

# 超密集边缘计算网络中面向能耗优化的任务卸载方法

曾蓉晖<sup>1</sup>, 林兵<sup>1</sup>, 王明芬<sup>2</sup>, 林凯<sup>1</sup>, 卢宇<sup>1,2</sup>

(1. 福建师范大学 物理与能源学院, 福州 350000; 2. 福建师范大学 协和学院, 福州 350000)

**摘要:** 传统网络架构部署下的边缘服务器难以满足大规模用户设备的接入和通信质量要求。为增加网络容量, 提高频谱利用率, 通过密集化基站的部署, 构建一种面向超密集边缘计算网络的任务卸载优化模型。面对信道状态的变化、移动设备的动态需求以及服务器和频谱资源的有限性对任务卸载带来的挑战, 结合任务类型和服务器的计算能力, 并考虑信道状态变化、移动设备的动态需求以及干扰约束对卸载策略的影响, 提出一种基于自适应模拟退火遗传(AGASA)算法的任务卸载方法, 在满足任务截止期限的同时, 对任务卸载能耗进行优化。同时, 为得到最优上传功率, 采用黄金分割算法解决功率控制问题, 从而降低传输能耗。实验结果表明, AGASA算法在信道状态变化时可保证通信质量和计算效率, 相比混合遗传粒子群算法, 能够在满足截止期约束的同时使卸载能耗降低 15.56%。

**关键词:** 超密集网络; 移动边缘计算; 任务卸载; 信道分配; 自适应模拟退火遗传算法

开放科学(资源服务)标志码(OSID):



**中文引用格式:** 曾蓉晖, 林兵, 王明芬, 等. 超密集边缘计算网络中面向能耗优化的任务卸载方法[J]. 计算机工程, 2022, 48(11): 39-48.

**英文引用格式:** ZENG R H, LIN B, WANG M F, et al. Task offloading method for energy consumption optimization in ultra-dense edge computing network[J]. Computer Engineering, 2022, 48(11): 39-48.

## Task Offloading Method for Energy Consumption Optimization in Ultra-Dense Edge Computing Network

ZENG Ronghui<sup>1</sup>, LIN Bing<sup>1</sup>, WANG Mingfen<sup>2</sup>, LIN Kai<sup>1</sup>, LU Yu<sup>1,2</sup>

(1. College of Physics and Energy, Fujian Normal University, Fuzhou 350000, China;

2. Concord University College, Fujian Normal University, Fuzhou 350000, China)

**[Abstract]** The edge server deployed in a conventional network architecture exhibits difficulty meeting the requirements of large-scale user equipment access and communication quality. To increase network capacity and improve spectrum utilization, dense base station deployment is combined with Ultra-Dense Network (UDN) to develop a task offloading optimization model for an ultra-dense edge computing network. The reasons for changes in channel status, the dynamic requirements of mobile devices, and the limitations of servers and spectrum resources pose challenges for offloading. A genetic algorithm based on an Adaptive Genetic Algorithm with Simulated Annealing (AGASA)'s task offloading method optimizes the energy consumption of task offloading while meeting the task deadline by combining the task type and the computing power of the server and considering the influence of channel state changes, mobile device dynamic requirements, and interference constraints on the offloading strategy. Meanwhile, to improve upload power, this study solves the power control problem with the golden section algorithm, saving transmission energy consumption. The experimental results demonstrate that when the channel state changes, the proposed task offloading strategy ensures communication quality and computational efficiency. It can meet deadline constraints while reducing its offloading energy consumption by 15.56% when compared to the hybrid genetic particle swarm algorithm (GAPSO).

**[Key words]** Ultra-Dense Network (UDN); Mobile Edge Computing (MEC); task offloading; channel allocation; Adaptive Genetic Algorithm with Simulated Annealing (AGASA) algorithm

**DOI:** 10.19678/j.issn.1000-3428.0063104

**基金项目:** 国家重点研发计划(2018YFB1004800); 福建省高校产学研合作项目(2021H6026); 福建省自然科学基金(2019J01286, 2019J01244, 2018J01619); 福建省教育厅中青年教育科研项目(JT180098); 福建省社会科学规划青年项目(FJ2020C025); 福建师范大学协和学院智能计算与应用团队项目(2020-TD-001)。

**作者简介:** 曾蓉晖(1996—), 女, 硕士研究生, 主研方向为超密集边缘计算; 林兵, 副教授、博士; 王明芬, 副教授; 林凯, 硕士研究生; 卢宇(通信作者), 教授。

**收稿日期:** 2021-11-01 **修回日期:** 2021-12-16 **E-mail:** fzluyu@163.com

## 0 概述

2022年,全球移动设备的数量将会达到123亿台<sup>[1]</sup>。随着智能汽车、手机、无人机等一些移动设备数量的爆炸式增长,移动应用的数量和类型也在不断地增加,这些移动应用通常需要强大的计算资源才能在响应截止期内被执行完成,这对移动设备的计算能力和电池寿命都提出了严峻的挑战<sup>[2]</sup>。

移动边缘计算(Mobile Edge Computing, MEC)作为一种新的计算框架,可满足移动用户低时延高可靠的计算需求<sup>[3]</sup>。MEC服务器通常部署在距离用户较近的位置,从而扩展移动设备的计算能力,将移动设备产生的计算任务从移动设备卸载到网络边缘,从而有效缓解移动设备的计算压力,降低任务卸载的能耗和时延<sup>[4]</sup>。然而边缘服务器的计算资源有限,传统蜂窝无线网络部署下的MEC很难满足大规模用户设备的接入和通信质量要求,传输过程中可分配的无线频谱资源也会变得更加稀缺。为了更好地应对5G和物联网时代对计算资源和频谱资源的激增要求,迫切需要更先进的组网方式和无线接入技术来增加网络容量,提高频谱利用率<sup>[5]</sup>。因此,研究者在此基础上引入超密集网络(Ultra-Dense Network, UDN)。

UDN具有强大的接入能力,能够在传统宏小区覆盖的基础上,加密小型小区的部署,从而为移动设备提供足够的频谱资源,此外,通过重用小区间的有限频谱,能够显著提高网络频谱利用率和网络容量<sup>[6]</sup>。在UDN的基站上部署MEC服务器,可以形成超密集边缘计算网络,其中小基站(Small Base Station, SBS)的密集部署能够覆盖更多移动设备,为它们提供接入服务,打破传统宏网络单层覆盖的缺陷,提高网络容量,增加频谱效率;而连接小基站的MEC服务器则为移动设备提供了丰富的计算资源。文献[7]也阐述了在UDN中,移动运营商通过部署大量宏基站和小功率的微基站为移动设备提供服务,能够应对大规模设备连接和数据流量,有效降低系统能耗。然而超密集边缘计算网络中的任务卸载也存在一系列的挑战。首先,每个连接MEC服务器的SBS同时为多个移动设备提供接入服务时,需要在考虑信道状态变化的同时,将计算资源分配给不同的服务请求;其次,移动设备需要选择对应的信道将任务卸载到MEC服务器上,大量移动设备选择同一个信道竞争有限的频谱资源时,将会产生设备间干扰,影响通信质量;最后,在任务传输过程中,任务执行延迟和能耗会受到发射功率的影响。

近年来,关于超密集边缘计算网络中的任务卸载研究大多集中在计算卸载和资源分配的联合优化:文献[8]提出一种新的卸载方案和资源分配方案,将多个任务卸载到同一个边缘服务器,在计算资源和延迟的约束下降低了卸载能耗;文献[9]利用凸

优化的方法研究TDMA和OFDMA系统中部分卸载的最优策略,提出一种基于门限的卸载优先级函数;文献[10]则仅在TDMA系统中用凸优化研究了多用户接入单个边缘服务器场景下进行计算任务卸载决策的最优策略。然而,这些研究多数只适用于单个服务器的情况,而在实际超密集边缘计算网络的场景中,往往需要考虑多个服务器部署的情况。对此:文献[11]研究系统节能卸载方案,提出一种遗传算法和粒子群算法的优化算法;文献[12]研究多个用户通过多个小基站共享一个边缘服务器的场景,考虑不同用户与小基站通信时的干扰问题,提出一种分步优化的卸载决策,计算资源分配以最小化系统能耗和时延的加权和,同时利用图染色法解决无线资源分配的问题并最小化用户间干扰。

对于混合性能指标的问题,在计算和通信过程中,时延和能耗需要进行互相妥协。然而在实际的超密集边缘计算网络场景下,一些服务请求往往需要对时延进行硬性约束。为此,许多研究侧重于将时延约束下能耗最小化作为优化目标:为了解决边缘服务器计算资源有限以及传输干扰等问题,文献[13]针对多移动设备场景下的卸载问题,在考虑信道和计算资源成本的同时,通过制定有效的资源分配策略,最小化用户的总消耗;文献[14]提出一种基于博弈的任务卸载算法,解决了在小小区网络架构下的计算卸载问题;文献[15]讨论了物联网设备下的动态任务卸载问题,在软件定义接入网络中提出了一种基于物联网设备的任务卸载方案,最后通过线性化方法,将问题转化为整数线性规划问题,最终所提出的方案可以有效降低平均延迟和能量消耗。

以上研究提出了许多计算资源分配和计算卸载的方案,也考虑了用户与基站通信时的干扰问题,但忽略了信道状态动态变化时对卸载结果的影响,以及发射功率对移动设备与基站通信过程的影响。本文重点考虑信道状态、发射功率等因素,讨论任务传输干扰对卸载结果的影响,设计一种基于自适应模拟退火遗传(Adaptive Genetic Algorithm with Simulate Annealing, AGASA)算法的任务卸载策略,在满足截止期约束的同时,对任务卸载能耗进行优化,并通过黄金分割算法求解最优上传功率。

## 1 系统模型

### 1.1 网络模型

本文主要讨论一个宏基站覆盖范围下, $N$ 个SBS部署的超密集边缘计算网络的场景,考虑将大量用户设备(User Equipment, UE)产生的任务请求卸载到连接SBS的MEC上以及本地处理,具体的系统框架如图1所示。其中,宏基站是整个网络的信息中心,可以收集整个网络的信息,而宏基站和各个SBS之间可共享整个信道的频谱资源。同时,每个UE每

次只能将任务请求发送到一个SBS上,SBS与一个边缘服务器相连接,边缘服务器表示为 $S=\{s_1, s_2, \dots, s_n\}$ ,同时每个服务器的计算资源每次只能分配给一个UE。

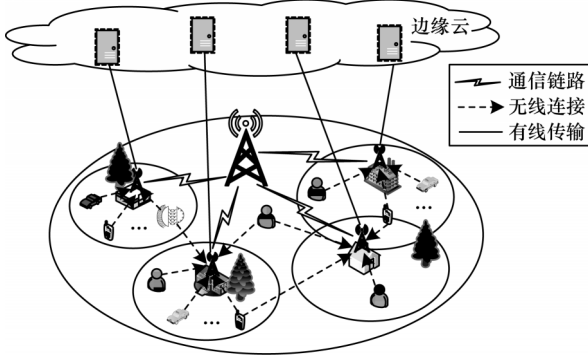


图1 超密集边缘计算网络架构

Fig.1 Ultra-dense edge computing network architecture

根据正交频分复用技术,本文将每个SBS覆盖范围内的信道划分成 $C$ 个子信道,并将信道描述为 $K=\{k_1, k_2, \dots, k_C\}$ ,不同信道之间互不干扰。每个移动设备产生一个计算任务,且任务卸载过程中UE不移动。因为不同小区的UE可能会复用相同的子信道,所以不同用户在相同的信道传输任务时会产生用户间的干扰。在本文系统中,UE的集合表示为 $U=\{U_1, U_2, \dots, U_M\}$ ,对应的UE产生的任务集合表示为 $T=\{T_1, T_2, \dots, T_m\}$ ,服务器和UE的计算资源表示为 $f=\{f_m, f_{s1}, f_{s2}, \dots, f_{sn}\}$ 。

### 1.2 任务模型

由于每个设备产生一个任务,因此将 $T_i=(\omega_i, s_i, d_i, t_{i\_deadline})$ 表示为一个UE产生的计算任务。其中: $\omega_i$ 表示完成任务 $T_i$ 所需的计算量大小; $s_i$ 表示每个任务 $T_i$ 的数据量大小,为产生任务 $T_i$ 的UE到SBS的距离; $t_{i\_deadline}$ 表示完成任务 $T_i$ 的截止时间。在超密集边缘计算网络中,每个UE都有一个计算任务需要被处理,因此,定义二元决策变量 $x_{i,j}=\{0,1\}$ 表示任务的卸载决策, $x_{i,j}=0$ 表示任务 $T_i$ 选择本地处理, $x_{i,j}=1$ 表示任务 $T_i$ 需要卸载到服务器 $s_j$ 。此外,定义二元决策变量 $\gamma_{i,k}=\{0,1\}$ 表示任务 $T_i$ 的信道分配决策, $\gamma_{i,k}=1$ 表示任务 $T_i$ 通过信道 $k$ 传输, $\gamma_{i,k}=0$ 表示不通过信道传输。

### 1.3 通信模型

当UE选择将任务 $T_i$ 卸载到第 $n$ 个SBS连接的服务器 $s_n$ 上,不同移动用户共享相同信道频谱资源时, $U_i$ 会接收到来自其他移动用户的传输干扰。因此,由移动用户 $U_i$ 产生的任务 $T_i$ 在信道 $k$ 上传输时的有效干扰表示为:

$$I_{i,k} = \frac{g_{i,k}}{\sigma^2 + \sum_{n,k} r_{n,k} P_n g_{n,s}} \quad (1)$$

其中: $g_{i,k}$ 表示任务 $T_i$ 在信道 $k$ 上传输时对应的信道

增益; $\sigma^2$ 表示噪声功率谱密度。

因此,任务 $T_i$ 在信道 $k$ 传输时的信噪比为:

$$S_{i,k}^{\text{SINR}} = P_{s,i} \frac{g_{i,k}}{\sigma^2 + \sum_{n,k} r_{n,k} P_n g_{n,s}} \quad (2)$$

通过计算,得到UE产生的任务 $T_i$ 在信道 $k$ 上的传输速率为:

$$R_{i,k} = W \log(1 + I_{i,k} P_{i,k}) \quad (3)$$

其中: $W$ 表示子信道的带宽; $P_{i,k}$ 表示任务 $T_i$ 的上传功率。

### 1.4 计算模型

当计算任务选择本地处理时,需要产生任务请求 $T_i$ 的移动设备 $U_m$ 提供计算资源。令 $f_m$ 表示移动设备 $U_m$ 的计算能力,并将移动设备 $U_m$ 的本地计算功率表示为 $P_m$ ,则任务本地处理时的能耗 $E_{i,0}$ 和计算时间 $T_{i,0}^{\text{comp}}$ 分别表示为:

$$E_{i,0} = P_m \frac{\omega_i}{f_m} \quad (4)$$

$$T_{i,0}^{\text{comp}} = x_{i,j} \frac{\omega_i}{f_{s,j}} \quad (5)$$

当选择卸载计算时,通过无线网络传输任务时会产生相应的传输能耗和计算能耗,其中传输能耗 $E_{i,k}^s$ 、传输时间 $T_{i,k}^{\text{comm}}$ 、计算能耗 $E_{i,j}^{\text{comp}}$ 和计算时间 $T_{i,j}^{\text{comp}}$ 分别表示为:

$$E_{i,k}^s = \gamma_{i,k} P_{i,k} \frac{s_i}{R_{i,k}} \quad (6)$$

$$T_{i,k}^{\text{comm}} = \gamma_{i,k} \frac{s_i}{R_{i,k}} \quad (7)$$

$$E_{i,j}^{\text{comp}} = x_{i,j} P_0 \frac{\omega_i}{f_{s,j}} \quad (8)$$

$$T_{i,j}^{\text{comp}} = x_{i,j} \frac{\omega_i}{f_{s,j}} + (1 - x_{i,j}) \frac{\omega_i}{f_m} \quad (9)$$

将选择本地处理的总能耗表示为 $E_m$ ,那么对于超密集边缘计算网络中所有的UE,此时的任务总能耗为:

$$E_m = \sum_{i=1}^M E_{i,j}^{\text{comp}}(x_{i,0}) = \sum_{i=1}^M P_m \frac{\omega_i}{f_m} \quad (10)$$

同时,将卸载的用户总能耗表示为 $E_c$ ,在整个超密集边缘计算网络中,所有需要将UE产生的任务请求卸载到MEC的总能耗,包括将任务请求发送到SBS上的传输能耗,以及在MEC上处理的计算能耗两部分,那么卸载任务的总能耗为:

$$E_c = \sum_{i=1}^M [E_{i,j}^{\text{comp}}(x_{i,j}) + E_{i,k}^s(\gamma_{i,j})] \quad (11)$$

卸载任务产生的总时延包括传输时延和MEC上的计算时延,表示为:

$$T^z = \sum_{i=1}^M [T_{i,k}^{\text{comm}} + T_{i,j}^{\text{comp}}] \quad (12)$$

由于下行链路的传输数据量很小,对整体卸载结果影响不大,因此本文不考虑接收返回数据时的能耗和时延。

## 2 问题描述与解决方案

### 2.1 基于信道状态的卸载决策优化

移动设备通过无线信道向 MEC 服务器传输数据,当任务通过无线信道上传到 SBS 上时,基站密集部署下网络环境比较复杂,而实际任务传输过程中信道状态是不断变化的,因此,信道状态的好坏会影响当前任务的传输速率。为了更好地描述实际信道状态变化对任务传输过程通信质量的影响,本文将超密集边缘计算网络中的无线信道拟合成 Gilbert Elliot 模型<sup>[16]</sup>,提出一种基于信道状态的任务卸载方案。首先假设在时间片  $t$  内检测到的无线信道的状态为  $g_t$ ,当前信道状态较差时,信道状态描述为  $g_B$ ,当前信道状态较好时,信道状态描述为  $g_G$ 。将  $g_t = g_G$  时对应的传输速率描述为  $R_t = R_G$ ,同时将  $g_t = g_B$  时对应的传输速率描述为  $R_t = R_B$ 。此外,为了具体描述超密集边缘计算网络中实际的信道状态,用状态转移概率来表示实时信道状态变化的情况,当信道状态从差变成好时,信道状态转移概率为  $P_{BG}$ ,而当信道状态从好变成差时,信道状态转移概率描述为  $P_{GB}$ 。因此,信道状态较好时信道状态的稳定概率为  $\frac{P_{BG}}{P_{BG} + P_{GB}}$ ,信道状态不好时的稳定概率为  $\frac{P_{GB}}{P_{BG} + P_{GB}}$ 。根据信道状态的稳定概率,可以得到信道传输速率的期望值为:

$$e_{xp}(R) = \frac{P_{BG}}{P_{BG} + P_{GB}} R_G + \frac{P_{GB}}{P_{BG} + P_{GB}} R_B \quad (13)$$

在任务卸载过程中,由于信道状态是实时变化的,当信道状态不好时,数据传输速率较低,因此会产生更多的传输能耗和时延;同时,随着接入 SBS 的移动设备  $U_i$  数量增加,任务卸载能耗和时延也会随之增加。所以,本文根据当前超密集边缘计算网络的信道状态、任务属性和所需计算资源制定了相应的卸载方案。在该方案中,当任务  $T_i$  初始化选择本地处理时,如果满足式(14),则表明综合考虑计算任务  $T_i$  所需计算资源以及当前信道状态情况,选择该任务上传到服务器  $S_m$  能够在保证通信质量的同时节约计算资源。因此,任务卸载方案需要更新为卸载到服务器上。

$$\frac{s_i}{\omega_i} < \frac{e_{xp}(R)}{f_m} \quad (14)$$

同理,如果任务  $T_i$  初始化选择卸载到 MEC 上处理,那么当任务类型满足式(15),则表明针对任务  $T_i$ ,相比于本地计算,卸载到 MEC 可能无法保证通信质量,更好地节约计算资源,则当前的卸载方案需要更新为本地处理。

$$\frac{s_i P_{si}}{\omega_i} \geq \frac{e_{xp}(R) P_{i,k}}{f_{s,n}} \quad (15)$$

上述方案基于当前的信道状态和任务类型不断更新卸载决策,以保证卸载的任务能够在信道状态良好的情况下传输和处理,最终得到尽可能优的卸

载决策,降低任务卸载的总能耗。

### 2.2 卸载用户的信道分配方案

由于每个任务只能选择一个信道进行传输,而任务传输能耗主要受传输速率和信道分配决策的影响,因此为尽可能降低任务传输过程的能耗,本文给出了信道分配决策的优化方案。由于同一基站服务范围内的移动用户以及其他小区的移动用户可能会共用相同的频谱资源,从而产生信道干扰,因此为保证任务请求  $T_i$  通过信道  $k$  的传输速率,需要对传输过程的信道干扰大小进行一定的限制。由此,根据约束条件 C1(下文给出)可得,任务  $T_i$  有效干扰需要满足:

$$I_{i,k} \geq \frac{1}{P_{i,k}} \left( 2^{\frac{1}{W_{s,i}}} - 1 \right) \quad (16)$$

其中:

$$\zeta_i = t_{i\_deadline} - \frac{\omega_i}{\gamma_{i,k} f_{s,j}} \quad (17)$$

最后得到信道分配方案  $\gamma_{i,k}$ ,如式(18)所示,表明在任务传输过程中,需要选择传输能耗最低且满足约束条件 C6(下文给出)的信道分配决策。

$$\gamma_{i,k} = 1 \mid_{k = \arg \min E \cap I_{i,k} \geq \frac{1}{P_{i,k}} \left( 2^{\frac{1}{W_{s,i}}} - 1 \right)} \quad (18)$$

### 2.3 卸载用户的上传功率控制方案

同理,由截止时间的约束条件 C1(下文给出)可以得到最小任务上传功率如下:

$$P_{i,k} \geq \frac{1}{I_{i,k}} \left( 2^{\frac{1}{W_{s,i}}} - 1 \right) \quad (19)$$

因此,上传功率的范围为  $\left[ \frac{1}{I_{i,k}} \left( 2^{\frac{1}{W_{s,i}}} - 1 \right), P_{\max} \right]$ 。

因为上传功率主要影响传输能耗,所以将传输能耗作为上传功率的函数,并利用黄金分割算法求解最优发射功率。上传功率控制问题的优化目标为:

$$E_i^s = \gamma_{i,k} P_{i,k} s_i / W \ln(1 + S_{i,k}^{\text{SINR}})$$

$$\forall i \in M, \frac{2^{\frac{s_i}{W_{s,i}}} - 1}{I_{i,k}} \leq P_{i,k} \leq P_{\max} \quad (20)$$

为保证传输质量,首先对发射功率的范围进行约束,将最小上传功率表示为  $a_i = (2^{\frac{s_i}{W_{s,i}}} - 1) / I_{i,k}$ ,保证移动设备能够将任务请求传输到一定距离的 SBS 上,将最大上传功率表示为  $b_i = P_{\max}$ ;其次,为了找到使得传输能耗更低的上传功率,需要进一步缩小上传功率的范围,上传功率上下界搜索值的更新策略分别为  $P_{i,k}^l = b_i - \tau(b_i - a_i)$ ,  $P_{i,k}^r = a_i + \tau(b_i - a_i)$ ;接着,由式(20)计算对应的目标函数值,并判断是否满足  $|P_{i,k}^r - P_{i,k}^l| > \varepsilon$ ,若满足以上条件,则根据  $P_{i,k} = (P_{i,k}^r + P_{i,k}^l) / 2$  可得到最优发射功率;若不满足以上条件,则后续需要判断功率搜索边界值对应的传输能耗是否满足  $E^s(P_{i,k}^l) > E^s(P_{i,k}^r)$ ,若满足以上条件,则需要进一步缩小功率的左边界,若不满足以上条件,则需要进一步缩小功率的右边界,直到满足  $|P_{i,k}^r - P_{i,k}^l| > \varepsilon$ ,再根据

$P_{i,k} = (P_{i,k}^r + P_{i,k}^l)/2$  输出最优发射功率。其中,  $\tau = 0.618$ 。该算法能够将搜索范围不断缩小,保证在一定功率范围内获取每个请求的最优上传功率。

#### 2.4 优化目标

因为本文的目标是在截止时间内最小化任务卸载的总能耗,所以将超密集边缘计算网络中所有移动设备任务卸载的总能耗作为优化目标。在该网络中,一些UE选择任务请求进行本地处理,则总能耗为  $E_{i,0}^{\text{comp}}$ ;一些UE选择将任务请求卸载到连接SBS的边缘服务器上,则总能耗由传输能耗  $E_{i,k}^s$  和计算能耗  $E_{i,j}^{\text{comp}}$  两部分组成。由于优化目标是一个单目标多变量的问题,为了降低问题的复杂性,利用分治的思想将优化目标分解成任务卸载、信道分配和功率控制三个子问题。因此,该网络中的总能耗可表示为:

$$\begin{aligned} P: \min E &= \min \sum_{i=1}^M (E_{i,0}^{\text{comp}} + E_{i,j}^{\text{comp}} + E_{i,k}^s) \\ \text{s.t. C1: } T^z &\leq \sum_{i=1}^M t_{i,\text{deadline}} \\ \text{C2: } \forall T_i \in T, S_{i,k}^{\text{SINR}} &\geq S_s^{\text{SINR}} \\ \text{C3: } \forall i \in M, \frac{2^{\frac{S_i}{W_{i,k}}} - 1}{EI_{i,k}} &\leq P_{i,k} \leq P_{\max} \\ \text{C4: } \sum_{j=1}^N x_{i,j} &\leq 1, \forall i \in M, j \in N \\ \text{C5: } \sum_{k=1}^C \gamma_{i,k} &\leq 1, \forall i \in M, k \in C \\ \text{C6: } I_{i,k} &\geq \frac{1}{P_{i,k}} \left( 2^{\frac{1}{W_{i,k}}} - 1 \right) \\ \text{C7: } \gamma_{i,k} &= 1 \Big|_{k = \arg \min E \cap I_{i,k} \geq \frac{1}{P_{i,k}} \left( 2^{\frac{1}{W_{i,k}}} - 1 \right)} \end{aligned} \quad (21)$$

其中:C1表示完成任务  $T_i$  的最大可容忍时延;C2表示传输过程中任务  $T_i$  在信道  $k$  的信噪比大小大于阈值,才能保证传输过程的可靠性;C3表示上传功率的范围;C4、C5保证了一个任务只能卸载到一个服务器上处理,同时一个任务也只能通过一个信道传输;C6表示任务的有效干扰需要大于阈值,才能保证任务传输质量;C7是信道选择的条件。

### 3 算法设计与分析

#### 3.1 基于黄金分割算法的功率控制方案

发射功率越高,任务可传输的距离越远,一旦发射功率过高,那么传输过程中会产生更多的能耗,而发射功率过低时,移动设备产生的任务请求无法及时到达对应的SBS。为了更好地优化传输能耗,需要将任务上传功率控制在合理范围内。针对上传功率控制的问题,本文采用一种黄金分割算法,以确定最优的发射功率<sup>[17]</sup>,以期通过较少的迭代次数找到最优解,得到使传输能耗最低的上行功率极值。黄金分割算法描述如算法1所示。

##### 算法1 黄金分割算法

输入  $a_i = (2^{\frac{S_i}{W_{i,k}}} - 1)/EI_{i,k}$ ,  $b_i = P_{\max}$ ,  $\tau = 0.618$ ,  $\varepsilon = 0.5$

输出  $P_{i,k}$

```
1. 计算  $P_{i,k}^l = b_i - \tau(b_i - a_i)$ ,  $P_{i,k}^r = a_i + \tau(b_i - a_i)$ 
2. While  $|P_{i,k}^r - P_{i,k}^l| > \varepsilon$  do
3. 计算  $E^s(P_{i,k}^l)$ ,  $E^s(P_{i,k}^r)$ 
4. if  $E^s(P_{i,k}^l) > E^s(P_{i,k}^r)$ 
5.  $a_i = P_{i,k}^l$ ,  $P_{i,k}^l = P_{i,k}^r$ ,  $E^s(P_{i,k}^l) = E^s(P_{i,k}^r)$ ,  $P_{i,k}^r = a_i + \tau(b_i - a_i)$ 
6. else:
7.  $b_i = P_{i,k}^r$ ,  $P_{i,k}^r = P_{i,k}^l$ ,  $E^s(P_{i,k}^r) = E^s(P_{i,k}^l)$ ,  $P_{i,k}^l = b_i - \tau(b_i - a_i)$ 
8. End if
9. End While
10.  $P_{i,k} = (P_{i,k}^r + P_{i,k}^l)/2$ 
11. 输出  $P_{i,k}$ 
```

#### 3.2 基于AGASA算法的任务卸载方法

研究人员常采用传统智能算法来解决优化问题<sup>[18-19]</sup>,而在超密集边缘计算网络的场景下,利用传统智能算法同时求解任务卸载决策和信道分配决策难度较大,因此,本文考虑在自适应遗传算法(Adaptive Genetic Algorithm, AGA)的基础上结合模拟退火(Simulate Annealing, SA)算法,提出AGASA算法,在更短时间内得到卸载方案的最优解。

AGA算法<sup>[20]</sup>往往具有较强的全局搜索能力,但是在算法开始阶段,由于个体间的差异较大,父代产生的子代个数与父代个体的适应度值成正比,一些优秀的个体则充斥着整个子代种群,导致算法在后期形成“早熟”的现象,最终陷入局部最优解。而SA算法是根据物理退温降火的原理建立的算法,该算法具备较强的局部搜索能力,并且能够很好地跳出局部搜索的范围。但是该算法对于整个搜索空间的掌握不够全面,导致算法的搜索效率较低<sup>[21]</sup>。

AGASA算法能够很好地弥补以上两种算法的不足。首先将AGA算法初始化的种群适应度值作为模拟退火的初始解,将交叉变异后的适应度值作为模拟退火的新解。将根据模拟退火的更新规则得到的新解对应的卸载方案作为AGA算法下一代的初始方案。将AGA算法和SA算法结合能够克服彼此的缺陷,AGASA算法不仅在效率上优于传统的遗传算法,而且还提高了全局搜索能力,更好地找到卸载最优解。AGASA算法描述如算法2所示。

##### 算法2 AGASA算法

输入 种群大小  $N$ , 迭代次数, 设备数, 小基站数, 截止时间, 上传功率

输出 卸载计划  $X$ , 信道分配方案  $\gamma$

```
1. For  $i = 1$  to  $N$  do:
2. 产生初始种群, 生成初始化的卸载决策  $x_{i,j}$ , 信道分配决策  $\gamma_{i,k}$ 
3. End for
4. 根据算法1更新上传功率
5. For  $i = 1$  to  $N$  do:
6. 根据条件式(14)和式(15)更新卸载决策
7. 根据条件式(18)更新信道分配决策
8. 根据式(21)计算种群和每个个体的适应度值
9. End for
```

10. 获取初始的最优适应度值以及每个个体的最优适应度值  
 11. While  $i < D_{\text{iters}}$  do  
 12. For  $j = 1$  to  $N$  do  
 13. 通过选择、交叉、变异,更新任务卸载方案  
 14. 根据式(22)计算种群和每个个体的适应度值  
 15. 比较更新前后的适应度值  
 16. if  $F_{\text{new}} < F$   
 17. 保留更新后的适应度值  
 18. else:  
 19. 根据退火概率决定是否保留更新前的适应度值  
 20. End for  
 21. 从  $N$  个卸载计划中找到最优的卸载计划  
 22. For  $j = 1$  to  $N$  do  
 23. 根据式(25)和式(26)计算交叉概率和变异概率  
 24. 根据式(24)计算退火概率  
 25. End for  
 26. End While  
 27. 返回最优的卸载计划  $X$  和信道分配方案  $\gamma$

在初始化过程中,首先将可能的计算卸载方案作为遗传算法中的染色体,将每个任务对应的位置作为染色体的基因,那么每个染色体共有  $M$  个基因。编码方式为整数编码,每个任务可以选择卸载到任意一个服务器上,并选择对应的一个信道去传输,也可以选择本地处理。同时,为了衡量每个种群对生存环境的适应性,还可以通过适应度值来判断并淘汰适应程度更差的种群。适应度值表示为:

$$F = \begin{cases} E, T < \sum_{i=1}^M t_{i\_deadline} \\ E + l \left( T - \sum_{i=1}^M t_{i\_deadline} \right), T \geq \sum_{i=1}^M t_{i\_deadline} \end{cases} \quad (22)$$

当任务完成时间超过该任务的最大截止时间时,在能耗的基础上加入惩罚函数<sup>[22]</sup>:  $l = 10^{-2.5}$ ,适应度函数值为能耗和惩罚函数值之和。当任务完成时间小于截止时间,将适应度值设置为能耗的值。 $t_{i\_deadline}$  表示单个任务的截止时间,本文通过以下的方式计算截止时间:

$$\sum_{i=1}^M t_{i\_deadline} = D_l + k_1(D_u - D_l) \quad (23)$$

其中:  $D_l = T^z$  表示 AGASA 算法下的任务完成时间;  $D_u = 3T^z$ ;  $k_1$  是截止时间比例,取值范围为  $0 \sim 1$ <sup>[23]</sup>。在 AGASA 算法的选择过程中,为了防止最优个体在下一代发生丢失,导致无法收敛到最优解,本文主要采用二元锦标赛结合精英选择的选择策略<sup>[24]</sup>。首先保留当前种群中最优的个体进入子代;然后以一定的选择概率在种群中选择两个个体,将适应度值较低的个体加入到子代,直到所有个体都完成锦标赛。在交叉过程中,在父代染色体的前半段和后半段分别选择两个交叉点,进行两点交叉,形成子代染色体。在变异过程中,选择单点变异的策略。交叉变异结束后,需要比较变异前后种群的适应度值,如果变异前的适应度值更低,那么以一定的退火概率接受变异前的个体,退火概率设置为:

$$P_c = e^{(F_{\text{new}} - F)/T_{\text{em}}} \quad (24)$$

将交叉变异后的适应度值作为最终的适应度值  $F_{\text{new}}$ 。此外,交叉概率和变异概率对算法的收敛性有很大的影响。虽然交叉概率较大时种群更容易产生新个体,但是当其变大时,优良个体在种群中保留率也相应降低;而对变异概率而言,若其过大则该算法相当于普通的随机算法,失去了 AGASA 算法的意义。所以,为了提高算法的性能,本文采取动态变化的交叉概率和变异概率,如式(25)和式(26)所示:

$$P_c = \begin{cases} P_{c1} - \frac{(P_{c1} - P_{c2})(f' - f_{\text{average}})}{f_{\text{max}} - f_{\text{average}}}, f' \geq f_{\text{average}} \\ P_{c1}, f' < f_{\text{average}} \end{cases} \quad (25)$$

$$P_m = \begin{cases} P_{m1} - \frac{(P_{m1} - P_{m2})(f' - f_{\text{average}})}{f_{\text{max}} - f_{\text{average}}}, f' \geq f_{\text{average}} \\ P_{m1}, f' < f_{\text{average}} \end{cases} \quad (26)$$

其中:  $f_{\text{average}}$  表示平均适应度值;  $f'$  表示当前个体的适应度值;  $f_{\text{max}}$  表示最大适应度值;  $P_{c1}$ 、 $P_{c2}$  分别表示交叉概率;  $P_{m1}$ 、 $P_{m2}$  分别表示变异概率。

### 3.3 算法的时间复杂度分析

对 AGASA 算法的时间复杂度进行分析。由于初始化种群大小为  $N$ ,任务请求的数量为  $M$ ,迭代次数为  $D_{\text{iters}}$ ,一条染色体的长度为  $2 \times M$ ,那么 AGASA 算法的时间复杂度分析如下:初始化种群的时间复杂度为  $O(M \times 2 \times N) \approx O(M \times N)$ 。由于初始化过程只迭代一次,因此可以忽略不计;适应度值计算的时间复杂度可以表示为  $O(M \times N)$ ;选择过程的时间复杂度为  $O(M \times M)$ ;交叉过程的时间复杂度为  $O(M/2) = O(M)$ ;变异过程的时间复杂度为  $O(M)$ ;模拟退火过程的时间复杂度为  $O(M)$ 。综上可知,最终整个 AGASA 算法的时间复杂度为  $O(D_{\text{iters}} \times M \times (M + N))$ 。

### 3.4 超密集边缘计算网络卸载过程

20个移动设备随机分布在两层的超密集边缘计算网络中,在该网络有1个MBS和3个SBS,每个小区覆盖范围内的移动用户任务卸载过程如图2所示,具体步骤如下:

1) 初始化移动设备产生的任务请求,包括任务卸载的位置和信道位置。假设有5个移动设备分别产生5个任务请求,该请求的卸载位置如图3所示,其中,00表示卸载位置为本地设备端,不通过传输信道传输,12表示卸载位置为SBS1,传输信道为信道2,以此类推。

2) 由算法1得到最优任务上传功率,并计算初始卸载策略下的总能耗。

3) 由算法2更新卸载方案和任务卸载方案,并计算和比较更新前后的总能耗,得到最能耗最低的卸载方案,更新后的结果如图4所示,任务请求2的卸载方案由12更新为11,任务请求3的卸载方案由11更新为21,任务请求4的卸载方案由21更新为00,任务请求5的卸载方案由00更新为22。

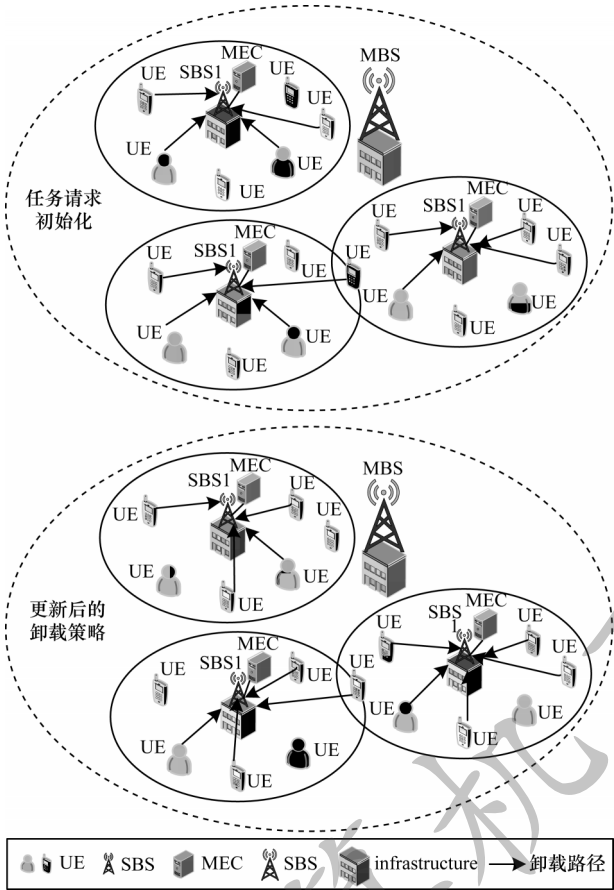


图2 超密集边缘计算网络卸载策略

Fig.2 Offloading strategy of ultra-dense edge computing network

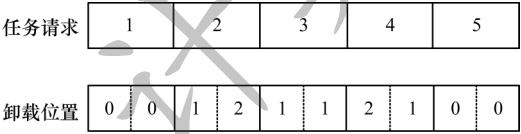


图3 初始化卸载位置

Fig.3 Initial uninstalled location

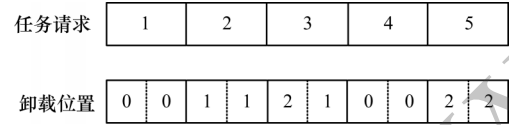


图4 更新后的卸载位置

Fig.4 Uninstall location after update

4 实验设置与结果分析

4.1 实验参数设置

使用 Python3.7 实现算法,并在一台配备 8 GB-RAM 的 Intel I7-2.6 GHz 的 PC 上进行实验,系统参数设置如表 1 所示。考虑有多个 SBS 的超密集边缘计算网络的场景,其中 SBS 的数量设置为 30 个,每个 SBS 对应 1 个 MEC 服务器,则 UDN 由 30 个 SBS 和 1 个 MBS 组成,每个小区的信道频谱被分成多个子频段,并随机分布在 UDN 内的任意位置。将噪声功率谱密度设置为 -100 dB,子信道的带宽设置为 [0.1 MHz, 0.5 MHz],本地计算资源  $f_0 = 0.1$  MHz,服

务器的计算资源为  $f_m = [0.15 \text{ MHz}, 1 \text{ MHz}]$ 。移动设备与 SBS 之间的无线连接采用的则是瑞利衰落信道,路径损耗为  $38.46 + 20 \lg l$  dB。假设距离以 m 为单位,那么设备到 SBS 的距离为 [0 m, 100 m]。将最大上传功率设置为 2 W,在实验次数为 20 次的情况下,取平均值作为最终结果。

表 1 系统参数设置

Table 1 System parameter setting

参数	参数含义	取值
$P_m/\text{W}$	移动设备的计算功率	0.5
$P_0/\text{W}$	移动设备的空载功率	0.001
$P_s/\text{W}$	最大发射功率	2
$R_G/(\text{kb} \cdot \text{s}^{-1})$	信道状态好时的传输速率	100
$R_B/(\text{kb} \cdot \text{s}^{-1})$	信道状态差时的传输速率	5
$N_{\text{SBS}}$	小基站数	30
$C$	每个小区的信道数	[1, 3]
$f_0/\text{MHz}$	移动设备的计算能力	0.1
$f_m/\text{MHz}$	服务器的计算能力	[0.1, 0.5]
$K/\text{MHz}$	子信道带宽	[0.1, 0.5]
$N_{\text{MEC}}$	服务器数	30
$U_{\text{POP}}$	种群数	100

4.2 对比方案

为评估 AGASA 算法的性能,本文将该算法与遗传算法 (Genetic Algorithm, GA)<sup>[24]</sup> 和混合遗传粒子群 (GAPSO) 算法<sup>[25]</sup> 进行对比,并结合本文的应用场景进行相应的调整。同时,参照文献 [26] 设置了相应的算法参数,并结合超密集边缘计算的实际场景进行了一定的改进,如表 2 所示。

表 2 算法参数设置

Table 2 Algorithm parameter setting

参数	参数含义	取值
$P_{c1}$	最大交叉概率	0.99
$P_{c2}$	最小交叉概率	0.88
$P_{m1}$	最大变异概率	0.1
$P_{m2}$	最小变异概率	0.01
$P_s$	选择概率	0.8
$T_{\text{max}}$	最大迭代次数	1 000
$T_0$	初始温度	100
$T_f$	最低温度	$1 \times 10^{-8}$
$W_{\text{max}}$	最大惯性权重	1
$W_{\text{min}}$	最小惯性权重	0.1

4.3 实验分析

在实验分析部分,主要进行以下工作,以评估超密集边缘计算网络相比与传统网络的对卸载结果的影响:1)对比 3 种算法下的能耗和适应度值;2)对比不同信道参数下的算法能耗,同时对比小区不同子信道分配情况下的任务卸载能耗;3)对比不同小区密集程度下的卸载能耗。

#### 4.3.1 移动设备数对算法性能的影响

在超密集边缘计算网络中,主要目标是在截止期限内最小化任务卸载的总能耗,因此,可将适应度值和能耗作为对算法的度量标准。将移动设备数设为30、60、90、120、150、180,每个小区的子信道数设置为2,任务的数据量大小设置为[0 Mb/s, 1.6 Mb/s],并将迭代次数设置为1 000次。将本文提出的AGASA算法与GA算法、GAPSO算法进行对比。图5和图6分别给出了3种算法下能耗和适应度值随移动设备数的变化情况。

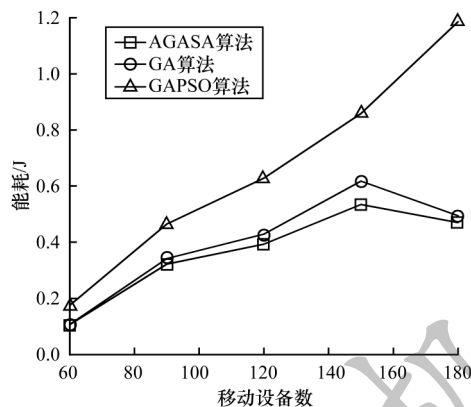


图5 不同算法下的能耗对比

Fig.5 Comparison of energy consumption under different algorithms

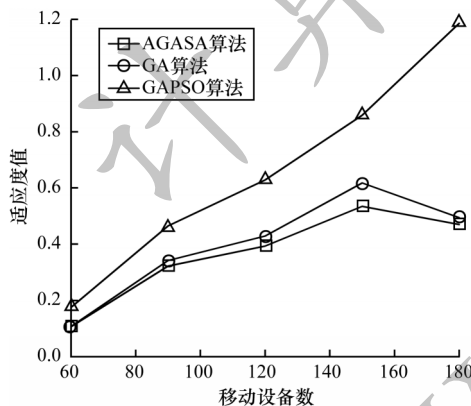


图6 不同算法下的适应度对比

Fig.6 Comparison of fitness value under different algorithms

由图5可以看出:当移动设备产生的任务数据量一致时,随着移动设备不断增加,所需要的计算资源不断增加,因此任务卸载的能耗不断增加;当移动设备数相同时,AGASA算法的任务卸载能耗最低,这是因为该算法的搜索能力更强,能够更快找到最优的卸载方案,因此性能最好;由于本文的任务请求为离散型,随着负载压力增加,GAPSO算法由于更适合连续型的任务请求,因此不容易找到卸载最优解。

由图6可以看出:当相同移动设备数时,AGASA算法下的适应度值最低。

#### 4.3.2 子信道数对卸载能耗的影响

为探究不同信道情况对卸载能耗的影响,对比任务数为90、120、150、180时,每个小区子信道数分别为1、2、3情况下的能耗。如图7所示,任务数的增加意味着移动设备数增加,一方面,所需要的频谱资源和计算资源不断增加,另一方面,同时通过同一个信道传输竞争有限频谱资源的移动设备数增加,信道压力增大带来更多的传输干扰,因此卸载能耗不断增加。当小区的子信道数为3时,卸载能耗最低,当任务数为180时,若仅通过一个信道传输,此时卸载能耗为0.535 5;当子信道数为2时,卸载能耗为0.47;当子信道数为3时,此时卸载能耗为0.459 4,随着子信道的数量越多,信道压力越小,任务传输时的干扰越小,因此卸载能耗越低。同时,当每个小区只有一个信道时,任务卸载能耗最高,这是因为当多个任务在一个信道上传输时,信道过于拥挤,传输干扰很大,所以传输过程中会消耗更多的能量。同时由图7可以观察到,AGASA算法下的能耗最低,说明该算法能够在超密集边缘计算网络中任务请求数较多的情况下,根据信道状态更新任务卸载方案,找到最优的分配策略。

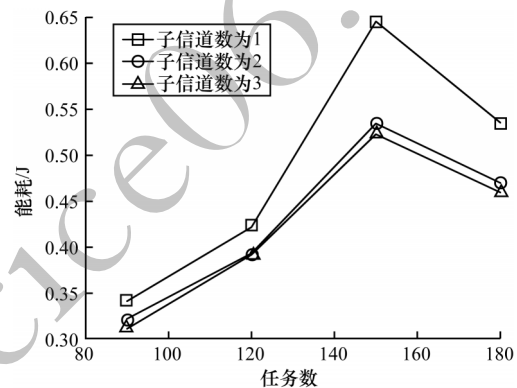


图7 不同子信道数下的能耗对比

Fig.7 Comparison of energy consumption under different number of sub-channels

#### 4.3.3 迭代次数对算法性能的影响

为验证AGASA算法的收敛性,比较AGASA、GAPSO以及GA算法在不同迭代次数下的能耗。令此时移动设备的数量为90,种群的数量为100个,迭代次数为1 000,实验结果如图8所示,可以看出:随着迭代次数的不断增加,GAPSO算法收敛速度最慢,且容易过早收敛,这是因为该算法的局部搜索能力差;而AGASA算法的收敛性最好,在迭代次数的影响下,当迭代次数为1 000时,该算法任务卸载的总能耗低于其他对比算法,这是因为该算法结合了SA和GA的优点,全局搜索能力和局部搜索能力都很强,在面对大量任务请求的同时,能够根据当前信道状态变化和计算资源,不断更新卸载决策,更快找到最优解。

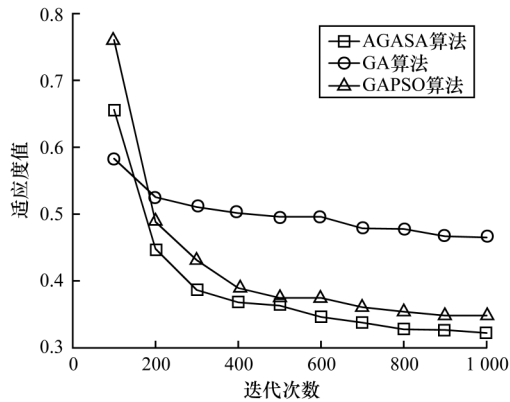


图8 不同算法下的收敛性对比

Fig.8 Comparison of convergence under different algorithms

4.3.4 信道状态对算法性能的影响

为对比不同信道状态下各个算法的性能,将移动设备数 $M$ 设置为120,服务器数 $N$ 设置为30,每个小区的子信道数 $C$ 设置2。如表3所示,从GAPSO的结果中可以看出:当 $P_{BG}$ 和 $P_{GB}$ 较小时,信道处于较稳定的状态,此时任务卸载的能耗较低;随着 $P_{BG}$ 和 $P_{GB}$ 的不断增大,信道处于不稳定的状态,将卸载决策更改为本地处理的计算任务增多,因此任务卸载能耗增加。另一方面,相比于GAPSO和GA算法,AGASA受信道状态变化的影响最小,且能够根据实际的环境随时不断调整参数,对环境的自适应性和搜索能力更强,因此任务卸载能耗最低。

表3 不同算法下信道参数对能耗的影响

Table 3 The influence of channel parameters on energy consumption under different algorithms 单位:J

参数设置	AGASA 算法	GA 算法	GAPSO 算法
$P_{GB}=0.01, P_{BG}=0.01$	0.399 2	0.418 9	0.579 0
$P_{GB}=0.05, P_{BG}=0.01$	0.393 3	0.427 0	0.562 8
$P_{GB}=0.10, P_{BG}=0.10$	0.402 2	0.448 8	0.597 4
$P_{GB}=0.50, P_{BG}=0.50$	0.411 6	0.444 2	0.639 4

4.3.5 SBS 部署密度对卸载能耗的影响

为分析SBS部署密度对卸载结果的影响,对比不同SBS数量下的卸载能耗。将任务数分别设置为90、120、150、180,对比SBS数量为1、10、20、30、40、60情况下的卸载能耗。实验结果如图9所示,可以看出:随着SBS数量从1到30不断增加,小区部署密度增加,卸载能耗变低,这是因为一方面部署在SBS侧的服务器数量越来越多,可供给移动设备的计算资源更加丰富;另一方面部署点越来越密集,小区可覆盖的移动设备数量增多,移动设备与小区的距离更近,且可供给移动设备的频谱资源更加丰富。因此,传输能耗和计算能耗更低。尤其是随着任务数增多,当SBS的数量为1时,所产生的卸载能耗最高,而当SBS为30时,卸载能耗明显降低。同时由图9可以看出:相比于传统网络架构下的边缘计算,UDN对任务卸载能耗的降低具有明显的优越性,然而随着SBS数量从40增加到60,卸载能耗反而增加,这是由于一方面SBS过于密集的情况下,交叉覆

盖区域的移动设备频繁切换基站会产生更大的网络压力,另一方面SBS之间距离更近的情况下,传输干扰也会随之增加,因此任务卸载能耗增大。

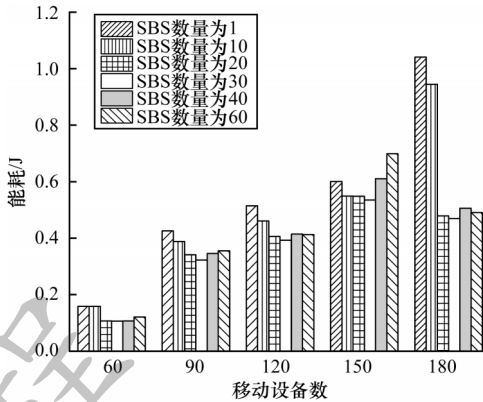


图9 不同SBS部署密度下的能耗对比

Fig.9 Comparison of energy consumption under different SBS deployment densities

4.3.6 超密集边缘计算网络下不同算法的性能

为验证本文所提算法在超密集边缘计算网络中的优越性,对比不同小区部署密度下,AGASA算法与GA和GAPSO算法的卸载能耗。在移动设备数为120的情况下,将SBS的数量设置为1、5、10、20、30、40,其中,SBS数量为1表示所有的移动设备将任务卸载到同一个SBS对应的MEC上。实验结果如图10所示,可以看出:随着SBS数量的不断增加,宏基站覆盖范围内的小基站部署更加密集,可提供的计算资源越来越丰富,卸载能耗越来越低;而相比与GA、GAPSO算法,AGASA算法下的能耗最低,这是因为在超密集边缘计算网络中,该算法结合了GA和SA算法的优势,具有更强的搜索能力,在密集部署的环境中能够更好地找到最优的卸载策略。

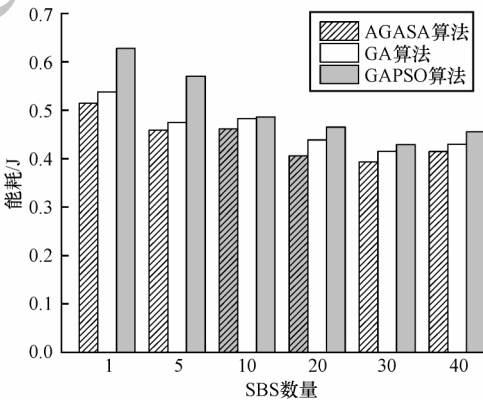


图10 超密集边缘计算网络中不同算法的能耗对比

Fig.10 Comparison of energy consumption of different algorithms in ultra-dense edge computing network

5 结束语

针对超密集边缘计算网络中任务卸载的能耗优化问题,本文考虑信道状态的变化、传输干扰等因素,结合AGA和SA算法提出AGASA算法,并基于此进行任务卸载,在满足截止期约束下对任务卸载

能耗进行优化,同时通过功率控制的黄金分割算法得到更优的上传功率。实验结果表明,本文方法可获得任务卸载能耗最低的信道分配决策和任务卸载决策,并且小区部署越密集,可提供的频谱资源和计算资源越丰富,任务卸载能耗越低。相比于传统网络架构,超密集边缘计算网络在传输和计算方面都具有明显的优势。后续将在超密集边缘计算网络中引入非正交多址传输的方式,提高传输效率,降低传输干扰,并且将计算开销作为优化目标,考虑时延敏感型和能耗敏感性任务的区别,对任务卸载、信道分配和功率控制方案做进一步优化。

### 参考文献

- [1] ETSI. Mobile-edge computing-introductory—technical white paper[EB/OL]. [2021-09-20]. <https://max.book118.com/html/2018/1006/8013022103001125.shtm>.
- [2] Cisco. Cisco visual networking index: global mobile data traffic forecast update, 2017-2022[EB/OL]. [2021-09-20]. <https://max.book118.com/html/2015/1002/26523192.shtm>.
- [3] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading [J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1628-1656.
- [4] 杨天, 杨军. 移动边缘计算中的卸载决策与资源分配策略[J]. 计算机工程, 2021, 47(2): 19-25.  
YANG T, YANG J. Offloading decision and resource allocation strategy in mobile edge computing [J]. Computer Engineering, 2021, 47(2): 19-25. (in Chinese)
- [5] YU S, CHEN X, ZHOU Z, et al. When deep reinforcement learning meets federated learning: intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network [J]. IEEE Internet of Things Journal, 2021, 8(4): 2238-2251.
- [6] DAI Y Y, XU D, MAHARJAN S, et al. Joint computation offloading and user association in multi-task mobile edge computing [J]. IEEE Transactions on Vehicular Technology, 2018, 67(12): 12313-12325.
- [7] HU S H, LI G H. Dynamic request scheduling optimization in mobile edge computing for IoT applications [J]. IEEE Internet of Things Journal, 2020, 7(2): 1426-1437.
- [8] CAO X, WANG F, XU J, et al. Joint computation and communication cooperation for energy-efficient mobile edge computing [J]. IEEE Internet of Things Journal, 2018, 6(3): 4188-4200.
- [9] LIU J, MAO Y Y, ZHANG J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems [C]//Proceedings of IEEE International Symposium on Information Theory. Washington D. C., USA: IEEE Press, 2016: 1451-1455.
- [10] TI N T, LE L B. Computation offloading leveraging computing resources from edge cloud and mobile peers [C]//Proceedings of IEEE International Conference on Communications. Washington D. C., USA: IEEE Press, 2017: 1-6.
- [11] ZHAO P T, TIAN H, QIN C, et al. Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing [J]. IEEE Access, 2017, 5: 11255-11268.
- [12] GUO F X, ZHANG H L, JI H, et al. An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing [J]. IEEE/ACM Transactions on Networking, 2018, 26(6): 2651-2664.
- [13] HUANG P Q, WANG Y, WANG K Z, et al. A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing [J]. IEEE Transactions on Cybernetics, 2020, 50(10): 4228-4241.
- [14] GUO J, ZHANG H L, YANG L C, et al. Decentralized computation offloading in mobile edge computing empowered small-cell networks [C]//Proceedings of IEEE GLOBECOM Workshops. Washington D. C., USA: IEEE Press, 2017: 1-6.
- [15] RANADHEERA S, MAGHSUDI S, HOSSAIN E. Computation offloading and activation of mobile edge computing servers: a minority game [J]. IEEE Wireless Communications Letters, 2018, 7(5): 688-691.
- [16] MISRA S, SAHA N. Detour: dynamic task offloading in software-defined fog for IoT applications [J]. IEEE Journal on Selected Areas in Communications, 2019, 37(5): 1159-1166.
- [17] PANG S, WANG S. Joint wireless source management and task offloading in ultra-dense network [J]. IEEE Access, 2020, 8: 52917-52926.
- [18] 郑利阳, 刘茜萍. 移动云计算环境下基于任务依赖的计算迁移研究 [J]. 计算机应用与软件, 2019, 36(7): 1-7, 82.  
ZHENG L Y, LIU X P. Computing migration based on task dependencies in mobile cloud computing environment [J]. Computer Applications and Software, 2019, 36(7): 1-7, 82. (in Chinese)
- [19] 王妍, 葛海波, 冯安琪. 云辅助移动边缘计算中的计算卸载策略 [J]. 计算机工程, 2020, 46(8): 27-34.  
WANG Y, GE H B, FENG A Q. Computation offloading strategy in cloud-assisted mobile edge computing [J]. Computer Engineering, 2020, 46(8): 27-34. (in Chinese)
- [20] LEI D M. Self-adjusting genetic algorithm [J]. Systems Engineering and Electronics, 1999, 21(11): 70-71. (in Chinese)  
雷德明. 自调整遗传算法 [J]. 系统工程与电子技术, 1999, 21(11): 70-71.
- [21] 程国建, 刘丽景, 石彩云, 等. 一种混合遗传算法在云计算负载均衡中的应用研究 [J]. 西安石油大学学报(自然科学版), 2012, 27(2): 93-97, 122.  
CHENG G J, LIU L J, SHI C Y, et al. Application of a hybrid genetic algorithm in the load balancing of cloud computing [J]. Journal of Xi'an Shiyou University (Natural Science Edition), 2012, 27(2): 93-97, 122. (in Chinese)
- [22] 罗斌, 于波. 移动边缘计算中基于粒子群优化的计算卸载策略 [J]. 计算机应用, 2020, 40(8): 2293-2298.  
LUO B, YU B. Computation offloading strategy based on particle swarm optimization in mobile edge computing [J]. Journal of Computer Applications, 2020, 40(8): 2293-2298. (in Chinese)
- [23] 张欣. 截止时间及预算约束的云 workflow 调度策略 [D]. 重庆: 重庆大学, 2017.  
ZHANG X. Scheduling of cloud workflow on budget and deadline constraints [D]. Chongqing: Chongqing University, 2017. (in Chinese)
- [24] 高基旭, 王珺. 一种基于遗传算法的多边缘协同计算卸载方案 [J]. 计算机科学, 2021, 48(1): 72-80.  
GAO J X, WANG J. Multi-edge collaborative computing unloading scheme based on genetic algorithm [J]. Computer Science, 2021, 48(1): 72-80. (in Chinese)
- [25] WU X, YING W, ZHANG T. An improved GAPSO hybrid programming algorithm [C]//Proceedings of 2009 International Conference on Information Engineering and Computer Science. Washington D. C., USA: IEEE Press, 2009: 1-4.
- [26] 刘通, 唐伦, 何小强, 等. 融合区块链与雾计算系统中基于网络时延和资源管理的优化任务卸载方案 [J]. 电子与信息学报, 2020, 42(9): 2180-2185.  
LIU T, TANG L, HE X Q, et al. Optimal task offloading scheme based on network delay and resource management in joint blockchain and fog computing system [J]. Journal of Electronics & Information Technology, 2020, 42(9): 2180-2185. (in Chinese)