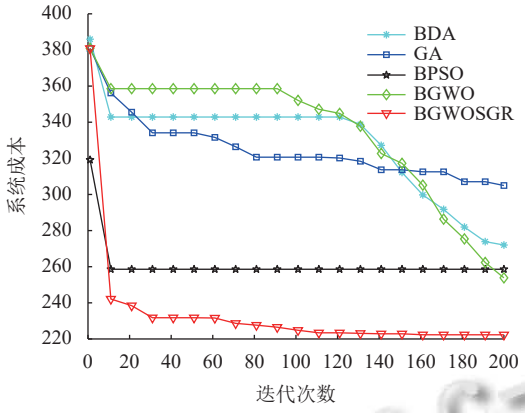

图 3 迭代次数与系统成本的关系

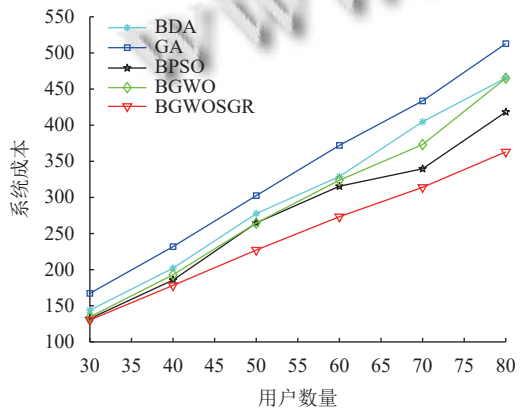

图 4 用户数量与系统成本的关系

3.2.3 不同任务模块计算量下的性能分析

从图 5 可知, 计算任务模块计算量的提高导致了所需要的执行时间和执行能耗的增加, 从而导致系统成本变大. 通过与其他 4 种算法对比, 当任务模块计算量较小时, 不同算法的系统成本接近, 但是当任务模块计算量逐渐增大后, BGWOSGR 算法的系统成本明显比其他 4 种算法更低, 具有更好的性能.

3.2.4 不同任务模块数据量下的性能分析

图 6 为不同任务模块数据量下的系统成本的变化. 每一种算法的系统成本都随着任务模块数据量的增加而增大, 这是由于任务模块数据量的增加导致了任务模块的传输时延变大, 从而影响了整体的系统成本. 实验数据表明, 对比方案下的 4 种算法的系统成本均大于 BGWOSGR 算法, BGWOSGR 算法大大降低了系统成本, 减少了资源浪费.

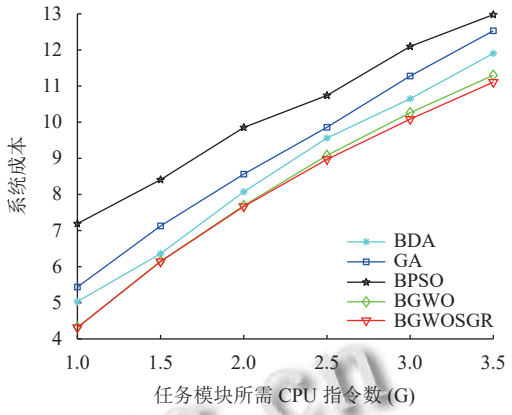

图 5 任务模块计算量与系统成本的关系

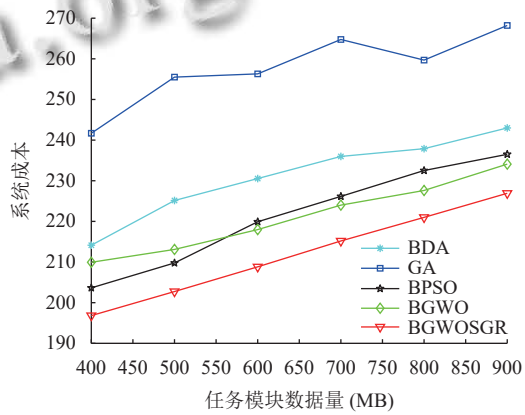

图 6 任务模块数据量与系统成本的关系

4 结束语

针对无人机辅助边缘计算的场景中具有依赖性的任务卸载问题, 本文通过构建依赖性矩阵来表示同一任务间不同模块的依赖关系. 提出一种 BGWOSGR 算法来求解多用户、多模块下的最小化系统时延-能耗加权开销的卸载策略. 通过最终的仿真结果可知, 与对照的方案相比, 本文提出的算法收敛性和降低系统成本的能力更强, 具有更好的优越性.

本文只考虑了单个任务卸载的情况, 实际情况下存在多个任务协同卸载的场景. 因此, 未来计划研究多个任务的协同卸载, 并对研究问题进行实验验证.

参考文献

1 Ji TX, Wan XL, Guan XJ, *et al*. Towards optimal application offloading in heterogeneous edge-cloud computing. IEEE Transactions on Computers, 2023, 72(11): 3259–3272. [doi: 10.1109/TC.2023.3290494]

2 Huang D, Wang P, Niyato D. A dynamic offloading algorithm for mobile computing. IEEE Transactions on

Wireless Communications, 2012, 11(6): 1991–1995. [doi: 10.1109/TWC.2012.041912.110912]

3 Zhang C, Zhang LY, Zhu LP, *et al*. 3D deployment of multiple UAV-mounted base stations for UAV communications. IEEE Transactions on Communications, 2021, 69(4): 2473–2488. [doi: 10.1109/TCOMM.2021.3049387]

4 Yuan XW, Xie ZD, Tan X. Computation offloading in UAV-enabled edge computing: A stackelberg game approach. Sensors, 2022, 22(10): 3854. [doi: 10.3390/s22103854]

5 Hosny KM, Awad AI, Khashaba MM, *et al*. Optimized multi-user dependent tasks offloading in edge-cloud computing using refined whale optimization algorithm. IEEE Transactions on Sustainable Computing, 2024, 9(1): 14–30. [doi: 10.1109/TSUSC.2023.3294447]

6 Dhingan D, Ghosh S, Naik BB, *et al*. Energy and delay efficient partial offloading for UAV-assisted MEC systems using differential evolution algorithm. Proceedings of the 3rd International Conference on Secure Cyber Computing and Communication (ICSCCC). Jalandhar: IEEE, 2023. 415–420. [doi: 10.1109/ICSCCC58608.2023.10176657]

7 Yan J, Bi SZ, Zhang YJ, *et al*. Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency. IEEE Transactions on Wireless Communications, 2020, 19(1): 235–250. [doi: 10.1109/TWC.2019.2943563]

8 Yuan Y, Su W. A game theory-based strategy for allocating and offloading computing resources in 5G networks. Proceedings of the 2023 International Conference on Networking and Network Applications (NaNA). Qingdao: IEEE, 2023. 92–97.

9 Lv XY, Du HW, Ye Q. TBTOA: A DAG-based task offloading scheme for mobile edge computing. Proceedings of the 2022 IEEE International Conference on Communications. Seoul: IEEE, 2022. 4607–4612.

10 Li QY, Peng B, Li Q, *et al*. A latency-optimal task offloading scheme using genetic algorithm for DAG applications in edge computing. Proceedings of the 8th International Conference on Cloud Computing and Big Data Analytics (ICCCBDA). Chengdu: IEEE, 2023. 344–348.

11 Bai ZY, Lin YF, Cao Y, *et al*. Delay-aware cooperative task offloading for multi-UAV enabled edge-cloud computing. IEEE Transactions on Mobile Computing, 2024, 23(2): 1034–1049. [doi: 10.1109/TMC.2022.3232375]

12 Fan YN, Zhai LB, Wang H. Cost-efficient dependent task offloading for multiusers. IEEE Access, 2019, 7: 115843–115856. [doi: 10.1109/ACCESS.2019.2936208]

13 Zhao M, Li XH, Sun C, *et al*. Dependency-aware hybrid task offloading in mobile edge computing networks. Proceedings of the 27th IEEE International Conference on Parallel and Distributed Systems (ICPADS). Beijing: IEEE, 2021. 225–232.

14 Xu L, Qian KN, Cai MK, *et al*. Design of task offloading algorithm for mobile edge computing network based on UAV. Proceedings of the 4th International Conference on Neural Networks, Information and Communication Engineering (NNICE). Guangzhou: IEEE, 2024. 1048–1052.

15 Zhou H, Wang ZN, Min GY, *et al*. UAV-aided computation offloading in mobile-edge computing networks: A stackelberg game approach. IEEE Internet of Things Journal, 2023, 10(8): 6622–6633. [doi: 10.1109/JIOT.2022.3197155]

16 Wang H, Zhang HJ, Liu XN, *et al*. Joint UAV placement optimization, resource allocation, and computation offloading for THz band: A DRL approach. IEEE Transactions on Wireless Communications, 2023, 22(7): 4890–4900. [doi: 10.1109/TWC.2022.3230407]

17 Bian YW, Sun Y, Zhai MD, *et al*. Dependency-aware task scheduling and offloading scheme based on graph neural network for MEC-assisted network. Proceedings of the 2023 IEEE/CIC International Conference on Communications in China (ICCC Workshops). Dalian: IEEE, 2023. 1–6.

18 Xiang K, He YJ. UAV-assisted MEC system considering UAV trajectory and task offloading strategy. Proceedings of the 2023 IEEE International Conference on Communications. Rome: IEEE, 2023. 4677–4682.

19 Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Advances in Engineering Software, 2014, 69: 46–61. [doi: 10.1016/j.advengsoft.2013.12.007]

20 Liang JJ, Suganthan PN. Dynamic multi-swarm particle swarm optimizer. Proceedings of the 2005 IEEE Swarm Intelligence Symposium. Pasadena: IEEE, 2005. 124–129.

21 Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1997, 1(1): 67–82. [doi: 10.1109/4235.585893]

22 Xu ZD, Su YB, Yang F, *et al*. A whale optimization algorithm with distributed collaboration and reverse learning ability. Computers, Materials & Continua, 2023, 75(3): 5965–5986. [doi: 10.32604/cmc.2023.037611]

23 Singh S, Ho Kim D. Profit optimization for mobile edge computing using genetic algorithm. Proceedings of the 2021 IEEE Region 10 Symposium (TENSYMP). Jeju: IEEE, 2021. 1–6.

24 Dai WF. Joint task offloading, resource allocation and data caching in MEC-assisted vehicular network. Proceedings of the 4th International Conference on Computer Engineering and Application (ICCEA). Hangzhou: IEEE, 2023. 70–76.

(校对责编: 张重毅)