计算机科学
COMPUTER SCIENCE

一种时延能耗感知的在轨边缘计算任务卸载调度方法

王众晓, 彭青蓝, 孙若骁, 徐锡峰, 郑万波, 夏云霓

引用本文

王众晓, 彭青蓝, 孙若骁, 徐锡峰, 郑万波, 夏云霓. 一种时延能耗感知的在轨边缘计算任务卸载调度方法[J]. 计算机科学, 2024, 51(6A): 240100188-9.

WANG Zhongxiao, PENG Qinglan, SUN Ruoxiao, XU Xifeng, ZHENG Wanbo, XIA Yunni. Delay and Energy-aware Task Offloading Approach for Orbit Edge Computing [J]. Computer Science, 2024, 51(6A): 240100188-9.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.240100188

一种时延能耗感知的在轨边缘计算任务卸载调度方法

王众晓[1] 彭青蓝[1] 孙若骁[1] 徐锡峰[2] 郑万波[3] 夏云霓[2]
1 河南大学人工智能学院 郑州 475004
2 重庆大学计算机学院 重庆 400044
3 昆明理工大学理学院 昆明 650031
(zhongxiaoWang@henu.edu.cn)

摘 要 全球智能设备的迅速增长引发了对计算资源下沉至边缘的巨大需求,催生了边缘计算范式的出现。同时,计算资源稀缺的偏远地区用户对算力的需求又推动了在轨边缘计算(Orbit Edge Computing,OEC)概念的提出和发展。在OEC场景下,偏远地区用户可以通过星地和星间通信链路将计算任务卸载至部署在低轨卫星上的边缘服务器,以此突破地面计算通信基础设施的限制,为偏远地区的用户提供低时延和高可靠的服务。然而,OEC场景中卫星算力受有限载荷和太阳能转化效率约束,同时还存在低轨卫星绕地导致的高度动态的星地连接造成的可用时隙有限的限制,面临着计算资源稀缺和可用通信时间有限所带来的挑战。因此,需要高质高效的任务卸载决策算法来保证OEC系统的高效运行。然而,目前在OEC场景下任务卸载方法大多在处理任务时无法兼顾计算任务卸载时延与能耗,此外传统方法还缺少对任务多样性的考量。针对上述问题,提出了一种基于自适应大邻域搜索的在轨边缘计算任务卸载方法OEC-ALNS,该方法以任务类型加权的任务处理成本为优化目标,并针对性地提出了基于最小化时延的破坏算子和修复算子来进一步提升搜索效率和卸载调度质量。基于Walker Delta低轨卫星星座和真实计算任务数据的实验结果表明,与传统的OEC-TA(OEC Task Allocation)方法相比,提出的OEC-ALNS方法在多个任务集异构的OEC场景中最多能够减少42.22%的加权任务处理成本和降低42.46%的平均时延。

关键词:在轨边缘计算;低轨卫星星座;计算任务卸载;自适应大邻域搜索
中图分类号 TP311

Delay and Energy-aware Task Offloading Approach for Orbit Edge Computing

WANG Zhongxiao[1],PENG Qinglan[1],SUN Ruoxiao[1],XU Xifeng[2],ZHENG Wanbo[3] and XIA Yunni[2]
1 College of Artificial Intelligence,Henan University,Zhengzhou 475004,China
2 College of Computer Science,Chongqing University,Chongqing 400044,China
3 Data Science Research Center,Kunming University of Science and Technology,Kunming 650031,China

Abstract The rapid growth of smart devices around the world has created a huge demand for computing resources to sink to the edge,giving rise to the emergence of the edge computing paradigm. At the same time,the demand for computing power in remote areas where computing resources are scarce has driven the concept of orbit edge computing(OEC). In the OEC scenario,users in remote areas can offload computing tasks to edge servers deployed on LEO satellites for processing and execution through the communication link between ground station and satellite and the communication link between satellites in constellation,so as to provide low-latency and high-reliability services for users in remote areas by utilizing satellite computing resources. However,the satellite arithmetic in the OEC scenario is constrained by the limited load and solar energy conversion efficiency,and there is also the limitation of limited available time slots due to highly dynamic satellite-ground connection caused by LEO satellites circling around the earth,which is faced with the challenge of the scarcity of computational resources and the limited available communication latency. Therefore,excellent task offloading decision algorithms are needed to ensure the efficient operation of OEC systems. However,most of the current task offloading approaches for OEC scenario are unable to take into account the delay cost and energy cost when processing tasks,and the traditional approaches also lack the consideration of task diversity. To address the above problems,an adaptive large neighborhood search-based task offloading method for orbit edge computing,OEC-ALNS,is proposed,which takes the task processing cost weighted by task type as the optimization objective,and consists of destruction and

基金项目:国家自然科学基金(62172062,62162036);河南省重点研发专项(231111211900);河南省自然科学基金青年项目(242300421700);河南省高等学校重点科研项目(24A520005)
This work was supported by the National Natural Science Foundation of China(62172062,62162036),Key Research and Development Project of Henan Province(231111211900),Young Scientists Fund of the Natural Science Foundation of Henan Province(242300421700) and Key Scientific Research Projects of Colleges and Universities in Henan Province(24A520005).
通信作者:彭青蓝(qinglan.peng@henu.edu.cn)

repair operators based on the minimization of latency. Experimental results based on Walker Delta LEO satellite constellation and real computing task data show that, compared with the traditional OEC-TA(OEC task allocation) approach, the proposed OEC-ALNS approach could achieve at most 42.22% reduction on the weighted task processing cost and most 42.46% reduction of the average latency cost in OEC scenarios with heterogeneous multiple task sets.

Keywords Orbit edge computing, Low-orbit satellite constellation, Computation task offloading, Adaptive large neighborhood search

1 引言

近年来，随着全球智能设备的迅速增长与移动应用程序的不断发展，以高清直播[1]、工业物联网[2]和智能家居[3]为代表的众多新型应用对计算资源的需求也在不断增加。在这种需求的推动下，以移动边缘计算[4]和多接入边缘计算[5](Multi-access Edge Computing, MEC)为代表的新兴计算范式应运而生，其通过提供高可靠与低时延的计算服务极大地改善了网络服务质量。然而，由于远海、孤岛、沙漠等偏远地区通信基础设施欠缺，难以部署完整的算力网络，导致无法实现偏远地区的无缝网络覆盖，仅利用有限的地面网络资源无法满足海量的服务质量(Quality of Service, QoS)感知需求[6]。相比传统的5G网络架构，6G通信技术充分利用了卫星通信网络，致力于构建空天地一体化的卫星地面网络[7-8]，这为解决偏远地区的计算需求问题提供了前所未有的机遇。由于低地球轨道(Low Earth Orbit, LEO)卫星具有轨道高度低、传播时延小、链路损耗低以及信道条件好等优点，因此LEO卫星成为卫星互联网的核心组件，也为实现未来6G移动网络的全覆盖提供了可能。在6G网络下的边缘计算范式中，通过将卫星互联网与地面网络结合，能够在低时延与高用户体验质量的条件下应对巨大增长的数据流量[9]。

结合了多接入边缘计算MEC与卫星地面网络的优势，卡耐基梅隆大学的Bradley[10-11]等首次提出了在轨边缘计算(Orbit Edge Computing, OEC)的概念。在OEC中，通过将计算、存储以及传感器等硬件设备部署在空天地海一体化网络中的通信卫星上，并将分散在每一颗卫星上的计算资源整合到同一架构中，实现分散卫星计算能力的融合，为卫星处理海量数据任务提供了统一的在轨边缘计算能力。然而，OEC场景中卫星计算能力受到有限载荷和太阳能转化效率约束，且面临因低轨卫星绕地导致的高度动态的星地连接造成的可用时隙有限的问题。因此，面对稀缺的卫星资源和高度动态的星地和星间通信，优秀的任务卸载决策算法是系统高效运行的关键。如何合理地调度卸载计算任务成为亟待解决的关键问题之一。

本文的主要贡献体现在以下3个方面：1)基于现实偏远地区计算服务请求与Walker Delta卫星星座，搭建了包括卫星星座、地面站以及用户的空天地海一体化在轨边缘计算架构，并根据卫星与地面基站之间的星地通信链路与卫星之间的通信链路建立了由通信模型、计算模型以及能量模型组成的高精度系统模型；2)在问题建模中，本文充分考虑了现实OEC场景下计算任务的多样性，即将任务集划分为时延敏感型任务与时延容忍型任务，并根据不同类型任务的数量将任务类型加权的任务处理成本作为系统的优化目标；3)针对OEC场景下的计算任务卸载调度问题，本文提出了一种基于自适应大邻域搜索的在轨边缘计算任务卸载方法OEC-ALNS，并将其与传统方法的性能进行对比分析。

本文第2章回顾了相关工作并对其局限性进行了分析；第3章给出了系统模型与问题描述；第4章详细描述了所提出的基于ALNS的在轨边缘计算任务卸载调度方法并对其时间复杂度进行了分析；第5章描述了基于Walker Delta低轨卫星星座和真实计算任务数据的实验结果并对结果进行了分析；最后总结全文并对未来工作做出展望。

2 相关工作

随着6G空天地一体化网络概念和技术的不断发展，在轨边缘计算已成为一个新兴的关注点。国内外学者通过设计网络性能良好的OEC架构与提出高效的计算任务卸载策略来降低不同场景下计算任务的成本，涌现出了一些代表性的探索工作。

文献[12]将具有计算和存储资源的MEC服务器部署在LEO卫星上，并提出了一种联合请求调度和服务放置机制RDSP，基于LEO星座网络模拟的实验表明其在服务用户比率和平均跳数方面具有更好的性能。在部署有MEC服务器的低轨卫星网络中，文献[13]针对多用户、多任务和多服务器条件下的低轨卫星边缘计算任务卸载与资源分配，提出了一种基于深度确定性策略梯度的方法。文献[14]将携有基站的无人机加入卫星物联网架构中，通过空中基站收集与发送任务请求，将该架构下的多任务联合卸载问题建模为有向无环图。此外，还提出了一种基于注意力机制和近端策略优化的协同算法A-PPO，用于来学习得到最优的任务卸载策略。文献[15]通过整合低轨卫星上MEC服务器与地面基站的计算资源，提出了一种LEO卫星边缘辅助多层MEC系统，建立了最小化计算时延与最小化系统能耗的组合优化数学模型，并使用经典的交替优化方法对原始问题进行分解，利用复杂度较低的迭代算法求解每个子问题，维持了较低的计算时延与系统能耗。

考虑到计算任务卸载问题往往是混合整数非线性规划问题(Mixed Integer Non-linear Programming, MINLP)，为解决此类问题，文献[16]提出了一种具有3层计算架构的混合云和边缘计算LEO卫星网络CECLS，并使用二元变量松弛法将非凸的总能耗最小化问题转化为可求解的线性规划问题。此外，该研究还提出了一种分布式算法，以在满足LEO卫星计算能力有限的约束条件下逼近最优解。文献[17]针对卫星辅助下的车对车通信场景，将不同任务卸载与资源分配问题解耦为两个子问题，并提出了基于深度强化学习的决策算法，用于求解子问题，获得了较低的系统延时与能耗。文献[18]研究了一种HAP-satellites边缘计算，即利用高空平台(High Altitude Platform, HAP)与LEO卫星协同，共同为地面用户提供计算服务。该研究将优化目标定义为最小化HAP-satellites边缘计算的加权能耗总和，并将所求优化问题解耦为

两个子问题。为解决该问题，Mei 等提出了一种智能启发式算法，通过启发式的修改任务分配策略来逼近全局最优解，使得系统的总加权能耗维持在较低水平。文献[19]则基于将非凸的问题转化为凸优化问题与将复杂问题解耦为子问题的思想，提出了一种星地物联网架构下最小化能耗的任务卸载策略。

虽然上述 OEC 架构及计算卸载策略能够有效地降低系统的总计算任务卸载时延与能耗，但鉴于真实应用场景下任务集合中涵盖不同类型的计算任务，针对不同类型的计算任务，需要适应性地最小化时延与能耗。因此，本文将任务集划分为时延敏感任务与时延容忍任务，并提出了一种 OEC 场景下基于自适应大邻域搜索（OEC-ALNS）的任务卸载算法来最小化系统的总成本。

3 系统建模与问题描述

受限于固定网络基础设施的覆盖范围约束，位于偏远、恶劣环境中的用户缺乏可用的边缘计算资源，难以直接通过地面服务器完成计算任务[20]。为了给偏远地区用户的任务请求提供充足的计算资源，本文重点关注通过地面用户终端将任务上传至卫星星座中的边缘服务器进行处理的在轨边缘计算场景。如图 1 所示，本文所考虑的轨道边缘计算卸载场景主要包括卫星星座、地面站以及用户 3 个关键组成部分。

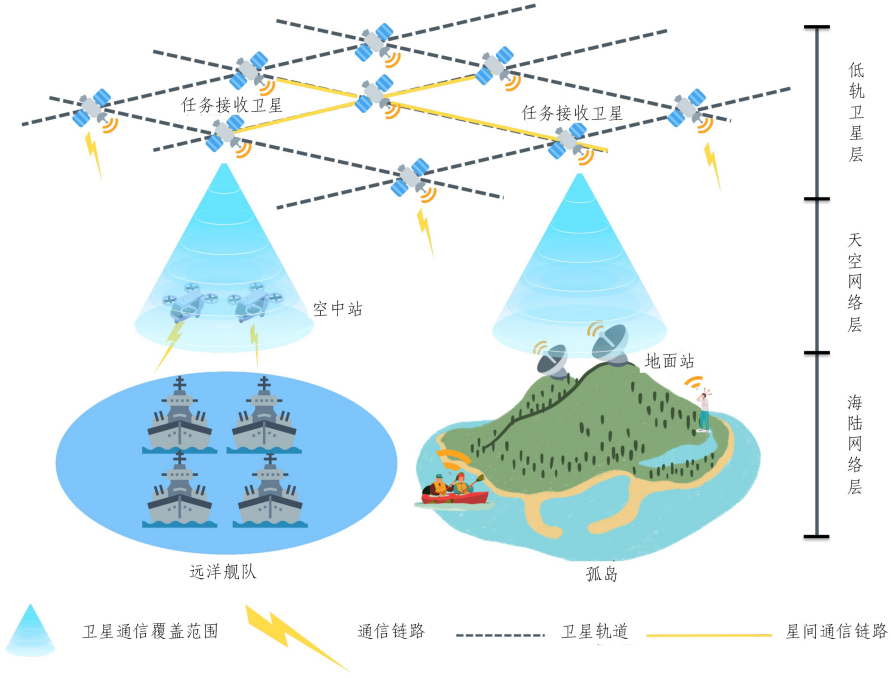

图 1 在轨边缘计算场景

Fig. 1 Orbital edge computing scenario

Walker Delta 卫星星座[21]是由大量分布在具有相同高度和倾角的圆形轨道上的低轨卫星所构成的 LEO 卫星星座。由于 Walker Delta 卫星星座中包含大量均匀分布在各个轨道平面上的 LEO 卫星，能够实现除两极地区外全球网络的无缝和连续覆盖，地面大部分用户在任何时候、任何位置都可以获取到卫星服务，因此本文采用 Walker Delta 星座架构为地面用户提供稳定的网络服务。

由于 LEO 卫星轨道速度高达 27 000 km/h，绕地一周约 90 min，因此对于地面用户来说，一个 LEO 卫星每次绕地的可用通信时间约为 15 min[20]。本文基于卫星与地面用户的空间几何模型[22]，通过 STK 软件对配备 45°半倾角传感器、550 km 轨道高度的低轨卫星[23]进行如图 2 所示的覆盖性分析，分析结果表明该场景中单个低轨卫星的通信覆盖范围大约为 3×10^5 km^2，能够实现大部分孤岛的全覆盖。本文用 $Sat=\{Sat_1,Sat_2,\cdots,Sat_M\}$ 表示某远海地区所有地面站当前可用的卫星集合，并选择所有地面站可用时隙最长的一个卫星 Sat_{access} 为此时地面用户的任务接收卫星。地面站负责汇总其覆盖区域内所有用户产生的计算任务，并通过地面网关以及地面站与卫星之间的星地通信链路将任务上传至接收卫星 Sat_{access}，由 Sat_{access} 根据部署的卸载策略决定任务是否在接收卫星本地处理。本文用 $Task=\{Task_1,Task_2,\cdots,Task_N\}$ 表示终端用户需要卸载的计算任务集合，其中每个 $Task_i$ 均为不可分割任务，由地面用户终端将任务集上传至卫星进行计算。与文献[15,17]一致，本文研究的任务卸载策略旨在优化任务的总卸载时延与能耗。一方面，任务分配的总时延涵盖了传输时延、传播时延、计算时延和排队时延等几个要素。另一方面，用户任务分配的总能量消耗包括传输能耗和计算能耗两个主要方面。接下来对涉及的时延模型和能耗模型进行详细讨论。

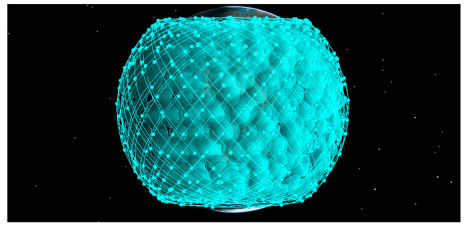

图 2 Walker Delta 卫星星座通信覆盖范围

Fig. 2 Communications coverage of Walker Delta satellite constellation

3.1 通信模型

为更好地评估在轨边缘计算任务卸载场景下的通信时延，本文建立了基于星地通信链路与星间通信链路的通信

模型。考虑到地面站与 LEO 卫星之间通信链路以及 Walker Delta 卫星星座中卫星之间的通信链路，用$\mathscr{C}_\alpha$表示地面站到任务接收卫星Sat_{access}的数据传输速率；C_β表示卫星到地面站的数据传输速率；$\mathscr{C}_\gamma$表示星座中各卫星之间的数据传输速率。因此，星间链路中卫星间的传输时延ρ_γ^t可被计算为[24]：

$$\rho_\gamma^t=\frac{\mathscr{R}_i}{\mathscr{C}_\gamma}+\frac{\widetilde{\mathscr{R}}_i}{\mathscr{C}_\gamma} \tag{1}$$

其中，$\mathscr{R}_i$为任务$Task_i$的数据量，$\widetilde{\mathscr{R}}_i$为完成任务$Task_i$后计算结果的数据量。

因此，任务接收卫星Sat_{access}本地处理任务时的传输时延$\rho_{i,access}^t$与Sat_{access}和其周围卫星协作处理任务时的传输时延$\rho_{i,j}^t(j\neq access)$可被计算为[25]：

$$\rho_{i,j}^t=\begin{cases}\dfrac{\mathscr{R}_i}{\mathscr{C}_\alpha}+\dfrac{\widetilde{\mathscr{R}}_i}{\mathscr{C}_\beta}, & j=access\\ \rho_{i,access}^t+\rho_\gamma^t, & j\neq access\end{cases} \tag{2}$$

任务的传播时延可定义为任务发送端到任务接收端之间的距离与数据传播速率的比值[25]。本文将从地面站发送任务到任务接收卫星的传播时延表示为$\rho_{i,j}^{\alpha_\beta}$，从卫星发送任务执行结果到地面站的传播时延表示为$\rho_{i,j}^{\beta_\alpha}$，卫星之间的传播时延表示为$\rho_{access,j}^{\gamma}$。因此任务接收卫星$Sat_{access}$本地处理任务时的传播时延$\rho_{i,access}^p$，以及$Sat_{access}$和其周围卫星协作处理任务时的传播时延$\rho_{i,j}^p(j\neq access)$可被计算为：

$$\rho_{i,j}^p=\begin{cases}\rho_{i,access}^{\alpha\_\beta}+\rho_{i,access}^{\beta\_\alpha}, & j=access\\ \rho_{i,access}^p+2\rho_{access,j}^{\gamma}, & j\neq access\end{cases} \tag{3}$$

3.2 计算模型

由于 OEC 场景下的任务请求需要在低轨卫星上部署的边缘服务器中完成计算，令δ_j为卫星Sat_j上搭载的 MEC 服务器处理 1 bit 数据所需要的 CPU 周期数；ξ_j为其 CPU 的执行速率。因此，任务 i 的计算时延$\rho_{i,j}^f$可被计算为[26]：

$$\rho_{i,j}^f=\frac{\mathscr{R}_i\cdot\delta_j}{\xi_j} \tag{4}$$

综上，在不考虑排队时延情况下，卸载任务 i 在卫星Sat_j中处理的总时延为：

$$\rho_{i,j}=\rho_{i,j}^t+\rho_{i,j}^p+\rho_{i,j}^f \tag{5}$$

然而，相比偏远地区旺盛的计算和通信资源需求，目前卫星计算资源还相对稀缺。当星上 MEC 服务器中有多个任务需要处理时，必须考虑任务计算所需的排队时延。当卫星Sat_j的任务处理队列中存在任务正在处理或排队等待处理时，考虑排队时延的模型总时延$t_{i,j}$被计算为：

$$t_{i,j}=\rho_{i,j}+t_{i,j}^q \tag{6}$$

$$t_{i,j}^q=\sum_{k=1}^{j-1}x_{ik}*\rho_{i,k} \tag{7}$$

其中，$t_{i,j}^q$为任务$Task_i$卸载至低轨卫星Sat_j执行时所产生的排队时延；x_{ik}为 0-1 决策变量，当其取值为 1 时，代表任务 i 卸载至卫星Sat_k执行；$\rho_{i,k}$为不考虑排队时延时任务$Task_i$卸载至卫星Sat_k执行所需要的时间。

3.3 能耗模型

由于 OEC 场景下涉及数据远距离传输和任务处理，导致了大量的能量消耗[27]。然而现有的 LEO 卫星大多数依赖太阳能帆板发电，这使得卫星上可用的能源相对有限。因此，在 OEC 场景下，如何有效地降低 LEO 卫星的任务卸载能耗成为了亟待解决的关键问题。为了更好地评估 OEC 场景下的任务卸载能耗，本文构建了基于星地通信链路的任务卸载能耗模型。令e_j为Sat_j卫星中 MEC 服务器计算任务时的能量消耗速率，则卫星中的 MEC 服务器计算任务的能量消耗可被计算为：

$$\varepsilon_j^{\mathrm{MEC}}=e_j\cdot\mathscr{R}_i\cdot\delta_j \tag{8}$$

因此，任务接收卫星Sat_{access}本地处理任务时的能量消耗$\varepsilon_{i,access}$与Sat_{access}和其周围卫星协作处理任务时的能量消耗$\varepsilon_{i,j}$($j\neq access$)可被计算为：

$$\varepsilon_{i,j}=\begin{cases}\dfrac{\mathscr{R}_i\cdot\varepsilon_\alpha}{\mathscr{C}_\alpha}+\dfrac{\widetilde{\mathscr{R}}_i\cdot\varepsilon_\beta}{\mathscr{C}_\beta}+\varepsilon_j^{\mathrm{MEC}}, & j=access\\ \dfrac{\mathscr{R}_i\cdot\varepsilon_\gamma}{\mathscr{C}_\gamma}+\dfrac{\widetilde{\mathscr{R}}_i\cdot\varepsilon_\gamma}{\mathscr{C}_\gamma}+\dfrac{\mathscr{R}_i\cdot\varepsilon_\alpha}{\mathscr{C}_\alpha}+\dfrac{\widetilde{\mathscr{R}}_i\cdot\varepsilon_\beta}{\mathscr{C}_\beta}+\varepsilon_j^{\mathrm{MEC}}, & j\neq access\end{cases} \tag{9}$$

其中，ε_α为地面站发送任务至接收卫星的能量消耗，ε_β为从卫星发送计算结果到地面的能量消耗，ε_γ为卫星链路之间任务传输的能量消耗。

3.4 问题描述与分析

基于上述系统建模，为了在提高用户体验质量的同时最小化卫星计算任务的总成本，在卫星的计算资源有限的条件下，本文致力于解决 OEC 场景下计算任务的卸载调度问题，该问题可以被描述为如下的一个最小化成本问题：

$$\min_X\sum_{i=1}^{N}\sum_{j=1}^{M}\{a*x_{ij}*t_{ij}+b*x_{ij}*\varepsilon_{ij}\} \tag{10}$$

$$\text{s.t. } t_{ij}=\rho_{ij}+\sum_{k=1}^{j-1}x_{ik}*\rho_{ik} \tag{11}$$

$$\sum_{j=1}^{M}x_{ij}=1 \tag{12}$$

$$x_{ij}*t_{ij}\leqslant\tau \tag{13}$$

$$x_{ij}*\mathscr{RS}_j^r\geqslant\mathscr{RS}_i \tag{14}$$

$$x_{ij}\in\{0,1\} \tag{15}$$

$$i=1,\cdots,N,j=1,\cdots,M \tag{16}$$

由于在现实场景中地面用户任务请求一般分为时延敏感型与时延容忍型两类，本文根据用户任务集 $Task$ 中时延敏感任务的数量N_s与时延容忍任务的数量N_t设计如式(16)所示的时延权重系数 a 与能量权重系数 b：

$$\begin{cases}a=\dfrac{N_s}{N_s+N_t}\\ b=\dfrac{N_t}{N_s+N_t}\end{cases} \tag{17}$$

从而根据任务集中不同类型任务的数量调整模型总成本中计算任务卸载时延和能耗的权重。

在建模的优化问题中，目标函数(9)中的x_{ij}为 0-1 决策变量，当其取值为 1 时，代表任务 i 卸载至卫星Sat_j执行。约束条件(10)是等式约束，用于计算考虑排队时延情况下的卸载时延t_{ij}；约束条件(12)是非线性不等式约束，表示任务 i 的完成时间不能超过上界τ；约束条件(13)是线性不等式约束，表示卫星Sat_j中的可用计算资源$\mathscr{RS}_j^r$要大于任务 i 所需计算资源$\mathscr{RS}_i$。

由于任务接收卫星Sat_{access}可获取到与其有星间链路的周围卫星当前剩余的计算资源，因此卫星Sat_j剩余的计算资源$\mathscr{RS}_j^r$可被计算为：

$$\mathscr{RS}_j^r=\mathscr{RS}_j-\mathscr{RS}_j^\varphi,\mathscr{RS}_j^\varphi\in[0,\mathscr{RS}_j] \tag{18}$$

其中，$\mathscr{RS}_j$为卫星Sat_j服务器的总计算资源，$\mathscr{RS}_j^\varphi$为卫星Sat_j服务器中计算任务所占用的计算资源。

由于约束条件(14)将x_{ij}约束为一个0-1变量，且约束条件(12)为非线性不等式约束，因此该优化问题是一个混合整数非线性规划(MINLP)问题。该问题已被广泛证明为NP-hard[28]，无法在多项式时间内得到精确解。因此，如何设计高效精确的近似算法用于求解成为一个关键挑战。

4 基于ALNS的在轨边缘计算任务卸载策略

为了通过有效的任务卸载和资源调度，优化整个系统的性能，提高用户体验质量，并在有限计算资源的条件下最小化总计算成本，本文提出了一种基于OEC-ALNS的轨道边缘计算任务卸载方法，其流程如图3所示。可以看出，地面用户首先通过其附近地面站将任务上传至任务接收卫星进行处理。然后为了使计算任务所消耗的时延和能量达到最小，需要通过任务接收卫星充分调度星座中其他卫星的计算资源来完成地面用户任务。最后当卫星完成计算任务后，将计算结果返回地面用户，从而完成整个计算流程。由于OEC场景下卫星的可用性在不断变化，因此计算卸载任务调度的时效性尤为重要，然而受限于所求问题的NP-Hard问题无法在多项式时间内得到全局最优解。为此，本文提出了一种如算法1所示的OEC-ALNS算法来逼近全局最优解。

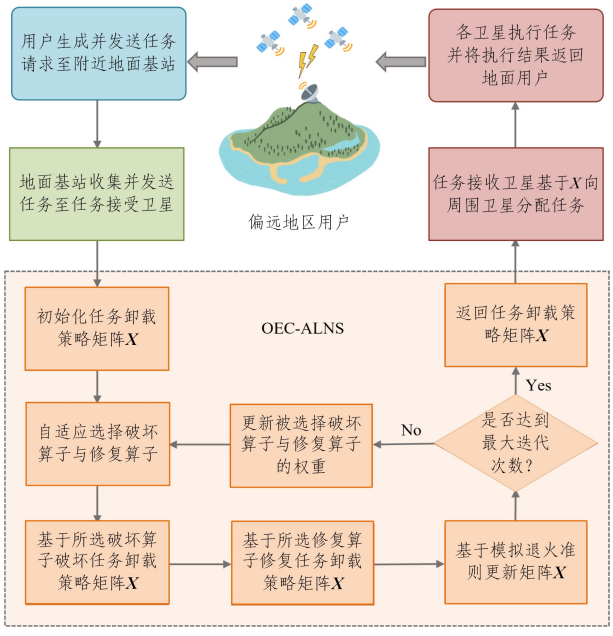

图3 OEC-ALNS算法的架构

Fig. 3 Framework of OEC-ALNS

算法1 基于自适应大邻域搜索的在轨边缘计算任务卸载算法(OEC-ALNS)

输入：任务数量 N，卫星数量 M

输出：分配策略矩阵 $\mathbf{X}^{\text{Index}}$，总成本 $\sigma_{\text{total}}(\mathbf{X}^{\text{Index}})$

1. 初始化：$\mathcal{R}_i$，$\widetilde{\mathcal{R}}_i$，$\tau$，$\mathcal{SP}_i$，$\mathcal{SP}_j^r$，$\xi_j$，$\delta_j$，$e_j$
2. 计算时延矩阵 $\boldsymbol{\rho}$ 和能量矩阵 $\mathbf{E}$
3. 初始化：$\mathbf{X}^{\text{Index}}$，$\mathbf{X}_{\text{best}}^{\text{Index}}$，I，$T_{\text{init}}^{\text{SA}}$，$T_{\text{end}}^{\text{SA}}$，$\boldsymbol{\omega}$，$\mathbf{S}_\omega$，$\mathbf{T}_\omega$
4. Foreach iteration Do
5. While $T_{\text{end}}^{\text{SA}} < T_{\text{init}}^{\text{SA}}$ Do
6. 基于轮盘赌选择算子选择破坏算子 f^d 和修复算子 f^r
7. $\mathbf{X}^{\text{destroy}} \leftarrow f^d(\mathbf{X}^{\text{Index}})$
8. $\mathbf{X}^{\text{repair}} \leftarrow f^r(\mathbf{X}^{\text{destroy}})$
9. If $\sigma_{\text{total}}(\mathbf{X}^{\text{repair}}) \leqslant \sigma_{\text{total}}(\mathbf{X}_{\text{best}}^{\text{Index}})$ Then
10. $\mathbf{X}_{\text{best}}^{\text{Index}} \leftarrow \mathbf{X}^{\text{repair}}$
11. End If
12. $\mathbf{X}^{\text{Index}} \leftarrow \text{Metropolis}(\mathbf{X}^{\text{repair}})$
13. $T_{\text{init}}^{\text{SA}} \leftarrow \alpha * T_{\text{init}}^{\text{SA}}$
14. 更新 ω，S_ω，T_ω
15. End While
16. End For
17. Return $\mathbf{X}^{\text{Index}}$，$\sigma_{\text{total}}(\mathbf{X}^{\text{Index}})$

首先OEC-ALNS算法使用OEC-TA算法[23]初始化一组可行的任务分配矩阵$\boldsymbol{X}^{\text{Index}}$，在初始化的过程中对不满足约束条件(12)和(13)的任务卸载策略进行标记，从而保证算法所求解的可行性(如算法1中的第1—3行所示)。在接下来的每一轮迭代中，使用轮盘赌选择算子[29]，依据破坏算子和修复算子的自适应权重ω选择破坏算子与修复算子(如算法1中的第6行所示)。选定算子后，针对当前解进行破坏和修复得到新解，对于得到的新解，先使用$\sigma_{\text{total}}(\cdot)$计算当前分配策略的总成本：

$$\sigma_{\text{total}}(\boldsymbol{X}) = \sum_{i=1}^{N}\sum_{j=1}^{M}\{x_{ij} * t_{ij} + x_{ij} * \varepsilon_{ij}\} \tag{19}$$

并判断该任务分配策略是否优于全局最优解，然后采用模拟退火准则$Metropolis(\cdot)$判断是否接受该新解[30]：

$$Prob = \begin{cases} 1, & \sigma_{\text{total}}(\boldsymbol{X}^{\text{repair}}) \leqslant \sigma_{\text{total}}(\boldsymbol{X}) \\ e^{\frac{\sigma_{\text{total}}(\boldsymbol{X}) - \sigma_{\text{total}}(\boldsymbol{X}^{\text{repair}})}{T^{SA}}}, & \sigma_{\text{total}}(\boldsymbol{X}^{\text{repair}}) > \sigma_{\text{total}}(\boldsymbol{X}) \end{cases} \tag{20}$$

其中，$Prob$为模拟退火准则接收新解的概率，根据此概率$Metropolis(\cdot)$返回更新后当前解X^{Index}的值。判断完成后，使用式(20)更新自适应权重[31]：

$$\omega = \begin{cases} \omega, & T_\omega = 0 \\ (1-w)\omega + w\dfrac{S_\omega}{T_\omega}, & T_\omega > 0 \end{cases} \tag{21}$$

其中，w为更新系数，S_ω为累加分数，T_ω为使用次数。对于累加分数S_ω，本文做如下规定：

$$S_\omega = S_\omega + Score_i \tag{22}$$

在更新算子分数过程中，当新解优于此时的最优解时，分值记为$Score_1$；当新解优于当前解时，分值为$Score_2$。注意到，当且仅当新解为劣解但在模拟退火准则$Metropolis(\cdot)$下接受该解时，得分为$Score_3$，否则得分为$Score_4$(如算法1中的第14行所示)。之后更新模拟退火温度T^{SA}继续迭代直至达到最低温度T_{end}^{SA}和最大迭代次数I，最后返回最终的计算任务卸载策略矩阵$\boldsymbol{X}$，至此OEC-ALNS算法结束，返回任务分配策略$\boldsymbol{X}^{\text{Index}}$与总成本$\sigma_{\text{total}}(\boldsymbol{X}^{\text{Index}})$。

从提出的OEC-ALNS方法流程可以看出，破坏算子与修复算子决定了算法能否逼近全局最优解，因此设计合理的破坏算子与修复算子成为了解决最小化成本计算任务卸载问题的关键挑战。为此，本文设计了4种破坏算子(随机破坏算子、混合破坏算子、最大时延破坏算子、最先处理任务破坏算子)以及3种修复算子(随机修复算子、第二时延修复算子、贪心修复算子)。

由于该优化问题需要考虑到排队时延，而传统的OEC-TA算法采用的是一种贪心策略，导致排队时延过大，难以获得时延较低的任务分配方案。为了减小排队时延，本文引入如算法2所示的混合破坏算子，混合破坏算子选择了l_1个时延最小的任务以及l_2个时延最大的任务，并将它们添加到破

坏列表 *Removelist* 中。

算法 2 混合破坏算子

输入：$\mathbf{X}^{\text{Index}}$，T，N，k，$\mathbf{T}^{\text{Index}}$

输出：**Removelist**

1. 初始化：**Removelist**，$\mathbf{list}_1$，$\mathbf{list}_2$，l_0，l_1，l_2
2. For k＜N Do
3. $\mathbf{T}_k^{\text{Index}} \leftarrow \mathbf{T}_k, X_k^{\text{Index}}$
4. End For
5. $\mathbf{list}_1 \leftarrow$argsort($\mathbf{T}^{\text{Index}}$)[0：$l_1$]
6. **Removelist**←**Removelist**∪$\mathbf{list}_1$
7. $\mathbf{list}_2 \leftarrow$argsort($\mathbf{T}^{\text{Index}}$)[$l_0-l_1$：$l_0$]
8. **Removelist**←**Removelist**∪$\mathbf{list}_2$
9. Return **Removelist**

为了修复出使得排队时延更小的卸载方案，本文引入如算法 3 所示的第二时延修复算子，该修复算子旨在将 ***Removelist*** 中的前l_1个任务用T_k中第二小时延的分配策略修复，其余任务使用最小时延分配策略修复。此外，本文还提出了一种如算法 4 所示的贪心修复算子，该修复算子首先使用 *sample*(·)函数随机从 ***Removelist*** 中采样出一个待修复的任务索引Rd_{index}，然后根据时延$\boldsymbol{T}_k$最小化的分配策略将该索引之前的任务进行分配，最后遍历$\boldsymbol{X}_{Rd_{\text{index}}}^{\text{Index}}$和$\boldsymbol{X}_{Rd_{\text{index}}+1}^{\text{Index}}$，得到一个使总成本更小的任务分配矩阵 $\boldsymbol{X}$。

算法 3 第二时延修复算子

输入：$\boldsymbol{X}^{\text{Index}}$，***Removelist***，$\boldsymbol{T}$

输出：$\mathbf{X}^{\text{Index}}$

1. 初始化：l_0，l_1，l_2
2. $l_2 \leftarrow l_0 - l_1$
3. For k＜l_1 Do
4. $\mathbf{X}_{\text{Removelist}_k}^{\text{Index}} \leftarrow$argsecondmin($\mathbf{T}_{\text{Removelist}_k}$)
5. End For
6. For k＜l_2 Do
7. $\mathrm{X}_{\text{Removelist}_{l_0-k}}^{\text{Index}} \leftarrow$argmin($\mathbf{T}_{\text{Removelist}_{l_0-k}}$)
8. End For
9. Return $\mathbf{X}^{\text{Index}}$

算法 4 贪心修复算子

输入：$\mathbf{X}^{\text{Index}}$，**Removelist**，T，N，M

输出：$\mathbf{X}^{\text{Index}}$

1. 初始化：l_0，l_1，l_2
2. $\mathbf{Rd}_{\text{index}} \leftarrow$sample(**Removelist**)
3. For k＜N Do
4. If k＜$\mathrm{Rd}_{\text{index}}$ Then
5. $\mathrm{X}_k^{\text{Index}} \leftarrow$argmin($\mathbf{T}_k$)
6. 更新 **T**
7. ElseIf k＝$\mathrm{Rd}_{\text{index}}$ Then
8. $\mathrm{X}_k^{\text{Index}} \leftarrow$argmin($\sigma_{\text{total}}(\mathbf{X}_{\mathrm{Rd}_{\text{index}}}^{\text{Index}})$)
9. $\mathrm{X}_{k+1}^{\text{Index}} \leftarrow$argmin($\sigma_{\text{total}}(\mathbf{X}_{\mathrm{Rd}_{\text{index}}+1}^{\text{Index}})$)
10. Break
11. End If
12. End For
13. Return $\mathbf{X}^{\text{Index}}$

综上，提出的 OEC-ALNS 算法由迭代、破坏当前解、修复当前解和接受新解 4 个步骤组成，令任务个数为 N，最大迭代次数为 I，模拟退火系数为 α。在迭代步骤中，算法执行的时间复杂度为 O(I)；在破坏当前解步骤中，基于破坏算子破坏解的时间复杂度为 O(N^2)；在修复当前解步骤中，基于修复算子修复解的时间复杂度为 O(l_0)；在接受新解步骤中，本文使用模拟退火准则来使算法跳出局部最优解，执行退火过程的时间复杂度为 O($\log_\alpha(T_{\text{end}}^{SA}/T_{\text{init}}^{SA})$)。因此，提出的 OEC-ALNS 算法的总时间复杂度为 O($I\times N^2\times l_0\times\log_\alpha(T_{\text{end}}^{SA}/T_{\text{init}}^{SA})$)。

5 实验与分析

5.1 实验设置

与相关工作[21-22]一致，本文采用 Walker Delta 卫星星座的 6 颗可用卫星组成 OEC 计算平台开展实验，以验证提出方法的正确性和优越性。在该架构下，任务接收卫星Sat_{access}与其周围 5 颗卫星之间存在星间链路，在此架构基础之上，本文的实验参数如表 1 所列。

表 1 实验参数配置

Table 1 Simulation parameters

参数	值	参数	值
$\mathscr{C}_\alpha$/Mbps	200	ε_γ/(J/s)	0.03-0.12(均匀分布)
$\mathscr{C}_\beta$/Mbps	300	$\mathscr{RF}_j^r$/Task	10-20
$\mathscr{C}_\gamma$/Gbps	10	τ/s	2
$\mathscr{R}_i$	10 Kbit～3 Mbit(均匀分布)	α	0.97
$\widetilde{\mathscr{R}}_i$	2 Kbit～5 Kbit(均匀分布)	I	1 000
$\rho_{i,j}^{\alpha,\beta}$，$\theta_{i,j}^{\beta,\alpha}$/ms	1.815	T_{init}^{SA}	100
δ_j/(cycles/bit)	1 000	T_{end}^{SA}	10
ξ_j/GHz	3～10(均匀分布)	***Score***	[1.5 1.2 0.8 0.6]
e_j/(J/GHz)	0.3～0.5	$\boldsymbol{\omega}$	全 1 矩阵
ε_α/(J/s)	4	$\boldsymbol{S}_\omega$	全 1 矩阵
ε_β/(J/s)	1	$\boldsymbol{T}_\omega$	全 0 矩阵

实验考虑将 Random Allocation 算法、Hungarian 算法[32]、OEC-TA[22] 3 种算法作为基线，其中 Random Allocation 算法是基于本文构建的通信模型与能耗模型，对问题场景进行蒙特卡洛模拟，随机分配任务的策略；Hungarian 算法是一种被广泛证明能解决指派问题[33-34]的组合优化算法，而本文所关注的在轨边缘计算任务卸载调度问题可抽象为一个指派问题，通过 Hungarian 算法能够在多项式时间内得到时延和能耗较低的可行解；OEC-TA 算法是一种基于贪心思想的任务分配算法，该算法对于不同类型的任务贪心选择最小化时延或最小化能耗来解决任务分配问题。

实验平台为 10-th Intel Core i7，16G RAM，512 G SSD，Win11 操作系统，所有算法和实验框架均由 C＋＋11 实现。注意到，由于对比算法中存在一些随机性，为保证算法性能对比的公平性，所有实验均重复进行 30 次，并取平均结果进行展示。

5.2 实验结果与分析

1)平均时延分析

首先对算法在不同任务量下的平均任务卸载时延进行评估。图4给出了不同算法下任务数量与平均任务卸载时延的关系，可以看到随着任务数量的增加，由于排队时延的不断增加，不同算法的平均总时延都在增加。与其他基线算法相比，本文提出的OEC-ALNS算法在最小化时延问题中取得了最低的任务卸载时延。具体而言，相比Random Allocation算法平均降低了36.92%的任务卸载时延；相比匈牙利算法平均降低了19.15%的任务卸载时延；相比OEC-TA算法平均降低了30.27%的任务卸载时延。注意到，提出的OEC-ALNS算法的优越性随着任务数量的增加越发明显，特别是当任务量达到30以上时，OEC-ALNS算法相比Hungarian算法可有效降低任务卸载时延达16.13%～26.28%。

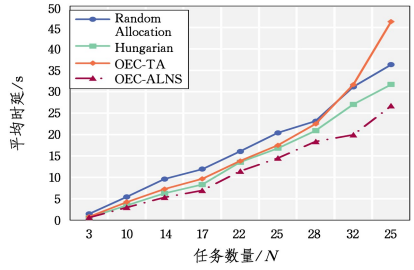

图4 任务数量与平均任务卸载时延在不同算法下的关系

Fig. 4 Relationship between the number of tasks and average total offloading delay of different algorithms

同时实验结果还表明了在任务数量小于等于17时，OEC-TA算法、Hungarian算法和OEC-ALNS算法之间的最小化时延性能并无显著差别。当任务数量继续增多时，OEC-ALNS算法在降低任务卸载时延方面的性能逐渐显著。这是因为OEC-TA算法在处理时延容忍问题时没有考虑计算任务卸载时延，这导致了多任务情况下的排队时延激增。提出的OEC-ALNS算法之所以能在Walker Delta星座中取得显著降低任务卸载时延的效果，是因为本文基于最小化时延的思想设计了破坏算子与修复算子，从而达到降低排队时延的目的。

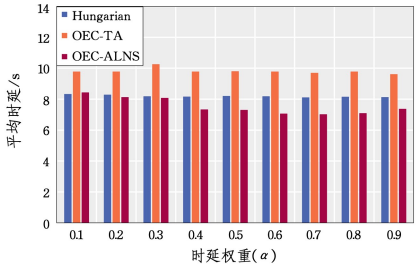

图5 时延权重系数与平均任务卸载时延在不同算法下的关系($N=17$)

Fig. 5 Relationship between the delay weight and average total offloading delay of different algorithms($N=17$)

此外，实验还对在任务量为17时的不同算法在不同时延权重系数下的平均任务卸载时延进行评估。图5给出了在不同时延权重系数下，提出的OEC-ALNS算法相比其他基线算法在最小化任务卸载时延问题中具有最优的性能，并且随着时延权重系数的增大，OEC-ALNS算法降低时延的性能逐渐显著。具体而言，相比Hungarian算法，OEC-ALNS算法平均降低了9.26%的任务卸载时延，相比OEC-TA算法平均降低了23.25%。这是因为提出的OEC-ALNS算法将加权后的任务卸载时延与能耗作为所关注的在轨边缘计算任务卸载调度问题的优化目标，并基于最小化时延的思想设计了用于降低排队时延的破坏算子与修复算子，从而在不同时延权重系数下呈现出优越的最小化任务卸载时延性能。

2)平均能耗分析

接下来对算法在不同任务量下的平均任务卸载能耗进行评估。图6给出了在不同任务量下提出的OEC-ALNS算法和基线算法的最小化能耗表现。可以看出，与Random Allocation算法相比，提出的OEC-ALNS算法平均降低了11.77%的任务卸载能耗；相比Hungarian算法，任务卸载能耗平均降低12.36%；相比OEC-TA算法，最小化能耗性能几乎无差。提出的OEC-ALNS算法之所以能在最小化能耗问题中具有较好的性能，是因为本文将计算任务卸载能耗加入优化问题的目标函数，从而保证算法得到能耗较低的任务卸载策略。而Hungarian算法在任务量多于卫星上MEC服务器数量时，会虚拟出服务器进行任务分配，这就导致了计算资源的浪费，使得任务分配策略的能耗较高。

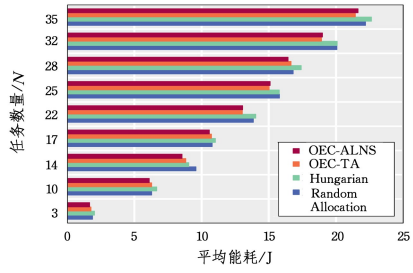

图6 任务数量与平均总能耗在不同算法下的关系

Fig. 6 Relationship between the number of tasks and average total energy consumption of different algorithms

3)平均成本分析

实验还对提出的OEC-ALNS方法和基线算法在17个任务中不同时延权重系数下的总成本进行了评估。图7给出了在不同时延权重系数下，提出的OEC-ALNS算法相比其他基线算法在最小化任务卸载成本问题中具有最优的性能。具体而言，相比Random Allocation算法，OEC-ALNS算法平均降低了23.15%的任务卸载成本；相比Hungarian算法平均降低了11.85%；相比OEC-TA算法平均降低了16.82%。

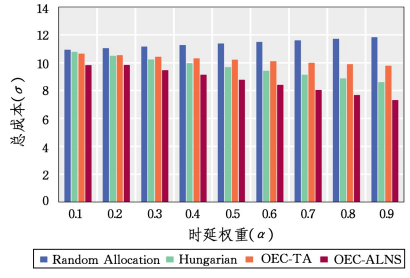

图7 时延权重系数与平均总成本在不同算法下的关系($N=17$)

Fig. 7 Relationship between the delay weight and average total cos of different algorithms($N=17$)

这是因为本文提出的OEC-ALNS算法同时考虑到任务卸载时延与任务卸载能耗，设计了如式(10)所示的目标函数，

并且基于最小化时延的思想设计了破坏算子与修复算子，从而在时延权重系数增大时显著地降低了总成本。

此外，实验还对算法在时延权重系数不同时在不同任务量下的总成本进行评估。图 8 给出了在任务数量相同时，随着时延权重系数的增大，提出的 OEC-ALNS 算法最小化成本性能较 OEC-TA 算法越显著。在时延权重系数相同时，随着任务数量的增加，提出的 OEC-ALNS 同样能够在最小化成本问题中展现出优越的性能。具体而言，当时延权重系数为 0.1 时，提出的 OEC-ALNS 算法较 OEC-TA 算法的总成本平均降低 9.53%；时延权重系数为 0.5 时，较 OEC-TA 算法的总成本平均降低 22.04%；时延权重系数为 0.9 时，较 OEC-TA 算法的总成本平均降低 40.11%。

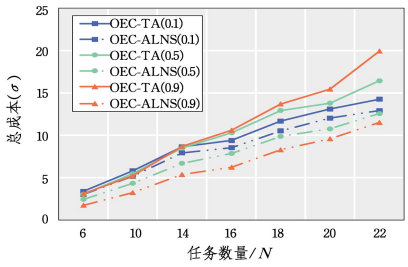

图 8 不同时延权重系数时不同任务量下不同算法的总成本

Fig. 8 Total cost of different algorithms with different number of task and different delay weight coefficients

提出的 OEC-ALNS 算法之所以能够在时延权重系数不同时在不同任务量下较 OEC-TA 算法具有优越的最小化成本性能，原因是 OEC-TA 算法处理时延容忍任务时并未考虑卸载时延，导致在最小化能耗时无法兼顾最小化时延，因此在多任务量时排队时延过大，导致总成本过高。而提出的 OEC-ALNS 算法处理不同类型任务时同时考虑到最小化时延与最小化能耗，因此在不同任务量和不同时延权重系数下较 OEC-TA 算法均能显著地降低总成本。

4)算法求解时间分析

最后实验对不同任务量下 OEC-ALNS 算法的求解时间进行评估。图 9 给出了随着任务数量的增加，算法的平均求解时间逐渐增大。具体而言，OEC-ALNS 算法的最长单次求解时间为 0.461 s，最短求解时间为 0.108 s，平均求解时间为 0.304 s。

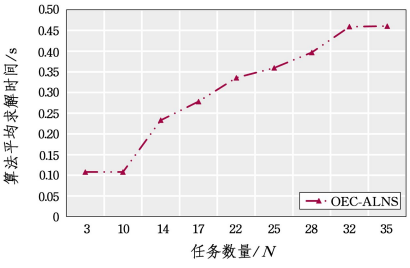

图 9 任务数量与 OEC-ALNS 算法求解时间的关系

Fig. 9 Relationship between the number of tasks and solution time of OEC-ALNS algorithm

综上，与 Random Allocation 算法、Hungarian 算法和 OEC-TA 等传统卫星边缘计算任务卸载方法相比，提出的 OEC-ALNS 算法能够在最小化能耗的同时有效地降低计算任务卸载时延，从而显著地降低计算任务的总成本。与 OEC-TA 算法相比，在能耗成本相近的效果下，任务卸载时延平均降低 30.30%，最高降低 42.46%。与 Hungarian 算法相比，总成本平均降低 11.85%，最高降低 17.36%。这是由于本文在构建目标函数时同时考虑了计算任务卸载时延与能耗，并且基于最小化时延设计了合理的破坏算子与修复算子，因此能够得到同时具备低能耗与低时延的任务卸载策略。

结束语 本文针对偏远地区计算资源稀缺的问题，构建了空天地海一体化的在轨边缘计算架构，并在考虑到计算任务多样性的基础上，提出了一种基于自适应大邻域搜索的在轨边缘计算任务卸载调度方法 OEC-ALNS。基于真实星座数据的实验表明，与传统基线方法相比，提出的方法可以以更低的卸载成本来满足偏远地区用户的计算任务需求。

随着在轨边缘计算技术的不断成熟与落地，下一步的工作拟从以下几个方向展开：1)拟将低轨卫星处理任务的成本数据建模为图结构，使用图神经网络提取各个节点的特征向量，基于节点特征向量完成智能化决策；2)将应用场景拓展至灾害应急，把配有高性能任务卸载算法的 LEO 卫星集群部署在地面通信网络架构中，从而为地面通信受干扰的灾害地区提供可靠的计算服务辅助救援任务；3)未来会在系统模型中考虑更多的 QoS 指标(如任务卸载成功率和数据隐私性)，以实现高鲁棒性的调度决策。

参 考 文 献

[1] CHEN W Y, CHOU P Y, WANG C Y, et al. Dual Pricing Optimization for Live Video Streaming in Mobile Edge Computing With Joint User Association and Resource Management [J]. IEEE Transactions on Mobile Computing, 2023, 22(2): 858-873.

[2] DAI X X, XIAO Z, JIANG H B, et al. Task Co-Offloading for D2D-Assisted Mobile Edge Computing in Industrial Internet of Things [J]. IEEE Transactions on Industrial Informatics, 2023, 19(1): 480-490.

[3] ZHU M Y, LI J. Blockchain based on Reliable Task Offloading Strategy for Edge Computing in Smart Home[C]// 2023 8th International Conference on Intelligent Computing and Signal Processing. IEEE, 2023: 1364-1368.

[4] XIA X Y, CHEN F F, HE Q, et al. OL-MEDC: An Online Approach for Cost-Effective Data Caching in Mobile Edge Computing Systems [J]. IEEE Transactions on Mobile Computing, 2023, 22(3): 1646-1658.

[5] PENG Q L, WU C R, XIA Y N, et al. DoSRA: A Decentralized Approach to Online Edge Task Scheduling and Resource Allocation [J]. IEEE Internet of Things Journal, 2022, 9(6): 4677-4692.

[6] KIM J, HAM D, KIM T, et al. Survey on Satellite-Mobile Code Offloading[C]// 2022 13th International Conference on Information and Communication Technology Convergence. IEEE, 2022: 921-923.

[7] FANG X R, FENG W, WEI T, et al. 5G Embraces Satellites for 6G Ubiquitous IoT: Basic Models for Integrated Satellite Terrestrial Networks [J]. IEEE Internet of Things Journal, 2021, 8(18): 14399-14417.

[8] HAN Z Z, XU C, LIU K, et al. A Novel Mobile Core Network Architecture for Satellite-Terrestrial Integrated Network[C]// 2021 IEEE Global Communications Conference. IEEE, 2021: 1-6.

[9] LI X T，XU S，ZHAO Z P，et al. A Survey on Computing Offloading in Satellite-Terrestrial Integrated Edge Computing Networks[C]// 2023 15th International Conference on Communication Software and Networks. IEEE，2023：172-182.

[10] DENBY B，LUCIA B. Orbital Edge Computing：Nanosatellite Constellations as a New Class of Computer System[C]// Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. Association for Computing Machinery，2020：939-954.

[11] DENBY B，LUCIA B. Orbital Edge Computing ：Machine Inference in Space [J]. IEEE Computer Architecture Letters，2019，18(1)：59-62.

[12] LI CC，ZHANG Y S，HAO X K，et al. Jointly optimized request dispatching and service placement for MEC in LEO network[J]. China Communications，2020，17(8)：199-208.

[13] WU J，JIA M，GUO Q，et al . Joint Optimization Computation Offloading and Resource Allocation for LEO Satellite with Edge Computing[C]// 2023 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting. IEEE，2023：1-5.

[14] CHAI F R，ZHANG Q，YAO H P，et al. Joint Multi-Task Offloading and Resource Allocation for Mobile Edge Computing Systems in Satellite IoT [J]. IEEE Transactions on Vehicular Technology，2023，72(6)：7783-7795.

[15] CAO X L，YANG B，SHEN Y L，et al. Edge-Assisted Multi-Layer Offloading Optimization of LEO Satellite-Terrestrial Integrated Networks [J]. IEEE Journal on Selected Areas in Communications，2023，41(2)：381-398.

[16] TANG Q Q，FEI Z S，LI B，et al. Computation Offloading in LEO Satellite Networks With Hybrid Cloud and Edge Computing[J]. IEEE Internet of Things Journal，2021，8(11)：9164-9176.

[17] CUI G F，LONG Y T，XU L X，et al. Joint Offloading and Resource Allocation for Satellite Assisted Vehicle-to-Vehicle Communication[J]. IEEE Systems Journal，2021，15(3)：3958-3969.

[18] MEI C L，GAO C，XING Y J，et al. An Energy Consumption Minimization Optimization Scheme for HAP-Satellites Edge Computing[C]// 2022 IEEE 22nd International Conference on Communication Technology. IEEE，2022：857-862.

[19] SONG Z Y，HAO Y Y，LIU Y W，et al. Energy-Efficient Multi-access Edge Computing for Terrestrial-Satellite Internet of Things [J]. IEEE Internet of Things Journal，2021，8(18)：14202-14218.

[20] PFANDZELTER T，BERMBACH D. Celestial：Virtual Software System Testbeds for the LEO Edge[C]// Proceedings of the 23rd ACM/IFIP International Middleware Conference. Association for Computing Machinery，2022：69-81.

[21] QI X X，ZHANG B，QIU Z L，et al. Using Inter-Mesh Links to Reduce End-to-End Delay in Walker Delta Constellations [J]. IEEE Communications Letters，2021，25(9)：3070-3074.

[23] ZHANG Y R，CHEN C，LIU L，et al. Aerial Edge Computing on Orbit：A Task Offloading and Allocation Scheme [J]. IEEE Transactions on Network Science and Engineering，2023，10(1)：275-285.

[24] DAI X，CHEN X，JIAO L B，et al. Priority-Aware Task Offloading and Resource Allocation in Satellite and HAP Assisted Edge-Cloud Collaborative Networks[C]// 2023 15th International Conference on Communication Software and Networks. IEEE，2023：166-171.

[25] DONG Q，XU X D，HAN S J，et al. DDPG-Based Task Offloading in Satellite-Terrestrial Collaborative Edge Computing Networks[C]// 2023 IEEE International Conference on Communications Workshops. IEEE，2023：1541-1546.

[26] EI N N，YOON J S，HONG C S. Energy-Aware Task Offloading and Resource Allocation in Space-Aerial-Integrated MEC System[C]// 2022 23rd Asia-Pacific Network Operations and Management Symposium. IEEE，2022：1-6.

[27] CHEN B C，LI N，LI Y，et al. Energy Efficient Hybrid Offloading in Space-Air-Ground Integrated Networks[C]// 2022 IEEE Wireless Communications and Networking Conference. IEEE，2022：1319-1324.

[28] CHU W B，JIA X M，YU Z，et al. Joint Service Caching，Resource Allocation and Task Offloading for MEC-based Networks：A Multi-Layer Optimization Approach[J]. IEEE Transactions on Mobile Computing，2024，23(4)：2958-2975.

[29] WANG S H，LIU Y，HUANG X Y，et al. Adaptive large neighborhood search for the dynamic vehicle routing problem with electric vehicles[C]// 2023 35th Chinese Control and Decision Conference. IEEE，2023：3776-3780.

[30] CHE A D，WANG W J，MU X H，et al. Tabu-Based Adaptive Large Neighborhood Search for Multi-Depot Petrol Station Replenishment With Open Inter-Depot Routes [J]. IEEE Transactions on Intelligent Transportation Systems，2023，24(1)：316-330.

[31] HUANG K Y，MA J B，LIU X Y. Research on Vehicle Route Planning with Capacity Limitation Based on Adaptive Large-scale Neighborhood Search Algorithm[C]// 2021 6th International Symposium on Computer and Information Processing Technology. IEEE，2021：6-10.

[32] HAMDI M，HAMED A B，YUAN D，et al. Energy-Efficient Joint Task Assignment and Power Control in Energy-Harvesting D2D Offloading Communications [J]. IEEE Internet of Things Journal，2022，9(8)：6018-6031.

[33] LI Q Y. Research on Key Technologies for Joint Optimization of Access and Backhaul in 6G Dense Network [D]. Nanjing：Nanjing University of Posts and Telecommunications，2023.

WANG Zhongxiao，born in 2003，His main research interests include orbit edge computing and computation task offloading strategy.

PENG Qinglan，born in 1994，Ph.D，lecturer，master supervisor. His main research interests include edge computing，service computing，and cloud computing.