计算机科学
COMPUTER SCIENCE

超密集物联网络中多任务多步计算卸载算法研究

彭云聪，秦小林，张力戈，顾勇翔

引用本文

周天清，岳亚莉. 超密集物联网络中多任务多步计算卸载算法研究[J]. 计算机科学, 2022, 49(6): 12-18.

ZHOU Tian-qing, YUE Ya-li. Multi-Task and Multi-Step Computation Offloading in Ultra-dense IoT Networks[J]. Computer Science, 2022, 49(6): 12-18.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.211200147

超密集物联网络中多任务多步计算卸载算法研究

周天清　岳亚莉
华东交通大学信息工程学院　南昌 330013

摘　要　随着物联网(Internet of Things,IoT)的迅速发展,各种物联网移动设备(IoT Mobile Device,IMD)需要处理越来越多的计算密集型和延迟敏感型任务,这给移动边缘计算网络带来了新的挑战。为了应对这些挑战,装备移动边缘计算(Mobile Edge Computing,MEC)的超密集物联网应运而生。在该网络中,IMD可将计算密集型任务卸载至边缘计算服务器上进行处理,从而节省自己的计算资源并降低能耗。然而,这样会造成额外的传输时间,进而导致更高的延迟。为了均衡能耗与时延,针对多用户多任务的超密集物联网络,提出了一个最小化能耗和时延的均衡问题,以联合优化用户(IMD)关联、计算卸载和资源分配。为了进一步平衡网络负载,充分利用计算资源,在问题建模时采用多步计算卸载。最后,利用智能算法——自适应粒子群算法(Particle Swarm Optimization,PSO)对所提问题进行求解。相比传统粒子群算法,自适应粒子群算法能降低20%～65%的总开销。

关键词:物联网;移动边缘计算;计算卸载;用户关联;资源分配;智能算法;自适应粒子群算法
中图法分类号　TN929

Multi-Task and Multi-Step Computation Offloading in Ultra-dense IoT Networks

ZHOU Tian-qing and YUE Ya-li
School of Information Engineering,East China Jiaotong University,Nanchang 330013,China

Abstract　With the rapid development of Internet of Things(IoT),various IoT mobile devices(IMDs) need to process more and more computing-intensive and delay-sensitive tasks,which puts forward new challenges for the mobile edge networks. To address these challenges,the MEC-equipped ultra-dense IoT has emerged. In such networks,IMDs can save their computation resources and reduce their energy consumption by offloading computing-intensive tasks to edge computing servers for processing. However,it will result in additional transmission time and higher delay. In view of this,an optimization problem is formulated for finding the trade-off between energy consumption and delay,which jointly considers the user(IMD) association,computation offloading and resource allocation for ultra-dense MEC-enabled IoT. To further balance the network load and fully utilize the computation resources,the optimization problem is finally modeled as multi-step computation offloading one. At last,an intelligent algorithm,adaptive particle swarm optimization(PSO),is utilized to solve the proposed problem. Compared with traditional PSO,the total cost of adaptive PSO reduces by 20%～65%.

Keywords　IoT,MEC,Computing offload,User association,Resource allocation,Intelligent algorithm,Adaptive PSO

1　引言

随着第五代(5th Generation,5G)移动通信技术的快速发展,“万物互联”[1]的时代已经到来。智能手机、车辆、平板电脑、AR(Virtual Reality)眼镜等物联网移动设备普及,从而对无线网络的接入和传输提出了新的要求。超密集异构蜂窝网络的研究便是在这种背景下展开的,它可以将热点地区的网络容量提高千倍以上[2]。与此同时,我国在2019年全面推进了5G通信的商业用途,在移动物联网、工业物联网、车联网、IPv6等方面都取得了不错的成果[3-4]。然而,随着移动通信和互联网技术的发展,这些物联网移动设备需要不断地增加新业务新应用,这些激增的需求大多都是计算密集型和延时敏感型的。因此,对移动网络在时延和可靠性等多个方面提出了更高的要求。同时,这些计算密集型和时延敏感型任务对实时性的要求很高,但传统的移动通信网络架构对于现阶段的物联网移动设备而言,已经显现出计算能力不足、电池容量过小等问题,移动设备已经无法满足使用需求。若这些物联网移动设备终端选择自己来执行这些计算密集型或时延敏感型任务,则将无法保障使用者的服务质量体验。

为了解决上述问题,近年来,移动边缘计算技术作为超低

到稿日期:2021-12-13　返修日期:2022-02-16
基金项目:国家自然科学基金(61861017,62171119);国家重点研发计划(2020YFB1807201)
This work was supported by the National Natural Science Foundation of China(61861017,62171119) and National Key R&D Program of China(2020YFB1807201).
通信作者:周天清(zhoutian930@163.com)

时延、高带宽、高能效、可靠的 5G 通信关键技术，在学术界和工业界引起了广泛的研究和讨论[5]。所谓的移动边缘计算指通过部署小基站(Small Base Station，SBS)，如微微基站和家庭基站等设备，将云计算能力扩展到网络边缘，从而提供更高的计算和存储能力。利用无线接入网络，MEC 具有实现更低的延迟、提供更灵活的计算体验和降低能耗成本等优势。

物联网移动设备将任务卸载到边缘服务器进行计算时，有可能会产生更高的上传时延，而物联网移动设备选择自己来计算时可能会带来更高的能耗。为了解决上述问题，移动边缘计算的通信时延和系统能耗优化是当前的一个主要研究方向。国内外有不少学者进行了相关研究。文献[6]考虑了具有多个 MEC 服务器的网络场景，提出使用一个三维决策矩阵来决定任务卸载的时间以及目标服务器，以做出最佳卸载决策，从而最小化应用程序的延迟。文献[7]在绿色和低延迟的物联网环境中提出了一个移动感知的分层 MEC 框架，并且采用了博弈论方法进行计算卸载，在优化服务提供商效用的同时，降低了智能设备的能量成本并缩短了任务执行时间。文献[8]研究了联合任务分配和资源分配的问题，设计了在时间约束条件下的移动终端能耗最小化方案。文献[9]用多个移动设备将任务上传到单个小区中的 MEC 服务器，并且在有限服务器资源的场景下，考虑为移动设备分配无线信道以节省移动设备能耗，最终提出了一种选择最大节约能源的算法。文献[10]基于 NOMA(Non-Orthogonal Multiple Access)-MEC 的车联网系统，提出了一种基于博弈论和 Q 学习的任务卸载、迁移与缓存优化策略，采用合作博弈算法获得最优用户分组以实现卸载时延优化。

不同的计算卸载方式对时延和能耗的影响也不容小觑，因此对计算卸载的研究较多。文献[11]同时考虑了 MEC 服务器的无线电资源和计算资源的分配以及任务延迟需求的多样性。文献[12]针对具有移动边缘计算功能的超密异构网络，提出了分布式的联合计算卸载和资源分配优化方案。它可以使边缘服务器执行的计算任务量最大化，同时将计算任务消耗的能量和任务处理延迟最小化。对于资源分配问题，文献[13]在移动设备的有限资源约束下根据应用程序运行队列的信息，通过计算卸载时延和本地执行时延的对比，获得最优资源分配策略以最小化应用运行时延。文献[14]提出了一种可满足不同要求的自适应计算卸载策略，在保证满足延迟约束的条件下达到降低所有移动设备任务卸载总成本的目的。文献[15]在超密集多用户多设备的物联网网络中，联合用户关联、计算卸载和资源分配，采用多步计算卸载方案最小化全网络能量消耗，并通过改进的层次自适应搜索算法对问题进行求解。

无论是在计算卸载、资源分配，还是时延和能耗的优化等方面，虽然对移动边缘计算进行了较多研究，但是在超密集异构蜂窝网络场景中的研究还是较少。本文则在超密集网络下联合计算卸载、资源分配、带宽分配以及用户关联等来最小化时延和能耗，具体工作如下：

(1)考虑了超密集物联网络中的多用户多任务移动边缘计算网络模型，融合了等频带划分以消除不同网络层间和层内的干扰。在这种网络模型下，设计了一步式和两步式计算卸载方案，并且采用部分卸载模式来处理任务。在一步式计算卸载中，用户可以直接与基站(Base Station，BS)关联，将计算任务部分卸载到 BS 进行处理；而在两步计算卸载中，用户首先将任务部分卸载到 SBS 上进行处理，然后 SBS 把自己的任务部分卸载到 MBS 进行处理。迄今为止，这种网络模型和计算卸载方式相结合的研究并不多，国内的相关研究则更少。

(2)针对上述模型，在利用根据 CPU 占比分配计算资源的前提下，提出了一个最小化加权时延和能耗的问题，以联合优化用户关联、功率控制、计算卸载和频带资源划分。根据研究现状容易发现，这在上述模型中属于一类全新的研究问题。

(3)考虑到上述优化问题为大规模非线性混合整数的形式，采用自适应粒子群算法求解，该算法属于智能算法。为了使用该算法，必须合理有效地完成粒子编码，再进行算法设计。为了验证所设计算法的有效性，仿真中引入其他算法以进行比较。

2 多用户多任务超密集物联网络模型

本节首先介绍网络模型，即超密集物联网络下多用户多任务的移动边缘计算模型，然后介绍通信模型、计算模型和多任务模型[15]。

2.1 网络模型

本文考虑超密集物联网络下多用户多任务的移动边缘计算模型[15]，如图 1 所示。在该网络中，部署了一个宏基站(Macro Base Station，MBS)和 $\bar{S}$ 个同类型的 SBS，其中 U 个 IMD 的集合表示为 $\mathcal{U}=\{1,2,\cdots,U\}$，SBS 的标签(序号)集合为 $\bar{\mathcal{S}}=\{1,2,\cdots,\bar{S}\}$，0 表示 MBS 的序号，所有基站的集合为 $\mathcal{S}=\bar{\mathcal{S}}\cup\{0\}$，$S=\bar{S}+1$ 表示基站数目。此外，假定 SBS 与 MBS 之间建立了有线回程链路，每个 IMD 拥有 K 个需要独立处理的任务，任务集合表示为 $\mathcal{K}=\{1,2,\cdots,K\}$。值得注意的是，在超密集物联网络中，IMD 的数量小于等于 SBS 的数量。

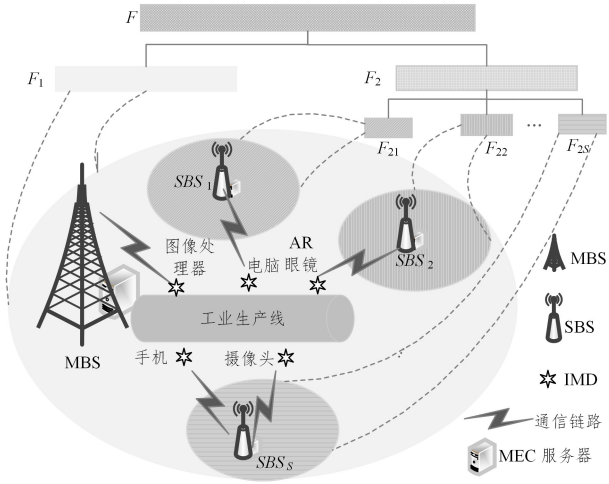

图 1 多用户多任务超密集物联网络模型

Fig. 1 Multi-user and multi-task ultra-dense IoT model

如图 1 所示，本文引入了一种高效的干扰管理机制[15-16]。具体而言，系统总频带 F 被划分为 F_1 和 F_2 两部分，其中频带 F_1 被用于 MBS，频带 F_2 被用于 SBS。值得注意的是，F_2 被均等地分配给每个 SBS，而每个 SBS 的频带又被均等地分配给其所关联的 IMD。频带 F，F_1 和 F_2 的带宽分别为 W，λW 和

$(1-\lambda)W$，其中 $0<\lambda<1$ 是频带划分因子。通过这样的设置，不同基站层间干扰与同类基站层内干扰被完全消除。尽管这种干扰管理机制的频谱利用率相对较低，但分配给每个 IMD 的频带资源应该足够，原因在于，在超密集物联网络中，每个 SBS 最多只能服务一个 IMD。

2.2 通信模型

如果 IMD_i（第 i 个 IMD）与 SBS_j（第 j 个 SBS）关联，则 IMD_i 至 SBS_j 的上行信噪比为：

$$SINR_{ij}=p_i\lambda_{ij}/\sigma^2,\forall j\in\bar{\mathcal{S}} \tag{1}$$

其中，p_i 表示 IMD_i 的发射功率，λ_{ij} 表示 IMD_i 与 SBS_j 关联的信道增益，σ^2 表示噪声功率。

因此，IMD_i 至 SBS_j 的上行数据速率为：

$$R_{ij}=\frac{(1-\lambda)W}{\bar{S}\sum\limits_{m\in\mathcal{U}}x_{mj}}\log_2(1+SINR_{ij}) \tag{2}$$

其中，$x_{ij}\in\{0,1\}$，$i\in\mathcal{U}$，$j\in S$ 表示 IMD_i 与 SBS_j 间的关联指示变量；$x_{ij}=1$ 表示 IMD_i 与 SBS_j 相关联，反之 $x_{ij}=0$。因此，$\sum\limits_{m\in\mathcal{U}}x_{mj}$ 表示与 SBS_j 关联的 IMD 总数，$(1-\lambda)W/\bar{S}\sum\limits_{m\in\mathcal{U}}x_{mj}$ 表示 SBS_j 分配给其所关联 IMD 的频带宽度。

类似地，若 IMD_i 直接与 MBS_0 关联，则 IMD_i 至 MBS_0 的上行信噪比为：

$$SINR_{i0}=p_i\lambda_{i0}/\sigma^2 \tag{3}$$

其中，λ_{i0} 表示 IMD_i 与 MBS 之间的信道增益。

因此，IMD_i 至 MBS 的上行数据速率为：

$$R_{i0}=\lambda W\log(1+SINR_{i0})/\sum_{m\in\mathcal{U}}x_{m0} \tag{4}$$

其中，$\sum\limits_{m\in\mathcal{U}}x_{m0}$ 表示与 MBS 相关联的 IMD 总数。此时，$\lambda W/\sum\limits_{m\in\mathcal{U}}x_{m0}$ 表示 MBS 分配给其所关联 IMD 的频带宽度。

2.3 计算模型

从能耗和任务处理时间的角度出发，本节将介绍计算模型[15]。假定 IMD_i 的一个应用程序由 K 个相互独立的计算任务组成，且它的第 k 个任务（任务 k）可表示为 $\psi_{ik}\triangleq(\theta_{ik},c_{ik})$，其中 θ_{ik} 表示 IMD_i 的任务 k 的数据量大小，c_{ik} 为处理一比特数据所需要的 CPU 周期。

为了充分利用网络计算资源，考虑了两步式和一步式计算卸载方案。在两步式计算卸载方案中，第一步是 IMD_i 将任务 k 的部分卸载到某个 SBS_j 上进行处理，第二步则是 SBS_j 将其所接受的任务 k 部分卸载到 MBS 上进行处理。具体而言，若 IMD_i 与 SBS_j 关联，那么它的任务 k 将有 $\bar{\theta}_{ijk}$ 的数据任务卸载到 SBS_j 上进行处理，而剩余 $\theta_{ik}-\bar{\theta}_{ijk}$ 的数据任务将会在本地完成。此外，SBS_j 将从 $\bar{\theta}_{ijk}$ 中卸载 $\hat{\theta}_{ijk}$ 的数据任务至 MBS 上进行处理，而剩余 $\bar{\theta}_{ijk}-\hat{\theta}_{ijk}$ 的数据任务则将交由 SBS_j 处理。在一步式卸载中，IMD_i 可以直接与 MBS 关联。因此，$\bar{\theta}_{i0k}$ 的数据任务将会被卸载到 MBS 上进行处理，而剩余 $\theta_{ik}-\bar{\theta}_{i0k}$ 的数据任务则留在本地处理。接下来将讨论不同情况下的时延和能耗。

（1）本地计算。当 IMD_i 与 SBS_j 关联时，那么其任务 k 需在本地执行的数据量为 $\theta_{ik}-\bar{\theta}_{ijk}$。此时，$IMD_i$ 在计算任务 k 时的执行时延 T_{ijk}^{LOC} 为：

$$T_{ijk}^{\mathrm{LOC}}=(\theta_{ik}-\bar{\theta}_{ijk})c_{ik}/f_{ik} \tag{5}$$

其中，f_{ik} 表示 IMD_i 分配给任务 k 的计算能力（CPU 周期每秒）。

为了提高计算资源的利用率，IMD_i 根据任务 CPU 占比给任务 k 分配计算资源[14]。具体而言，IMD_i 分配给任务 k 的计算能力 f_{ik} 为：

$$f_{ik}=\frac{\sum\limits_{j\in\mathcal{S}}x_{ij}(\theta_{ik}-\bar{\theta}_{ijk})c_{ik}}{\sum\limits_{j\in\mathcal{S}}\sum\limits_{l\in\mathcal{K}}x_{ij}(\theta_{il}-\bar{\theta}_{ijl})c_{il}}F_i^{\mathrm{UE}} \tag{6}$$

其中，F_i^{UE} 表示 IMD_i 的最大计算能力。

为了在本地处理任务 k 的剩余数据量 $\theta_{ik}-\bar{\theta}_{ijk}$，$IMD_i$ 需消耗的能量 E_{ijk}^{LOC} 为：

$$E_{ijk}^{\mathrm{LOC}}=\xi(\theta_{ik}-\bar{\theta}_{ijk})c_{ik}f_{ik}^2 \tag{7}$$

其中，ξ 是有效开关电容，它取决于芯片架构。

（2）卸载到 SBS。在两步式计算卸载模型中，IMD_i 先与 SBS_j 关联，再将其任务 k 部分卸载至 MBS 进行计算。此时，完成 IMD_i 任务 k 的时延由 4 部分组成。具体而言，第一部分为传输数据量 $\theta_{ik}-\bar{\theta}_{ijk}$ 时所产生的上传时延，第二部分为 SBS_j 处理数据量 $\bar{\theta}_{ijk}-\hat{\theta}_{ijk}$ 的计算时延，第三部分为 SBS_j 将数据量 $\hat{\theta}_{ijk}$ 上传到 MBS 的传输时延，第四部分为 MBS 处理数据量 $\hat{\theta}_{ijk}$ 的时延。因此，当 IMD_i 与 SBS_j 关联时，完成 IMD_i 任务 k 的时延 T_{ijk}^{SBS} 为：

$$T_{ijk}^{\mathrm{SBS}}=\frac{\bar{\theta}_{ijk}}{R_{ij}}+\frac{(\bar{\theta}_{ijk}-\hat{\theta}_{ijk})c_{ik}}{\bar{f}_{ijk}}+\frac{\hat{\theta}_{ijk}}{r_0}+\frac{\hat{\theta}_{ijk}c_{ik}}{\bar{f}_{i0k}} \tag{8}$$

其中，r_0 为 SBS 与 MBS 间的有线回程数据速率；$\bar{f}_{ijk}$ 表示 SBS_j 分配给 IMD_i 的任务 k 的计算能力；$\bar{f}_{i0k}$ 表示 MBS 分配给 IMD_i 的任务 k 的计算能力。

当 IMD_i 与 SBS_j 关联时，SBS_j 根据任务的 CPU 占比给任意 IMD_i 的任务 k 分配计算能力。具体而言，当 IMD_i 与 SBS_j 关联时，SBS_j 分配给 IMD_i 的任务 k 的计算能力 $\bar{f}_{ijk}$ 为：

$$\bar{f}_{ijk}=\frac{(\bar{\theta}_{ijk}-\hat{\theta}_{ijk})c_{ik}}{\sum\limits_{m\in\mathcal{U}}x_{mj}\sum\limits_{l\in\mathcal{K}}(\bar{\theta}_{ijl}-\hat{\theta}_{ijl})c_{ml}}F_j^{\mathrm{BS}} \tag{9}$$

其中，F_j^{BS} 表示 SBS_j 的最大计算能力。

在两步式计算卸载模型中，当某个 IMD 与 SBS 关联后，SBS 可以将所接收任务继续卸载至 MBS 上进行处理。当然，某些 IMD 也可以直接与 MBS 关联。因此，在 MBS 上处理的数据包括两部分：一部分为 IMD 与 SBS 关联后从 SBS 上传至 MBS 的数据任务，其数据总量为 $\chi_1=\sum\limits_{m\in\mathcal{U}}\sum\limits_{j\in\bar{\mathcal{S}}}x_{mj}\sum\limits_{l\in\mathcal{K}}\hat{\theta}_{mjl}c_{ml}$；另一部分为 IMD 与 MBS 关联后直接将部分数据任务卸载至 MBS，其数据总量为：

$$\chi_2=\sum_{m\in\mathcal{U}}x_{m0}\sum_{l\in\mathcal{K}}\theta_{m0l}c_{ml}$$

根据任务 CPU 占比[14]，MBS 分配给 IMD_i 的任务 k 的计算能力 $\bar{f}_{i0k}$ 为：

$$\bar{f}_{i0k}=\frac{\sum\limits_{j\in\bar{\mathcal{S}}}x_{ij}\hat{\theta}_{ijk}c_{ik}+x_{i0}\theta_{i0k}c_{ik}}{\chi_1+\chi_2}F_0^{\mathrm{BS}} \tag{10}$$

其中，F_0^{BS} 表示 MBS 的最大计算能力。

当 IMD_i 通过两步式计算卸载来处理它的任务 k 时，其能耗也由 4 部分组成。具体而言，第一部分是 IMD_i 通过无线

信道卸载数据量 $\theta_{ik}-\bar{\theta}_{ijk}$ 至 SBS_j 时所产生的传输能耗，第二部分为 SBS_j 处理 $\bar{\theta}_{ijk}-\hat{\theta}_{ijk}$ 时所产生的能耗，第三部分是 SBS_j 通过有线链路将 $\hat{\theta}_{ijk}$ 卸载到 MBS 上的传输能耗，第四部分是在 MBS 处理 $\hat{\theta}_{ijk}$ 时所产生的能耗。因此，当 IMD_i 与 SBS_j 关联时，完成 IMD_i 任务 k 的能耗 E_{ijk}^{SBS} 为：

$$E_{ijk}^{SBS}=p_i\bar{\theta}_{ijk}/R_{ij}+(\bar{\theta}_{ijk}-\hat{\theta}_{ijk})c_{ik}\varepsilon_j+\zeta\hat{\theta}_{ijk}/r_0+\hat{\theta}_{ijk}c_{ik}\varepsilon_0 \tag{11}$$

其中，ζ 表示有线链路中单位时间内的功耗，ε_j 和 ε_0 分别表示 SBS_j 和 MBS 的每个 CPU 周期的能耗。

(3)卸载到 MBS。当任意 IMD_i 直接与 MBS 关联时，其任务 k 的完成时延包括 IMD_i 将任务上传至 MBS 的上行传输时延和 MBS 处理所接收 IMD_i 的任务 k 的时延。因此，当 IMD_i 直接与 MBS 关联时，其任务 k 的完成时延 T_{i0k}^{MBS} 为：

$$T_{i0k}^{MBS}=\bar{\theta}_{i0k}/R_{i0}+\bar{\theta}_{i0k}c_{ik}/\bar{f}_{i0k} \tag{12}$$

在此过程中，所消耗的能量 E_{i0k}^{MBS} 为：

$$E_{i0k}^{MBS}=p_i\bar{\theta}_{i0k}/R_{i0}+\bar{\theta}_{i0k}c_{ik}\varepsilon_0 \tag{13}$$

其中，等式右边第一项为 IMD_i 的任务 k 上传至 MBS 所产生的传输能耗，第二项为 MBS 处理接收到 IMD_i 的任务 k 所产生的能耗。

2.4 多任务模型

不失一般性，假设 IMD 任务按顺序执行，即任务 k 需要在任务 $k-1$ 完成之后才能被执行。此外，假设局部计算和计算卸载操作同步进行，这就意味着，IMD_i 的总时延 T_i^{Sq} 为局部计算和计算卸载时延中的最大值，即：

$$T_i^{Sq}=\sum_{k\in\mathcal{K}}\max\left(\sum_{j\in\mathcal{S}}x_{ij}T_{ijk}^{LOC},\sum_{j\in\bar{\mathcal{S}}}x_{ij}T_{ijk}^{SBS}+x_{i0}T_{i0k}^{MBS}\right) \tag{14}$$

然而，IMD_i 的能耗 E_i^{Sq} 则是局部执行与计算卸载能耗之和，即：

$$E_i^{Sq}=\sum_{k\in\mathcal{K}}\left(\sum_{j\in\mathcal{S}}x_{ij}E_{ijk}^{LOC}+\sum_{j\in\bar{\mathcal{S}}}x_{ij}E_{ijk}^{SBS}+x_{i0}E_{i0k}^{MBS}\right) \tag{15}$$

3 问题规划

针对超密集物联网，本文考虑了多用户多任务的计算卸载方案设计。具体而言，通过最小化加权时延与能耗之和来最优化卸载决策 $\boldsymbol{X}=\{x_{ij},\forall i\in\mathcal{U},\forall j\in\mathcal{S}\}$，上行链路发射功率 $\boldsymbol{p}=\{p_i,\forall i\in\mathcal{U}\}$，卸载至 BS 的数据量 $\bar{\boldsymbol{\theta}}=\{\bar{\theta}_{ijk},\forall i\in\mathcal{U},\forall j\in\mathcal{S},\forall k\in\mathcal{K}\}$，频带划分因子 λ，及 SBS 卸载至 MBS 的数据量 $\hat{\boldsymbol{\theta}}=\{\hat{\theta}_{ijk},\forall i\in\mathcal{U},\forall j\in\mathcal{S},\forall k\in\mathcal{K}\}$。具体优化问题的规划如下：

$$\begin{aligned}
&\min_{\boldsymbol{X},\boldsymbol{p},\bar{\boldsymbol{\theta}},\hat{\boldsymbol{\theta}},\lambda}\ \Gamma(\boldsymbol{X},\boldsymbol{p},\bar{\boldsymbol{\theta}},\hat{\boldsymbol{\theta}},\lambda)=\sum_{i\in\mathcal{U}}\{\omega_iT_i^{sq}+(1-\omega_i)E_i^{sq}\}\\
&\text{s.t.}\ \ C1:\sum_{j\in\mathcal{S}}x_{ij}=1,\forall i\in\mathcal{U}\\
&\qquad C2:0\leqslant p_i\leqslant p_i^{\max},\forall i\in\mathcal{U}\\
&\qquad C3:x_{ij}\in\{0,1\},\forall i\in\mathcal{U},j\in\mathcal{S}\\
&\qquad C4:0\leqslant\sum_{j\in\bar{\mathcal{S}}}x_{ij}\hat{\theta}_{ijk}\leqslant\sum_{j\in\bar{\mathcal{S}}}x_{ij}\bar{\theta}_{ijk}\leqslant\theta_{ik},\forall i\in\mathcal{U},k\in\mathcal{K}\\
&\qquad C5:0\leqslant x_{i0}\bar{\theta}_{i0k}\leqslant\theta_{ik},\forall i\in\mathcal{U},k\in\mathcal{K}\\
&\qquad C6:0\leqslant\lambda\leqslant1
\end{aligned} \tag{16}$$

其中，$0\leqslant\omega_i\leqslant1$ 为 IMD_i 的权重，用于平衡延迟和能耗；C_1 和 C_3 表示一个 IMD 只能与一个基站相关联；C_2 表示 IMD_i 的非负发射功率不能大于最大发射功率 $p_i^{\max}$；C_4 表示当 IMD_i 将任务 k 部分卸载至 SBS_j 上进行处理时，卸载任务量 $\bar{\theta}_{ijk}$ 小于总任务量 θ_{ik}，从 SBS_j 卸载至 MBS 的任务量 $\hat{\theta}_{ijk}$ 不能超过 $\bar{\theta}_{ijk}$；C_5 表示当 IMD_i 直接与 MBS 关联，任务 k 卸载至 MBS 上的任务量 $\bar{\theta}_{i0k}$ 不能超过 IMD_i 的总量 θ_{ik}；C_6 表示非负的资源划分因子小于 1。

值得注意的是，式(16)具备非线性混合整数形式，需要优化的参数也是高度耦合的，它为非凸优化问题。在现实生活中，很难找到这类问题的最优解，而且其复杂度随着优化参量规模的增大而增加。显然，通过测试所有可能解来获取式(16)的解的方法不切实际且不可行。

4 自适应粒子群算法

粒子群算法是通过模拟鸟类觅食行为的智能优化方法，它常用于求各类优化问题，特别是高度复杂问题。在 PSO 中，每个优化问题的潜在解都可以被视为高维搜索空间中的一个点，这些点被称为“粒子”，所有粒子都有一个与目标函数紧密相关的适应值，且每个粒子还有一个速度决定它们飞行的方向和距离。在 PSO 中，粒子追随当前最优粒子在解空间中行进。然而，传统的粒子群算法容易陷入局部最优解。为求解问题(16)以获取其最优解，自适应粒子群算法[17]被视为首选。

4.1 粒子编码及初始化

为了利用自适应粒子群算法求解问题(16)，需要合理完成粒子编码。首先，定义目标函数为粒子的适应度值，再将优化参量 $\boldsymbol{X},\boldsymbol{p},\bar{\boldsymbol{\theta}},\hat{\boldsymbol{\theta}}$ 与 λ 分别编码为 $\boldsymbol{O}_z,\boldsymbol{Q}_z,\boldsymbol{B}_z,\boldsymbol{D}_z$ 与 l_z，其中 $\boldsymbol{O}_z=\{o_{zi},i\in\mathcal{U}\}$，$o_{zi}$ 表示在粒子 z 中与 IMD_i 关联的 BS 序号；$\boldsymbol{Q}_z=\{q_{zi},i\in\mathcal{U}\}$，$q_{zi}$ 表示粒子 z 中 IMD_i 的发射功率；$\boldsymbol{B}_z=\{b_{zi},i\in\bar{\mathcal{U}}\}$，$b_{zi}$ 表示粒子 z 中虚拟 IMD_i 卸载至 BS 上的数据量；$\boldsymbol{D}_z=\{d_{zi},i\in\bar{\mathcal{U}}\}$，$d_{zi}$ 表示粒子 z 中虚拟 IMD_i 卸载至 MBS 上的数据量；l_z 是粒子 z 中的频带划分因子。值得注意的是，虚拟用户集合为 $\bar{\mathcal{U}}=\{1,2,\cdots,K,K+1,K+2,\cdots,2K,\cdots,\bar{U}\}$，其中 $\bar{U}=UK$。定义粒子 z 的适应度函数为：

$$F(\boldsymbol{O}_z,\boldsymbol{Q}_z,\boldsymbol{B}_z,\boldsymbol{D}_z,l_z)=-\Gamma(\boldsymbol{O}_z,\boldsymbol{Q}_z,\boldsymbol{B}_z,\boldsymbol{D}_z,l_z) \tag{17}$$

与此同时，根据 C_1-C_6 的约束条件，对任意粒子 z 进行如下初始化操作：

$$\begin{cases}
o_{zi}^0=\text{randi}(\mathcal{S}),\forall i\in\mathcal{U}\\
q_{zi}^0=\text{rand}(p_i^{\max}),\forall i\in\mathcal{U}\\
b_{zi}^0=\text{rand}(\theta_{mk}),\forall i\in\bar{\mathcal{U}}\\
d_{zi}^0=\text{rand}(b_{zi}^0),\forall i\in\bar{\mathcal{U}}\\
l_z^0=\text{rand}(1)\\
[m,k]=\text{ind2sub}([UK],i)
\end{cases} \tag{18}$$

其中，$[m,k]=\text{ind2sub}([UK],i)$ 表示返回与线性索引 i 对应的 $U\times K$ 矩阵中的行下标 m 和列下标 k；randi($\mathcal{S}$)表示从集合 $\mathcal{S}$ 中随机输出一个元素；rand(a)表示从 0 到 a 间随机生成一个数。

4.2 粒子速度及位置更新

任何粒子具有两个属性，即速度和位置，其中速度反映粒子移动的快慢，位置代表粒子的移动方向。每个粒子在搜索

空间中单独地搜寻最优解，将其记为当前个体极值，并将个体极值与整个粒子群中的其他粒子共享。找到最优的个体极值作为整个粒子群的当前全局最优解，粒子群中的所有粒子根据自己找到的当前个体极值和整个粒子群共享的当前全局最优解来调整自己的速度和位置。鉴于此，任意粒子 z 的速度可更新为：

$$ov_{zi}^{t+1}=\vartheta_z ov_{zi}^t+w_1\eta_{zi}(op_{zi}^t-ol_{zi}^t)+w_2\hat{\eta}_{zi}(og_i^t-ol_{zi}^t),\quad \forall i\in\mathcal{U} \tag{19}$$

$$qv_{zi}^{t+1}=\vartheta_z qv_{zi}^t+w_1\eta_{zi}(qp_{zi}^t-ql_{zi}^t)+w_2\hat{\eta}_{zi}(qg_i^t-ql_{zi}^t),\quad \forall i\in\mathcal{U} \tag{20}$$

$$bv_{zi}^{t+1}=\vartheta_z bv_{zi}^t+w_1\gamma_{zi}(bp_{zi}^t-bl_{zi}^t)+w_2\hat{\gamma}_{zi}(bg_i^t-bl_{zi}^t),\quad \forall i\in\bar{\mathcal{U}} \tag{21}$$

$$dv_{zi}^{t+1}=\vartheta_z dv_{zi}^t+w_1\gamma_{zi}(dp_{zi}^t-dl_{zi}^t)+w_2\hat{\gamma}_{zi}(dg_i^t-dl_{zi}^t),\quad \forall i\in\bar{\mathcal{U}} \tag{22}$$

$$lv_z^{t+1}=\vartheta_z lv_z^t+w_1\alpha_z(lp_z^t-ll_z^t)+w_2\hat{\alpha}_z(lg^t-ll_z^t) \tag{23}$$

其中，ov_{zi}^t，qv_{zi}^t，bv_{zi}^t，dv_{zi}^t 和 lv_z^t 分别表示 o_{zi}，q_{zi}，b_{zi}，d_{zi} 和 l_z 在第 t 次迭代时的速度；ol_{zi}^t，ql_{zi}^t，bl_{zi}^t，dl_{zi}^t 和 ll_z^t 分别表示 o_{zi}，q_{zi}，b_{zi}，d_{zi} 和 l_z 在第 t 次迭代时的位置；ϑ_z 表示粒子 z 在第 t 次迭代时的惯性权重；$0<w_1<1$ 和 $0<w_2<1$ 分别为学习因子和社会因子；η_{zi}，$\hat{\eta}_{zi}$，γ_{zi}，$\hat{\gamma}_{zi}$，α_z 和 $\hat{\alpha}_z$ 为随机数。

为了便于后续描述，令 $\boldsymbol{op}_z^t=\{op_{zi}^t,\forall i\in\mathcal{U}\}$，$\boldsymbol{qp}_z^t=\{qp_{zi}^t,\forall i\in\mathcal{U}\}$，$\boldsymbol{bp}_z^t=\{bp_{zi}^t,\forall i\in\bar{\mathcal{U}}\}$，$\boldsymbol{dp}_z^t=\{dp_{zi}^t,\forall i\in\bar{\mathcal{U}}\}$，它们和 lp_z^t 是粒子 z 在第 t 次迭代时的历史最优位置；此外，考虑 $\boldsymbol{og}^t=\{og_i^t,\forall i\in\mathcal{U}\}$，$\boldsymbol{qg}^t=\{qg_i^t,\forall i\in\mathcal{U}\}$，$\boldsymbol{bg}^t=\{bg_i^t,\forall i\in\bar{\mathcal{U}}\}$，$\boldsymbol{dg}^t=\{dg_i^t,\forall i\in\bar{\mathcal{U}}\}$，它们和 lg^t 为粒子群在第 t 次迭代时的最优位置。

然后，惯性权重[15]更新为：

$$\vartheta_z^{t+1}=\begin{cases}\vartheta_z+t(\vartheta^{\max}-\vartheta^{\min})/T, & \text{最佳粒子}\\ \vartheta_z-t(\vartheta^{\max}-\vartheta^{\min})/T, & \text{其他}\end{cases} \tag{24}$$

其中，$\vartheta^{\max}$ 和 $\vartheta^{\min}$ 分别为惯性权重的最大值和最小值；T 为自适应粒子群算法的迭代次数。

由更新过的粒子速度可以更新任何粒子 z 的位置，表达式如下：

$$ol_{zi}^{t+1}=\mathrm{round}(ol_{zi}^t+ov_{zi}^{t+1}),\forall i\in\mathcal{U} \tag{25}$$

$$ql_{zi}^{t+1}=ql_{zi}^t+qv_{zi}^{t+1},\forall i\in\mathcal{U} \tag{26}$$

$$bl_{zi}^{t+1}=bl_{zi}^t+bv_{zi}^{t+1},\forall i\in\bar{\mathcal{U}} \tag{27}$$

$$dl_{zi}^{t+1}=dl_{zi}^t+dv_{zi}^{t+1},\forall i\in\bar{\mathcal{U}} \tag{28}$$

$$ll_z^{t+1}=ll_z^t+lv_z^{t+1} \tag{29}$$

其中，round(a)表示对 a 向下去整的函数。为了保持全局最佳粒子 $\bar{z}$ 向局部最小移动，根据文献[18]的规则，粒子 $\bar{z}$ 的速度再次更新为：

$$ov_{\bar{z}i}^{t+1}=-ol_{\bar{z}i}^t+og_i^t+w_3 ov_{\bar{z}i}^t+\beta^t(1-2\delta_{\bar{z}i}),\forall i\in\mathcal{U} \tag{30}$$

$$qv_{\bar{z}i}^{t+1}=-ql_{\bar{z}i}^t+qg_i^t+w_3 qv_{\bar{z}i}^t+\beta^t(1-2\delta_{\bar{z}i}),\forall i\in\mathcal{U} \tag{31}$$

$$bv_{\bar{z}i}^{t+1}=-bl_{\bar{z}i}^t+bg_i^t+w_3 bv_{\bar{z}i}^t+\beta^t(1-2\bar{\delta}_{\bar{z}i}),\forall i\in\bar{\mathcal{U}} \tag{32}$$

$$dv_{\bar{z}i}^{t+1}=-dl_{\bar{z}i}^t+dg_i^t+w_3 dv_{\bar{z}i}^t+\beta^t(1-2\bar{\delta}_{\bar{z}i}),\forall i\in\bar{\mathcal{U}} \tag{33}$$

$$lv_{\bar{z}}^{t+1}=-ll_{\bar{z}}^t+lg^t+w_3 lv_{\bar{z}}^t+\beta^t(1-2\hat{\delta}_{\bar{z}}) \tag{34}$$

其中，β^t 是第 t 次迭代时的比例因子；w_3 是常系数；$0\leqslant\delta_{\bar{z}i}\leqslant1$，$0\leqslant\bar{\delta}_{\bar{z}i}\leqslant1$ 和 $0\leqslant\hat{\delta}_{\bar{z}}\leqslant1$ 是[0,1]之间的随机数。

根据式(30)－式(34)，更新粒子位置为：

$$ol_{\bar{z}i}^{t+1}=\mathrm{round}(og_i^t+w_3 ov_{\bar{z}i}^t+\beta^t(1-2\delta_{\bar{z}i})),\forall i\in\mathcal{U} \tag{35}$$

$$ql_{\bar{z}i}^{t+1}=qg_i^t+w_3 qv_{\bar{z}i}^t+\beta^t(1-2\delta_{\bar{z}i}),\forall i\in\mathcal{U} \tag{36}$$

$$bl_{\bar{z}i}^{t+1}=bg_i^t+w_3 bv_{\bar{z}i}^t+\beta^t(1-2\bar{\delta}_{\bar{z}i}),\forall i\in\bar{\mathcal{U}} \tag{37}$$

$$dl_{\bar{z}i}^{t+1}=dg_i^t+w_3 dv_{\bar{z}i}^t+\beta^t(1-2\bar{\delta}_{\bar{z}i}),\forall i\in\bar{\mathcal{U}} \tag{38}$$

$$ll_{\bar{z}}^{t+1}=lg^t+w_3 lv_{\bar{z}}^t+\beta^t(1-2\hat{\delta}_{\bar{z}}) \tag{39}$$

其中，为驱使 PSO 随机搜索全局最佳位置 $\boldsymbol{gbest}=\{\boldsymbol{og},\boldsymbol{qg},\boldsymbol{bg},\boldsymbol{dg},lg\}$ 周围区域，比例因子 β 可利用式(40)所示的规则来更新，即：

$$\beta^{t+1}=\begin{cases}2\beta^t, & N_1>n_1\\ 0.5\beta^t, & N_2>n_2\\ \beta^t, & \text{其他}\end{cases} \tag{40}$$

其中，N_1 表示连续成功的次数；N_2 表示连续失败的次数；n_1 和 n_2 表示阈值参数。值得注意的是，$F(\boldsymbol{gbest}^t)=F(\boldsymbol{gbest}^{t-1})$ 显示搜索失败，其他情况暗示搜索成功。

综上所述，基于自适应 PSO 的式(16)的求解算法的具体过程如算法 1 所示。值得注意的是，不同于传统 PSO 算法，Adaptive PSO 算法自适应更新惯性权重，并二次优化全局最佳粒子。正因如此，Adaptive PSO 能找到比 PSO 更好的式(16)的解。

算法 1 自适应粒子群算法(Adaptive PSO)

输入：信干噪比，T，$\vartheta^{\max}$，$\vartheta^{\min}$，N_1，N_2，n_1，n_2，$p_i^{\max}$，θ_{ik}，c_{ik} 等

输出：$\boldsymbol{gbest}^T$

1. 利用式(18)初始化所有粒子。
2. 设置 t=1。
3. 初始化所有粒子的速度。
4. 初始化个体最佳粒子的速度。
5. 初始化所有粒子的位置。
6. 找到当前种群中的全局最佳粒子。
7. 若 t<T，则执行后续步骤，否则终止算法。
8. 利用式(24)更新惯性权重。
9. 利用式(19)－式(23)更新所有粒子的速度。
10. 利用式(25)－式(29)更新所有粒子的位置。
11. 利用式(17)计算所有粒子的适应度值。
12. 对于任意粒子，如果该粒子的当前适应度值大于其历史最优适应度值，则以当前粒子的速度和位置更新最佳粒子的速度和位置。
13. 找到全局最佳粒子。
14. 利用式(30)－式(34)更新全局最佳粒子的速度。
15. 利用式(35)－式(39)更新全局最佳粒子的位置。
16. 利用式(40)更新比例因子 β。
17. 令 t=t+1，并回到步骤 7。

5 仿真分析

假定 $\vartheta^{\max}=0.9$，$\vartheta^{\min}=0.4$，$w_1=2$，$w_2=2$ 及 $\xi=10^{-25}$；对于任意 IMD_i，设置 $\omega_i=\bar{\omega}=0.5$；对于任意 SBS_j，设置 $\varepsilon_0=\varepsilon_j=1(\mathrm{W/GHz})$[19]；$n_1=15$，$n_2=5$[20]；$\mathrm{IMD}_i$ 和 MBS 之间的路径损耗为 $128.1+37.6\log_{10}(\ell_{ij})$，$\mathrm{IMD}_i$ 和 SBS_j 之间的路径损耗为 $140.7+36.7\log_{10}(\ell_{ij})$，其中 ℓ_{ij} 表示基站 j 与 IMD_i 之间的距离(单位为 km)。其他一些重要参数如表 1 所列。

表 1 仿真参数

Table 1 Simulation parameters

参数	值
系统带宽/MHz	20
噪声功率 σ^2/mW	10^{-11}
基站 j 的 F_j^{BS}/GHz	20
IMD_i 的 F_i^{UE}/GHz	1
IMD_i 的任务 k 的 d_{ik}/kB	200～500
计算 1 比特 d_{ik} 所需的 c_{ik}/(cycles/bit)	50～100
对数正态阴影衰落/dB	8 标准偏差
有线回程链路速率 r_0/Gbps	1
有线功耗 ζ/(mW/s)	1

为了凸显本文算法(Adaptive PSO)的有效性，本文引入传统 PSO 算法进行比较。在仿真中，研究了 IMD 密度 $\bar{\rho}$、IMD 最大传输功率 $\bar{p}$ 与 IMD 权重对效用函数 Γ(优化问题的目标函数)的影响，并研究了 IMD 权重对总时延与总能耗的影响，其中 IMD 密度指单个宏小区内的 IMD 数量。此外，不失一般性，对于任意的 IMD_i，设置 $p_i^{\max}=\bar{p}$。

图 2 给出了 IMD 密度 $\bar{\rho}$ 对效用函数 Γ 的影响，其中单个宏小区内 SBS 的数量为 35，$\omega_i=0.5$，$\bar{p}=23$ dBm。如图 2 所示，随着 IMD 密度的增加，效用函数的值也随之增加。这是因为，随着 IMD 的增多，总能耗和总时延也会随之升高。此外，容易发现，Adaptive PSO 具有较传统 PSO 更小的效用函数值。其原因在于，前者通过更新惯性权重使算法不易陷入局部最优解，并二次优化全局最优粒子的解，从而最终搜索到更好的解，达到了更小的开销。

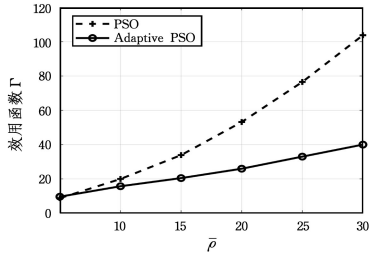

图 2 $\bar{\rho}$ 对效用函数 Γ 的影响

Fig. 2 Impacts of $\bar{\rho}$ on utility function Γ

图 3 给出了 IMD 的最大传输功率 $\bar{p}$ 对效用函数 Γ 的影响，其中单个宏小区内 SBS 的数量为 30，$\omega_i=0.5$，$\bar{\rho}=30$。如图 3 所示，效用函数的值随着 $\bar{p}$ 的增加而下降。其原因在于，随着 $\bar{p}$ 的增大，IMD 的上传速率提高，传输时延则随之下降。在效用函数中，当 $\bar{p}$ 对时延的影响大于其对能耗的影响时，时延的下降可能会导致效用函数值的下降。此外，类似于图 2，Adaptive PSO 具有较传统 PSO 更小的效用函数值。

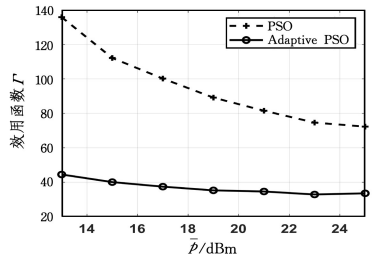

图 3 $\bar{p}$ 对效用函数 Γ 的影响

Fig. 3 Impacts of $\bar{p}$ on utility function Γ

为了更好地解释 IMD 权重 $\bar{\omega}$ 对效用函数 Γ 的影响，首先调查了其对总时延和总能耗的影响。

图 4 给出了 IMD 权重 $\bar{\omega}$ 对总时延的影响，其中单个宏小区内 SBS 的数量为 30，$\bar{p}=23$ dBm，$\bar{\rho}=30$。如图 4 所示，总时延随着 $\bar{\omega}$ 的增大而下降。究其原因，从优化问题(16)的目标函数可以看出，当 $\bar{\omega}$ 逐渐增大时，该优化问题逐步倾向于优化时延。特别地，当 $\bar{\omega}=1$ 时，问题(16)变成了总时延最小化问题。此外，由图 4 可知，Adaptive PSO 算法获得了比 PSO 算法更低的总时延。

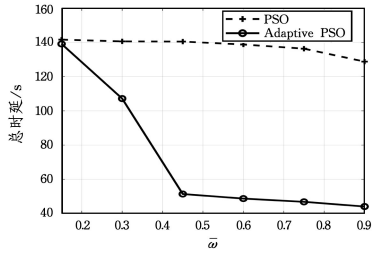

图 4 IMD 权重 $\bar{\omega}$ 对总时延的影响

Fig. 4 Impacts of $\bar{\omega}$ on total delay

图 5 给出了 IMD 权重 $\bar{\omega}$ 对总能耗的影响，其中单个宏小区内 SBS 的数量为 30，$\bar{p}=23$ dBm，$\bar{\rho}=30$。如图 5 所示，总能耗随着 $\bar{\omega}$ 的下降而下降。根据优化问题(16)的目标函数，很容易发现，当 $\bar{\omega}$ 逐渐下降时，该优化问题逐步倾向于优化能耗。特别地，当 $\bar{\omega}=0$ 时，问题(16)变成了总能耗最小化问题。此外，由图 5 可知，Adaptive PSO 算法获得了比 PSO 算法更高的总能耗。

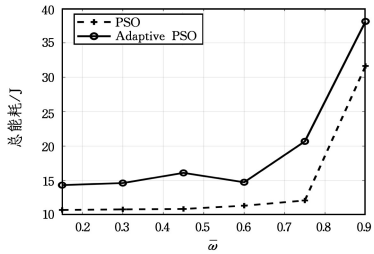

图 5 IMD 权重 $\bar{\omega}$ 对总能耗的影响

Fig. 5 Impacts of $\bar{\omega}$ on total energy consumption

图 6 给出了 IMD 权重 $\bar{\omega}$ 对效用函数 Γ 的影响，其中单个宏小区内 SBS 的数量为 30，$\bar{p}=23$dBm，$\bar{\rho}=30$。如图 6 所示，总体上，效用函数值随着 $\bar{\omega}$ 的增大而减小。显然，这一趋势取决于 $\bar{\omega}$ 对能耗与时延的影响程度。此外，类似于图 2，Adaptive PSO 具有比传统 PSO 更小的效用函数值。

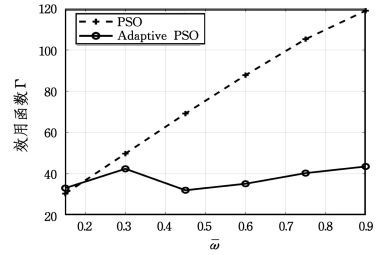

图 6 IMD 权重 $\bar{\omega}$ 对效用函数 Γ 的影响

Fig. 6 Impacts of $\bar{\omega}$ on utility function Γ

图 7 给出了 Adaptive PSO 算法和 PSO 算法的收敛

情况。如图 7 所示，PSO 算法和 Adaptive PSO 算法均能快速收敛，但 PSO 算法容易陷入局部最优解，而 Adaptive PSO 算法却可以找到更好的解。

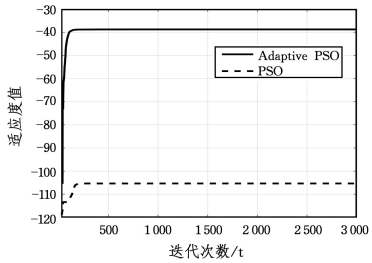

图 7 Adaptive PSO 算法和 PSO 算法的收敛性

Fig. 7 Convergence of adaptive PSO and PSO

结束语 针对超密集物联网络中的多任务多用户移动边缘计算模型，在等带宽分配和比例计算资源分配下执行了多步计算卸载。通过最小化能耗与时延的加权和，联合优化了用户关联、功率控制、卸载数据量与频带划分因子。然后，采用 Adaptive PSO 算法来解决所规划的问题。尽管本文利用等带宽分配来消除各类网络的干扰，但导致了较低的频谱利用率。针对上述问题，未来可以结合 NOMA 技术开展研究。

参 考 文 献

[1] SNYDER T, BYRD G. The internet of everything [J]. Computer, 2017, 50(6): 8-9.

[2] ADEDOYIN M A, FALOWO O E. Combination of ultra-dense networks and other 5G enabling technologies: a survey[J]. IEEE Access, 2020, 8: 22893-22932.

[3] TRAN T X, HAJISAMI A, PANDEY P, et al. Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges [J]. IEEE Communications Magazine, 2017, 55(4): 54-61.

[4] NDIKUMANA A, TRAN N H, HO T M, et al. Joint communication, computation, caching, and control in big data multi-access edge computing[J]. IEEE Transactions on Mobile Computing, 2020, 19(6): 1359-1374.

[5] KHAN W Z, AHMED E, HAKAK S, et al. Edge computing: a survey[J]. Future Generation Computer Systems, 2019, 97: 219-235.

[6] ZHU Y, HU Y, SCHMEINK A. Delay minimization offloading for interdependent tasks in energy-aware cooperative MEC networks[C]// 2019 IEEE Wireless Communications and Networking Conference. Marrakesh: IEEE, 2019: 1-6.

[7] ZHANG K, LENG S, HE Y, et al. Mobile edge computing and networking for green and low-latency internet of things[J]. IEEE Communications Magazine, 2018, 56(5): 39-45.

[8] DAB B, AITSAADI N, LANGAR R. A novel joint offloading and resource allocation scheme for mobile edge computing[C]// 2019 16th IEEE Annual Consumer Communications & Networking Conference. Piscataway: IEEE, 2019: 1-2.

[9] XU C, LEI J, LI W, et al. Efficient multi-user computation offloading for mobile-edge cloud computing[J]. IEEE/ACM Transactions on Networking, 2016, 24(5): 2795-2808.

[10] ZHANG H B, ZHANG Y F, LIU K J. Task offloading, migration and caching strategy in internet of vehicles based on NOMA-MEC[J]. Computer Science, 2022, 49(2): 304-311.

[11] KAN T, CHIANG Y, WEI H. Task offloading and resource allocation in mobile-edge computing system[C]// 2018 27th Wireless and Optical Communication Conference. Hualien: IEEE, 2018: 1-4.

[12] ZHANG J, XIA W, YAN F, et al. Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing [J]. IEEE Access, 2018, 6: 19324-19337.

[13] YANG L, ZHANG H, LI M, et al. Mobile edge computing empowered energy efficient task offloading in 5G[J]. IEEE Transactions on Vehicular Technology, 2018, 67(7): 6398-6409.

[14] DING Z, NG D, SCHOBRE R, et al. Delay minimization for NOMA-MEC offloading[J]. IEEE Signal Processing Letters, 2018, 25(12): 1875-1879.

[15] ZHOU T Q, YUE Y L, QIN D, et al. Joint device association, resource allocation and computation offloading in ultra-dense multi-device and multi-task IoT networks [J]. arXiv: 2112.05891, 2021.

[16] ZHOU T Q, QIN D, NIE X F, et al. Energy-efficient computation offloading and resource management in ultradense heterogeneous networks [J]. IEEE Transactions on Vehicular Technology, 2021, 70(12): 13101-13114.

[17] SHI Y, EBERHART R. A modified particle swarm optimizer [C] // 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence. Anchorage: IEEE, 1998: 69-73.

[18] DEN VAN BERGH F. An analysis of particle swarm optimizers [D]. South Africa: University of Pretoria, 2007.

[19] DAI 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.

[20] DEN VAN BERGH F, ENGELBRECHT A P. A new locally convergent particle swarm optimizer[C] // IEEE International Conference on Systems, Man and Cybernetics. Yasmine Hammamet: IEEE, 2002: 6-9.

ZHOU Tian-qing, born in 1983, Ph. D, associate professor, master supervisor, is a member of China Computer Federation. His main research interests include the resource management in ultra-dense networks and mobile edge computing networks.

（责任编辑：喻藜）