

基于 D2D 协同的边缘计算迁移机制研究

薛 锋, 张雅文, 陈思光
(南京邮电大学 物联网学院, 江苏 南京 210003)

摘 要:为了缓解边缘网络通信压力、降低物联网设备对服务节点的计算与通信负荷,提出了一种基于 Device-to-Device (D2D) 协同的边缘计算迁移机制。具体而言,通过综合考虑 D2D 设备、边缘节点的迁移决策以及传输功率分配,规划了一个任务完成总能耗最小化的优化问题,进一步,定义了 D2D 设备的积极性度量约束以促进 D2D 设备与普通用户间的协作。同时,提出了基于动态感知蝙蝠群体的高效计算迁移算法(Dynamic Sensing Bat Population-Based Efficient Computation Offloading Algorithm, DSBP-ECO)。该算法融合经典蝙蝠算法思想,引入一种自适应的动态惯性权重,以通过实时感知环境变化调整蝙蝠群体的移动方向和速度,并采用混沌映射理论对种群进行初始化。最后,仿真结果表明,所提方案能够以较快的速度收敛,并获得最优迁移和功率分配策略,与其他几种基准方案相比,该方案在降低系统能耗方面具有较大的优势。

关键词:边缘计算;计算迁移;资源分配;D2D 通信;用户协同;动态权重

中图分类号:TP393

文献标识码:A

文章编号:1673-629X(2023)06-0117-08

doi:10.3969/j.issn.1673-629X.2023.06.018

Research on Edge Computing Offloading Mechanism Based on D2D Collaboration

XUE Feng, ZHANG Ya-wen, CHEN Si-guang

(School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: In order to relieve the communication pressure of edge network and reduce the computing and communication load of Internet of Things (IoT) devices to service nodes, we propose an edge computing offloading mechanism based on Device-to-Device (D2D) collaboration. Specifically, based on the comprehensive consideration of offloading decisions and transmission power allocation, an optimization problem that minimizes the total energy consumption of task completion is formulated. Furthermore, the motivation measurement constraint of D2D device is defined to promote collaboration between cooperation users and common users. At the same time, an efficient computation offloading algorithm based on dynamic sensing of bat population (DSBP-ECO) is proposed. The algorithm combines the ideas of classical bat algorithm, and introduces an adaptive dynamic inertia weight. Through sensing environment in real time to adjust the movement direction and speed of bat population, and the chaotic mapping theory is used to initialize the population. Finally, the simulation results show that the proposed scheme can converge at a faster speed, and achieve the optimal offloading and power allocation strategy. Compared with other benchmark schemes, the proposed scheme has significant advantages in reducing system energy consumption.

Key words: edge computing; computation offloading; resource allocation; D2D communication; user collaboration; dynamic weight

0 引言

近年来,随着物联网(Internet of Things, IoT)与5G技术的快速发展,衍生出许多智能应用服务,例如增强现实、人机交互、智慧城市等^[1],这些新兴领域给物联网的发展带来了巨大突破。然而科技发展所面临

的问题也随之而来,面对激增的数据规模,原有的服务结构已难以满足用户的基本需求^[2],为了摆脱这一困境,云计算进入大众的视野。得益于云服务器丰富的计算资源,计算密集型任务能够以较低的成本被处理,同时云计算具有可扩展性及高性价比等优势^[3]。但是

收稿日期:2022-08-23

修回日期:2022-12-27

基金项目:国家自然科学基金(61971235);中国博士后科学基金(面上资助)(2018M630590);江苏省博士后科研资助计划(2021K501C);江苏省“333 高层次人才培养工程”(XX);南京邮电大学“1311”人才计划(XX)

作者简介:薛 锋(1996-),男,硕士研究生,研究方向为边缘计算;通讯作者:陈思光(1984-),男,CCF 会员(D6662M),博士,教授,研究方向为雾/边缘计算、智能物联网。

由于云服务器与终端设备之间的长距离通信及在通信过程中可能存在的网络拥塞问题,不仅会增加不必要的通信开销,还易引起高延迟响应服务,难以满足时延敏感型应用的基本需求,从而降低用户体验^[4-6]。

为了消除集中式计算带来的弊端,减轻网络传输压力,边缘计算被认为是一种有效的解决方案^[7]。边缘计算为用户提供多元化服务,如计算、存储和应用,在靠近数据源处设立处理节点,减少网络响应时间,满足用户在实时业务方面的基本需求,提高用户服务质量^[8]。

在边缘计算架构下,终端设备可将其计算任务迁移至节点(边缘服务器,其它终端设备等)处理以获得更高的服务质量。多种边缘计算迁移方案也围绕设备-边缘节点展开研究。例如文献[9-15],围绕设备-节点间任务迁移,在综合考量系统资源、迁移策略下,降低设备时延和能耗,减少系统成本。具体地,文献[9]在多用户移动边缘计算(Mobile Edge Computing, MEC)系统中构建了具有异构云的迁移框架,考虑到任务延迟约束,提出了基于动态规划的优化算法,衡量移动设备获取的带宽和计算资源,最大程度地减少用户的能耗,同时,提出了一种近似迁移算法,将能量稀疏处理提高算法效率,仿真结果表明,该算法能以较低的计算复杂度减少移动设备的总能耗。类似地,Xu 等人^[10]考虑了具有时延约束的多用户 MEC 系统,为了减小传输数据的大小,在计算迁移前对数据进行压缩,进而提出了一种凸优化算法,通过联合优化迁移决策、数据压缩和资源分配达到最小化总能耗的目的。文献[11]考虑了一个垂直和异构的多接入边缘计算系统,在通信链路容量的约束下,提出了一个联合迁移决策和无线资源分配的优化问题,由于该问题的非凸性,考虑将原始问题划分为多个子问题分别求解以寻找最优解,同时通过仿真验证了该方案能够最大程度地降低计算任务处理的总能耗。从降低系统时延的角度入手,文献[12]提出了一种两阶段计算迁移方案,首先终端用户根据服务器工作负载确定其迁移到服务器的数据量,在此基础上,提出了一种分布式算法来安排迁移任务的处理顺序。通过仿真验证了所提方案能够获得最优的迁移策略,同时提高了服务器处理效率并实现了任务处理时延的最小化。文献[13]聚焦于车载边缘计算系统,设计了一种虚拟边缘形成算法,该算法综合考虑了虚拟边缘的稳定性和系统中可用计算资源的合理分配,减少了因链路意外断开导致任务迁移失败的数量,同时降低了计算任务的执行时间。

此外,文献[14-15]在降低系统成本方面取得了显著成效。为了给用户提供高性能、低延迟的服务体验,Feng 等人^[14]规划了一个基于任务迁移策略、缓存

容量和 MEC 服务器计算能力的优化问题,为了解决该混合整数非线性规划问题,提出了一种分布式协作数据缓存和计算迁移迭代算法,可显著降低系统总成本。在文献[15]中提出了一种迭代算法,在多用户多服务器的物联网网络中,通过感知资源变化,综合考虑各项指标,如迁移策略、CPU 计算能力和终端设备的发射功率,以达到最小化系统总成本的目的。上述研究方案虽在一定程度上降低了系统能耗和时延,取得最优解。但传统数学求解方法使得该类方案更加适用于静态的场景,对于非确定性信息及动态网络环境的处理能力较差。同时,面对终端设备数量的急剧增加,设备-节点任务迁移处理架构显得力不从心。

在边缘计算网络中,部署了众多终端设备,由于其任务处理效率的差异,难免会有终端设备处于空闲状态。为了更加有效地利用这类计算资源,同时,进一步提高频谱利用率和网络吞吐量,Device-to-Device (D2D)通信技术已被广泛应用到边缘计算中^[16]。例如,在文献[17]中,Cheng 等人研究了认知无线网络(Cognitive Radio Network, CRN)中的计算迁移问题,提出了一种基于非正交多址接入(Non Orthogonal Multiple Access, NOMA)的 D2D 辅助计算迁移方案,在任务可容忍延迟和发射功率的约束下,通过坐标下降法和逐次凸逼近得到最优的主次用户迁移决策和功率控制,实现系统能耗的最小化。此外,文献[18]提出了一种混合整数随机逐次凸逼近算法,具体地,利用智能反射平面(Intelligent Reflective Surface, IRS)技术协助用户将任务迁移,同时,为了减少交换信道状态信息(Channel Status Information, CSI)产生的开销,联合设计了混合时间尺度上的波束成形和资源分配,大量仿真表明了该算法在降低任务完成时延方面的有效性。文献[19-21]研究了 D2D 辅助网络在时延和能耗的联合优化方面取得的成果。其中文献[19]提出了一种在线资源协调与分配方案,旨在解决 D2D 辅助边缘计算系统中任务响应时间和能耗的最小化问题,通过考虑多用户协作的部分迁移策略、传输调度和计算资源分配,最大程度地降低网络响应成本,从而提高用户服务质量。Fang 等人^[20]针对设备增强型 MEC 中的计算密集型任务,构建了一个最大化所有任务总收益的优化问题,该问题被建模为多用户计算迁移博弈,为了求解最优值,提出了一种基于回复的分布式多用户计算迁移算法,该算法从通信资源、D2D 选择和迁移模式的联合优化角度,降低了时延和能耗,达到了最大化系统总收益的目的。类似地,文献[21]考虑在蜂窝网络中规划了一个任务执行时间和能耗加权和优化问题,并引入一个联合任务管理架构以实现高效的信息交换,并提出一种启发式算法。首先,通过库

恩—曼克莱斯算法获得最优计算迁移决策;紧接着,利用拉格朗日对偶法求解资源分配策略,大量的仿真表明,该算法能有效降低任务执行的总成本。上述 D2D 辅助边缘计算的优化方案虽然在时延或能耗方面取得了成效,然而大多方案都建立在理想的环境下,即 D2D 设备在为其他用户提供计算资源支持时缺乏主动性,同时缺乏对迁移策略和传输功率的综合考量。

基于以上分析,针对边缘计算中任务迁移问题存在以下两个方面的挑战:

(1) 通常情况下,在求解优化问题时,大多应用传统的数学规划方法,然而面对复杂的非线性规划问题,传统数学方法存在一定的局限性,更甚之,传统方法对于非确定性信息的处理能力较差,难以应对实际问题中动态变化的网络环境。

(2) 上述基于 D2D 辅助的优化方案在处理任务迁移问题时,成效显著且可充分调动系统中的可用资源,进一步提高网络吞吐量,且能够有效降低系统时延或能耗,但是上述方案并未关注 D2D 设备共享资源的积极性,同时缺乏对迁移策略和传输功率的综合考虑。

针对上述问题,该文提出了一个基于 D2D 协同的边缘计算迁移机制,主要贡献总结如下:

- 研究了一个基于 D2D 协同的边缘计算迁移模型,基于任务迁移策略和功率分配的综合考虑,提出了一个最小化任务处理总能耗的优化问题。同时,考虑到 D2D 设备的积极性度量约束,并通过联合优化迁移策略和功率分配,实现最小化系统总能耗的目标。

- 针对上述的混合整数非线性规划问题,提出了一种基于动态感知蝙蝠群体的高效计算迁移算法,该算法融合经典蝙蝠算法思想,引入自适应方式调节迭代参数,使种群能够灵活地感知环境变化并调整移动方向,同时,为了提高求解精确度,采用混沌映射完成种群初始化,并结合动态惯性权重更新群体的飞行速度,更好地探索最优解,以提高算法的求解精确度。

- 通过大量仿真与分析,与其它两种同类型的基准方案相比,该算法能够以较快的速度收敛,并且该算法在降低能耗方面具有较大的性能优势,能够得到远优于其他方案的最优策略,实现了总能耗的最小化。

1 系统模型

1.1 网络模型

本节构建了一个基于 D2D 协同的边缘计算迁移系统模型,旨在优化执行和迁移计算任务产生的能量消耗。该系统由单个具有边缘服务器的基站和多个用户设备组成,其中用户设备包括普通用户 (General User) 和协作用户 (Cooperation User),具体的系统结构如图 1 所示。

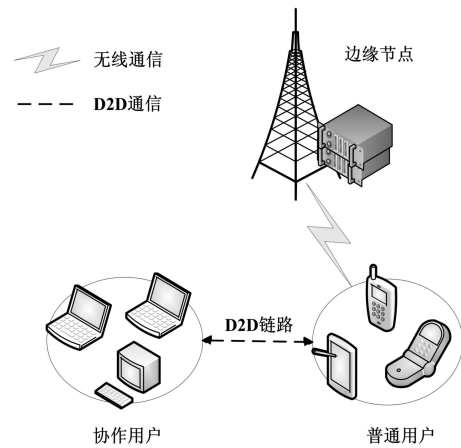


图 1 D2D 协同边缘计算模型

假设用户层有 N 个普通用户 ($i \in \{1, 2, \dots, N\}$), 每个用户 i 会生成计算任务 c_i , 考虑到用户本身的计算能力有限, 用户 i 会选择将任务在本地计算或迁移处理。同时, 为了缓解单个用户处理计算密集型任务的压力, 部署了 M 个协作用户 ($j \in \{1, 2, \dots, M\}$), 该用户本身不生成计算任务, 并且能够与普通用户建立 D2D 通信, 为其他用户提供计算服务, 减少资源浪费, 进一步提高用户服务质量。此外, 为了保证任务不被重复处理, 一个用户 i 最多只能请求一次任务传输, 一个用户 j 也只能为一个用户 i 提供协作服务。定义集合 $\Omega = \{1, 2, \dots, M, S\}$ 为迁移的目标, 用 x_{ij}^1 和 x_{is}^2 ($x_{ij}^1 + x_{is}^2 \leq 1$) 表示用户 i 的迁移决策, 对于给定的任务, 用户 i 有三种计算模式: (1) 本地计算: 当 $x_{ij}^1 = 0, x_{is}^2 = 0$ 时, 表示任务全部在本地设备计算完成; (2) D2D 迁移: 当 $x_{ij}^1 = 1, x_{is}^2 = 0$ 时, 表示用户 i 选择将任务通过 D2D 链路迁移至协作用户 j 处理; (3) 边缘迁移: 当 $x_{ij}^1 = 0, x_{is}^2 = 1$ 时, 表示用户 i 通过蜂窝网络将任务上传到计算能力更强的边缘节点。

边缘层部署了一个边缘节点, 该节点负责管理所有的链路通信, 并协助控制其覆盖范围内的用户间协作。同时, 由于边缘服务器强大的计算能力, 具备服务器的边缘节点支持用户设备的任务处理, 并将结果回传。

1.2 计算模型

本节介绍的计算模型主要包括三个部分: 本地计算、D2D 计算和边缘计算, 下面将详细说明这三个部分。

(1) 本地计算。

假设所有任务都能够执行成功。定义用户设备 i 的计算能力为 f_i^{local} (MCycles/s), 各个用户具有不同的计算能力。因此任务在本地计算所需时延可表示为:

$$t_i^{\text{local}} = \frac{c_i R}{f_i^{\text{local}}} \quad (1)$$

其中, c_i (K bit) 表示用户 i 的任务大小, R 表示本地设备处理 1 bit 任务所需的 CPU 周期数。

同时, 本地计算的能耗与用户设备的 CPU 性能成正比, 即:

$$e_i^{\text{local}} = k_i (f_i^{\text{local}})^2 c_i R \quad (2)$$

其中, k_i 表示用户设备 i 的有效电容系数。

(2) D2D 计算。

由于本地设备有限的计算资源, 用户在处理计算密集型任务时往往显得力不从心, 从而选择将任务迁移至边缘节点。然而, 在某一时段内, 如果大量的用户均选择迁移任务, 容易造成系统频谱资源匮乏, 从而降低用户服务质量。针对上述问题, 为了缓解通信压力, 降低对服务节点的负荷, 引入了 D2D 通信技术, 在通信资源不足的情况下, 普通用户可以将任务通过 D2D 链路传输至协作用户处理, 从而提升任务处理效率。假设 D2D 链路的信道条件处于理想状态下, 即信道增益是恒定的, 可以计算得到普通用户 i 与协作用户 j 之间 D2D 通信的数据速率为:

$$r_{ij}^{\text{d2d}} = W_{ij} \log_2 \left(1 + \frac{p_i^{\text{local}} d_{ij}^{-\tau}}{\sigma^2} \right) \quad (3)$$

其中, d_{ij} 表示用户 i 和 j 之间的通信距离, τ 表示信道损耗系数, p_i^{local} 表示用户 i 的发射功率。

因此, 任务在 D2D 计算时所消耗的时间与能耗分别为:

$$t_i^{\text{d2d}} = \frac{c_i}{r_{ij}^{\text{d2d}}} \quad (4)$$

$$e_i^{\text{d2d}} = p_i^{\text{local}} t_i^{\text{d2d}} + \frac{c_i R}{f_j^{\text{d2d}}} \quad (5)$$

其中, f_j^{d2d} 表示协作用户 j 的计算能力。

考虑到 D2D 用户作为个人用户易出现自私行为, 导致任务处理效率降低。为了提升 D2D 用户计算服务的积极性, 为每个 D2D 设备定义积极性度量, 假设第 j 个 D2D 设备的积极性度量指标为:

$$\text{Ini}_j = \overline{\text{Ini}}_j + t_j^{-\kappa} \quad (6)$$

其中, $\overline{\text{Ini}}_j$ 表示 D2D 设备 j 的初始积极性数值, κ 为大于零的常数, $t_j (t_j = \frac{c_i R}{f_j^{\text{d2d}}})$ 表示设备处理任务所花费的时间。

此外, 如果用户 i 想将任务迁移至其他 D2D 设备 j' 进行处理, 积极性度量应当满足:

$$\text{Ini}_{j'} > \text{Ini}_j \quad (7)$$

(3) 边缘计算。

考虑到本地设备计算资源稀缺的局限性, 用户往往选择将部分任务迁移处理, 得益于边缘节点丰富的计算资源, 在处理计算密集型任务时能够游刃有余。

用户层通过蜂窝网络与边缘层通信, 根据香农定理, 任务的上行传输速率为:

$$r_{is}^{\text{tran}} = W_{is} \log_2 \left(1 + \frac{p_i^{\text{tran}} d_{is}^{-\tau}}{\sigma^2} \right) \quad (8)$$

其中, W_{is} 表示用户 i 与边缘节点之间的通信带宽。

因此, 任务上传至边缘节点需要的时延和能耗分别为:

$$t_{is}^{\text{tran}} = \frac{c_i}{r_{is}^{\text{tran}}} \quad (9)$$

$$e_{is}^{\text{tran}} = p_i^{\text{tran}} t_{is}^{\text{tran}} \quad (10)$$

任务到达后, 边缘节点会为其分配计算资源, 待任务处理完毕后, 边缘节点将结果回传给原用户, 考虑到计算结果数据量极小, 返回过程不会给系统造成较大负担, 因此, 该文将忽略结果回传所产生的成本, 即用户 i 将任务迁移至边缘节点处理所需的时延和能耗分别可表示为:

$$t_{is}^{\text{com}} = \frac{c_i R}{f_s^{\text{edge}}} \quad (11)$$

$$e_{is}^{\text{com}} = p_s^{\text{edge}} t_{is}^{\text{com}} \quad (12)$$

因此, 任务迁移至边缘节点处理的总能耗如下:

$$e_{is}^{\text{edge}} = e_{is}^{\text{tran}} + e_{is}^{\text{com}} \quad (13)$$

2 优化问题描述

综上, 该文规划了一个任务完成总能耗的优化问题, 通过联合优化迁移决策 x_{ij}^1 、 x_{is}^2 以及本地设备的传输功率 p_i^{tran} , 实现任务完成总能耗的最小化。根据第 2 节定义的系统模型, 可将单个任务完成的能耗表示为:

$$e_i^{\text{total}} = (1 - x_{ij}^1) (1 - x_{ij}^2) e_i^{\text{local}} + x_{ij}^1 (1 - x_{is}^2) e_i^{\text{d2d}} + x_{is}^2 (1 - x_{ij}^1) e_i^{\text{edge}} \quad (14)$$

因此, 所有任务完成的总能耗可写作:

$$E = \min_{x_{ij}^1, x_{is}^2, p_i^{\text{tran}}} \sum_{i=1}^N e_i^{\text{total}} \quad (15)$$

$$x_{ij}^1 + x_{is}^2 \leq 1 \quad (15a)$$

$$\sum_{i=1}^N x_{ij}^1 \leq 1 \quad (15b)$$

$$\sum_{j=1}^M x_{ij}^1 \leq 1 \quad (15c)$$

$$x_{ij}^1, x_{is}^2 \in \{0, 1\} \quad (15d)$$

$$0 \leq p_i^{\text{tran}} \leq p_i^{\text{max}} \quad (15e)$$

$$\text{Ini}_{j'} > \text{Ini}_j, j, j' \in \{1, 2, \dots, M\} \quad (15f)$$

约束 (15a) 表示每个用户 i 只能选择一个目标进行任务迁移; 约束 (15b) 和 (15c) 表示一个用户 i 只能与至多一个 D2D 设备 j 通信, 一个用户 j 也只能为至多一个用户 i 提供计算服务; 约束 (15d) 表示 D2D 和边缘节点的迁移决策均为 0 或 1; 约束 (15e) 表示边缘节点分配给本地设备 i 的传输功率不能超过其最大

值,约束(15f)表示 D2D 通信的迁移约束。

3 基于动态感知蝙蝠群体的高效计算迁移算法

由于文中优化问题属于非凸且为混合整数非线性规划问题,直接求解相对复杂。通常,传统的数学方法在解决该类问题时往往需要较高的计算成本,为了提高求解效率,提出了一种基于动态感知蝙蝠群体的高效计算迁移算法(Dynamic Sensing Bat Population-Based Efficient Computation Offloading Algorithm, DSBP-ECOA)。该算法融合经典蝙蝠算法思想,并通过感知环境变化,引入一种动态惯性权重因子自适应

地调节移动速率与方向。同时,考虑到探索过程中种群越来越接近最优个体导致收敛速度下降的问题,在对位置和速度初始化时,该算法采用了混沌序列,确保了种群特征的差异性,从而提升了该算法的优化性能。

蝙蝠算法是一种基于群体智能的启发式搜索算法,该算法利用自然界生物蝙蝠的社会行为、觅食以及避开障碍物时使用的回声定位行为生成其搜索策略。由于经典蝙蝠算法存在收敛速度慢、搜索能力有限和易陷入局部极小值等弊端,该文提出了改进的蝙蝠算法,即自适应感知的蝙蝠优化算法求解目标函数。具体流程如图2所示。以常规假设为前提,构成了蝙蝠算法的基本思想:

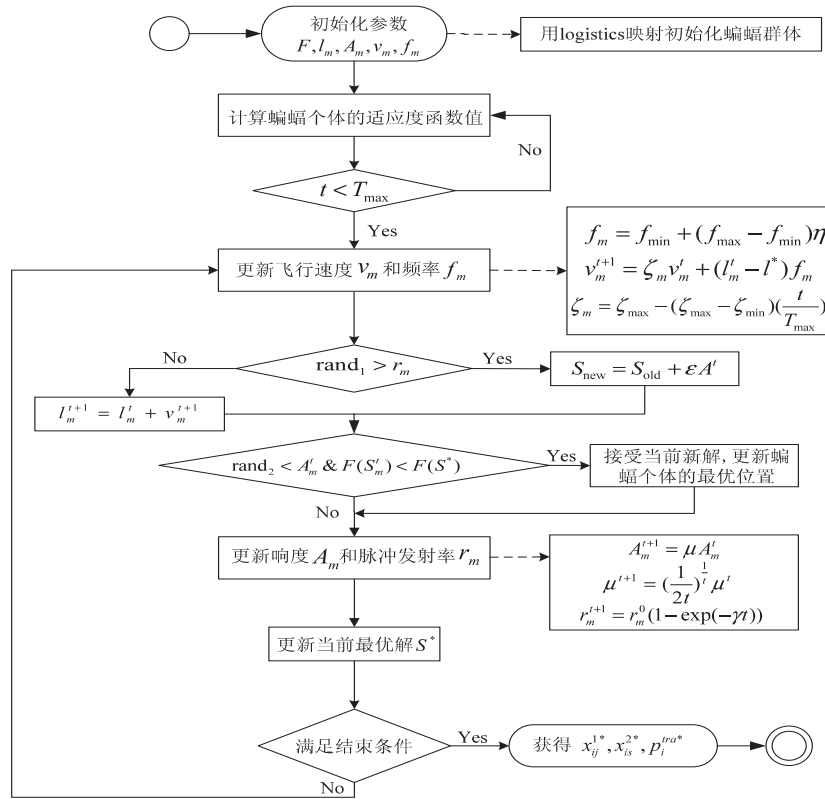


图2 DSBP-ECOA 算法流程

(1) 所有蝙蝠都使用回声定位机制来寻找目标物,并且能够甄别食物与障碍物之间的差别。

(2) 每个蝙蝠个体在位置 l_m 以速度 v_m 随机飞行,并根据声波反馈自动调整发射脉冲的频率 f_m ,同时能够调整脉冲发射率 $r(r \in [0,1])$ 使其接近目标位置,更新位置与飞行速度。

(3) 假设蝙蝠飞行时的响度 A_m 从常数 $A_0 (A_0 > 0)$ 变化到最小值 A_{\min} 。

首先对种群进行初始化,主要包括种群规模 Z 、初始位置 l_i 、初始速度 v_i 及响度 A_i 等。由于蝙蝠个体缺乏变异机制,在寻优过程中一旦受到局部约束很难摆脱,而且随着迭代次数的增加,蝙蝠个体会逐渐聚集,从而丧失种群多样性,导致算法收敛速度大大

降低。

BA 一般采用随机生成方式初始化群体,该方式易使算法陷入局部最优,因此为增加种群多样性,提高初始解的质量,该算法引入了混沌映射理论,利用混沌序列对种群的位置和速度初始化,使得种群能够均匀分布,从而提高了全局探索的多样性。该文采用 logistic 映射模型进行初始化,具体表示为:

$$z_{t+1} = \rho z_t (1 - z_t) \quad (16)$$

其中, ρ 表示控制参数。

定义集合 $S_m = (x_1^1, \dots, x_N^1; x_1^2, \dots, x_N^2; p_1^{\text{tran}}, \dots, p_N^{\text{tran}})$ 为该算法的解向量,将其代入适应度函数 F 评估当前的解的优劣,即:

$$F(S_m) = E(x_{ij}^1, x_{is}^2, p_i^{\text{tran}}) \quad (17)$$

假设 t 时刻的蝙蝠位置和速度分别为 l_m^t 和 v_m^t , 因此, $t + 1$ 时刻第 m 只蝙蝠的速度和位置可以写作:

$$f_m = f_{\min} + (f_{\max} - f_{\min})\eta \quad (18)$$

$$v_m^{t+1} = \zeta_m v_m^t + (l_m^t - l^*)f_m \quad (19)$$

$$l_m^{t+1} = l_m^t + v_m^{t+1} \quad (20)$$

式(18)表示第 m 只蝙蝠的频率更新过程, f_{\min} 和 f_{\max} 分别表示频率的下限和上限; η 是一个服从均匀分布的随机数, l^* 表示当前时刻的最优位置。

在其他群体智能算法中,例如粒子群算法,会考虑在速度更新时加入惯性权重,以适应当前环境。通常,在迭代初期,惯性权重较大会增强全局搜索能力,而在迭代后期,惯性权重较小能获得更精确的局部搜索,因此,为了提高所提算法的搜索能力及精确度,在式(19)中引入了一个动态惯性权重,即:

$$\zeta_m = \zeta_{\max} - (\zeta_{\max} - \zeta_{\min}) \left(\frac{t}{T_{\max}} \right)^2 \quad (21)$$

其中, T_{\max} 为迭代周期的最大轮次, ζ_{\min} 和 ζ_{\max} 分别为最小值与最大值。

蝙蝠算法是基于全局搜索与局部搜索相结合的搜索策略,因此,在局部搜索时,种群通过随机移动找寻最优解,即:

$$S_{\text{new}} = S_{\text{old}} + \varepsilon A^t \quad (22)$$

其中, ε 是 $[-1, 1]$ 范围内的随机数,用于调节随机移动的频率与方向, A^t 是当前时刻所有蝙蝠个体的平均响度。

通常情况下,蝙蝠在搜寻过程中的脉冲发射速率和响度都是不断变化的,而一旦蝙蝠个体找到目标物,它就会逐渐降低其脉冲发射的响度,并增加脉冲发射速率,其迭代过程表示如下:

$$A_m^{t+1} = \mu A_m^t \quad (23)$$

$$r_m^{t+1} = r_m^0 (1 - \exp(-\gamma t)) \quad (24)$$

其中, μ 表示脉冲响度衰减因子, r_m^0 为第 m 只蝙蝠的最大脉冲发射率, $\bar{\omega}$ ($\bar{\omega} > 0$) 表示脉冲频率更新系数。假设当 $t \rightarrow \infty$ 时, $A_m^t \rightarrow 0$, $r_m^t \rightarrow r_m^0$ 。

在传统的蝙蝠算法中, μ 通常是 $(0, 1)$ 区间范围内的固定常数。在迭代初期,蝙蝠种群能根据响度及频率的变化向更佳的位置移动,然而随着迭代次数的增加,参数不能根据种群当前进化状况自适应地调整数值。因此,为了增加种群与当前环境的适应性,提升搜索质量,该文考虑参数 μ 能够随着迭代次数的变化动态更新,可以表示为:

$$\mu^{t+1} = \left(\frac{1}{2t} \right)^{1/t} \mu^t \quad (25)$$

综上所述,为解决上一节构建的优化问题,所提算法体现了相应的针对性,为了便于理解,将上述求解的详细过程凝练如算法 1。

算法 1: 基于动态感知蝙蝠群体的高效计算迁移算法。

1. 输入: 任务大小. $c_i, i \in \{1, 2, \dots, N\}$
2. 输出: 最优值 x_{ij}^{1*} 、 x_{is}^{2*} 、 $p_i^{\text{tm}*}$ 和最小能耗 E^*
3. BEGIN
4. 参数初始化: 蝙蝠种群规模 Z , 适应度函数 F , 蝙蝠位置 l_m ($m = 1, 2, \dots, Z$), 响度 A_m , 初始速度 v_m 和声波频率 f_m ;
5. 根据 logistic 映射 (16) 初始化蝙蝠群体 z ;
6. 设置 $t = 1$;
7. 计算蝙蝠个体的适应度函数值 $F(S_m)$;
8. While $t < T_{\max}$ 且满足约束 (15a) - (15f); do
9. 通过公式 (18) - (21) 更新第 m 只蝙蝠的移动速度 v_m^t 和位置 l_m^t ;
10. If $\text{rand}_1 > r_m$;
11. 根据公式 (22) 生成局部新解 S_{new} , 并替换 S_m^t ;
12. End if
13. If $\text{rand}_2 < A_m^t$ and $F(S_m^t) < F(S^*)$
14. 接受当前新解, 并根据公式 (24) 更新 r_m^t , 根据公式 (23) 和 (25) 更新 A_m^t ;
15. End if
16. 更新当前最优解 S^* ;
17. End While
18. 获得最优迁移策略 (x_{ij}^{1*} , x_{is}^{2*} , $p_i^{\text{tm}*}$), 根据公式计算最小总能耗 E^*
19. END

4 仿真结果分析

本节通过一系列仿真实验评估了基于动态感知蝙蝠群体的高效计算迁移算法的有效性,并且与其他几种同类型算法对比,证明了所提算法的性能优势。

该文使用 Matlab 进行实验仿真,仿真实验均在一台配置为 Intel Core i5-8250U@1.60 GHz CPU, 8G RAM 的笔记本电脑上进行。D2D 链路和无线链路的信道带宽 W_{ij} 与 W_{is} 分别为 10 Mb/s 与 25 Mb/s。普通用户的计算能力 $f_i^{\text{local}} = 6$ Mcycles/s, 每个设备计算 1 Kb 任务所需的 CPU 周期数为 4 000 cycles, 用户 i 的发射功率 $p_i^{\text{tran}} = 8$ W; 设定协作用户 j 的计算能力 f_i^{ld} 为 $[7, 10] \times 10^6$ cycles/s 范围内任意常数; 将普通用户 i 与协作用户 j 之间的信道损耗系数设置为 0.01。边缘节点的计算能力 $f_s^{\text{edge}} = 12$ Mcycles/s, 计算 1 Kb 任务所需的 CPU 周期数为 200 cycles; 假设每个用户 i 的任务在区间 $[10, 30]$ Mb 范围内随机生成。

图 3 显示了所提方案初始化种群的分布情况。从图中种群个体的分布情况可以看出,初始化形成的种群个体能够较为均匀地分布在搜索范围内的各个位置,没有出现种群聚集状况,并且个体之间几乎没有特征重合的现象出现。这是因为提出的方案中采用了混沌映射理论,通过 logistic 映射模型对种群进行初始化,从而形成混沌序列,增加了种群多样性。由此可推

断,该方案能够有效改善种群的构成特征,提高算法初始解的质量。

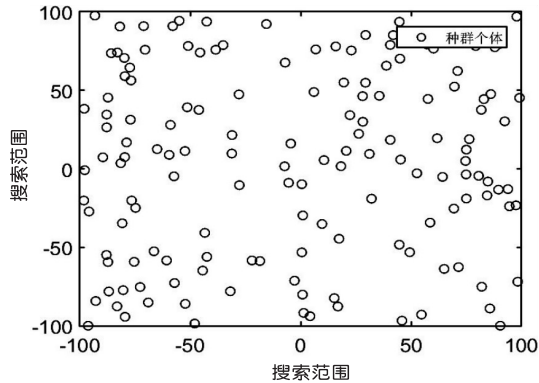


图3 初始化种群的分布情况

图4描绘的是在初始化种群时,设定不同数量的初始化种群对适应度函数的影响情况对比。由图中曲线可以看出,无论初始化种群的数量如何变化,在经过一定的迭代次数后,适应度函数均能达到收敛。其中当种群数量为30时,所提方案的性能达到最优,Fitness大概在迭代次数达到10次时收敛到稳定值,对比其他两种初始化种群数量,收敛速度更快。并且对比图中三条曲线发现,当Population number=30时,适应度函数的值最低,这是因为随着迭代次数的增加,种群数量越大,其探索能力越强,种群个体之间相互作用与影响,能够在较短的时间内寻得最优解。

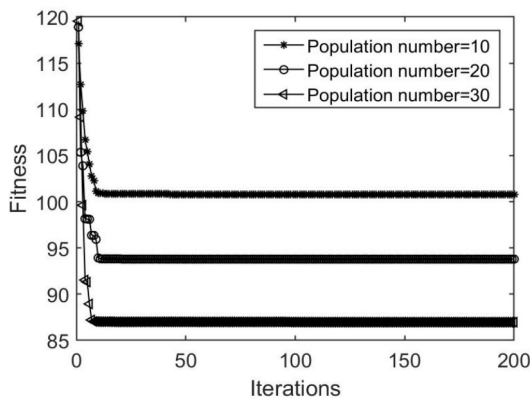


图4 不同种群数量对Fitness收敛性能的影响

图5展示了蝙蝠算法中脉冲频率更新系数 γ 的取值对Fitness收敛性能的影响。观察图中曲线的变化情况可以发现,随着迭代次数的增加,三条曲线均能收敛至稳定值。并且从图中可以看出, γ 的取值不同,其Fitness的收敛值也各不相同。具体地,当 $\gamma = 0.5$ 或 $\gamma = 0.9$ 时,Fitness都能够较快收敛, $\gamma = 0.7$ 时Fitness虽存在一定波动,但经过20次左右的迭代仍会达到稳定值。从图中可以看出,当 $\gamma = 0.9$ 时其余两项,Fitness值最小,这是因为当 γ 较小时,蝙蝠种群易陷入到局部极值范围内,在此范围内搜索得到最优值是局部而非全局最优解。综上所述,所提方案在实验仿真

中,选取 $\gamma = 0.9$ 作为实验数值。

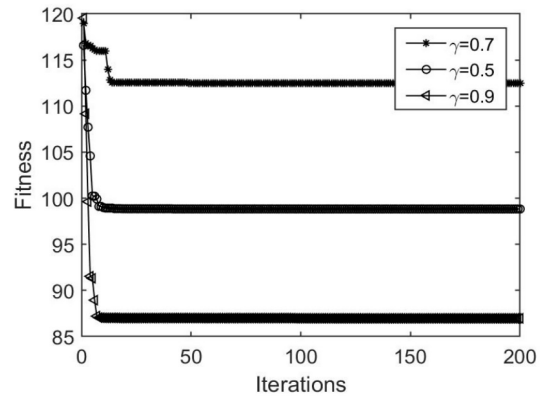


图5 不同gamma取值对Fitness收敛性能的影响

图6分析了三种不同方案的适应度函数收敛对比情况。其中“BA”表示基础蝙蝠算法,“DSBP-ECO”表示所提算法,“PSO”表示经典粒子群算法。从图中能够看出,随着迭代次数的增加,三种方案均收敛到达稳定值。在迭代初期,“BA”“PSO”两种方案与所提方案的Fitness均呈现下降趋势,迭代次数也大体一致。但所提方案与其它两种相关的经典方案相比,能够以较快的速度获得较低的Fitness值。这是因为所提方案采用混沌映射方法初始化蝙蝠种群,并引入动态惯性权重对蝙蝠个体的速度进行动态更新。因此,所提方案具有比基础蝙蝠算法更强的探索能力,从而得到最优的适应度函数值,达到快速收敛、得到系统最优解的效果。

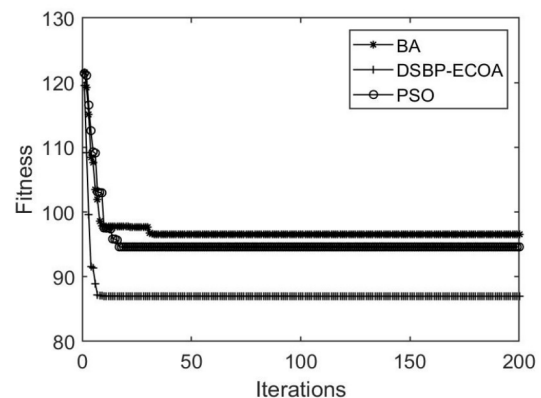


图6 不同方案的Fitness收敛情况对比

5 结束语

为了提高频谱利用率和网络吞吐量,提出了一种基于D2D协同的边缘计算迁移机制。首先,通过对迁移决策和功率分配的联合优化,规划了一个最小化任务完成总能耗的优化问题,针对该混合整数非线性规划问题,提出了一个基于动态感知蝙蝠群体的高效计算迁移算法,根据环境变化,结合惯性权重系数动态调整种群移动速度,从而获得最优迁移决策和功率分配

策略。大量的仿真证实了所提机制的有效性,并与其他基准机制相比,所提算法能有效降低系统能耗。该模型中用户设备种类较为单一,未考虑设备的异构性。此外,用户较为关注的数据隐私等问题也并未解决,未来研究方向将主要针对设备异构、数据隐私保护展开研究。

参考文献:

- [1] LIU Y, PENG M, SHOU G, et al. Toward edge intelligence: multiaccess edge computing for 5G and internet of things [J]. IEEE Internet of Things Journal, 2020, 7(8): 6722–6747.
- [2] DE DONNO M, TANGE K, DRAGONI N. Foundations and evolution of modern computing paradigms: cloud, IoT, edge, and fog [J]. IEEE Access, 2019, 7: 150936–150948.
- [3] LINTHICUM D S. Connecting fog and cloud computing [J]. IEEE Cloud Computing, 2017, 4(2): 18–20.
- [4] REN J, YU G, HE Y, et al. Collaborative cloud and edge computing for latency minimization [J]. IEEE Transactions on Vehicular Technology, 2019, 68(5): 5031–5044.
- [5] AVASALCAI C, TSIGKANOS C, DUSTDAR S. Decentralized resource auctioning for latency-sensitive edge computing [C]//Proceedings of IEEE international conference on edge computing (EDGE). Milan: IEEE, 2019: 72–76.
- [6] GRIFFIN D, PHAN T K, MAINI E, et al. On the feasibility of using current data centre infrastructure for latency-sensitive applications [J]. IEEE Transactions on Cloud Computing, 2020, 8(3): 875–888.
- [7] KHAN F, JAN M A, REHMAN A U, et al. A secured and intelligent communication scheme for IIoT-enabled pervasive edge computing [J]. IEEE Transactions on Industrial Informatics, 2021, 17(7): 5128–5137.
- [8] GOUDARZI M, WU H, PALANISWAMI M, et al. An application placement technique for concurrent IoT applications in edge and fog computing environments [J]. IEEE Transactions on Mobile Computing, 2021, 20(4): 1298–1311.
- [9] ZHAO T, ZHOU S, SONG L, et al. Energy-optimal and delay-bounded computation offloading in mobile edge computing with heterogeneous clouds [J]. China Communications, 2020, 17(5): 191–210.
- [10] XU D, LI Q, ZHU H. Energy-saving computation offloading by joint data compression and resource allocation for mobile-edge computing [J]. IEEE Communications Letters, 2019, 23(4): 704–707.
- [11] CHEN J, CHANG Z, GUO X, et al. Resource allocation and computation offloading for multi-access edge computing with fronthaul and backhaul constraints [J]. IEEE Transactions on Vehicular Technology, 2021, 70(8): 8037–8049.
- [12] GAO M, SHEN R, LI J, et al. Computation offloading with instantaneous load billing for mobile edge computing [J]. IEEE Transactions on Services Computing, 2020, 15(3): 1–13.
- [13] CHA N, WU C, YOSHINAGA T, et al. Virtual edge: exploring computation offloading in collaborative vehicular edge computing [J]. IEEE Access, 2021, 9: 37739–37751.
- [14] FENG H, GUO S, YANG L, et al. Collaborative data caching and computation offloading for multi-service mobile edge computing [J]. IEEE Transactions on Vehicular Technology, 2021, 70(9): 9408–9422.
- [15] GUO H, ZHANG J, LIU J, et al. Energy-aware computation offloading and transmit power allocation in ultradense IoT networks [J]. IEEE Internet of Things Journal, 2019, 6(3): 4317–4329.
- [16] NADEEM L, AZAM M A, AMIN Y, et al. Integration of D2D, network slicing, and MEC in 5G cellular networks: survey and challenges [J]. IEEE Access, 2021, 9: 37590–37612.
- [17] CHENG Y, LIANG C, CHEN Q, et al. Energy-efficient D2D-assisted computation offloading in NOMA-enabled cognitive networks [J]. IEEE Transactions on Vehicular Technology, 2021, 70(12): 13441–13446.
- [18] LIU Y, HU Q, CAI Y, et al. Latency minimization in intelligent reflecting surface assisted D2D offloading systems [J]. IEEE Communications Letters, 2021, 25(9): 3046–3050.
- [19] PENG J, QIU H, CAI J, et al. D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC [J]. IEEE Transactions on Wireless Communications, 2021, 20(8): 4858–4873.
- [20] FANG T, YUAN F, AO L, et al. Joint task offloading, D2D pairing, and resource allocation in device-enhanced MEC: a potential game approach [J]. IEEE Internet of Things Journal, 2022, 9(5): 3226–3237.
- [21] CHAI R, LIN J, CHEN M, et al. Task execution cost minimization-based joint computation offloading and resource allocation for cellular D2D MEC systems [J]. IEEE Systems Journal, 2019, 13(4): 4110–4121.