Journal of Computer Applications
计算机应用,2020,40(8):2293－2298
ISSN 1001-9081
CODEN JYIIDU
2020-08-10
http://www.joca.cn

文章编号:1001-9081(2020)08-2293-06
DOI:10.11772/j.issn.1001-9081.2019122200

移动边缘计算中基于粒子群优化的计算卸载策略

罗　斌[1,2*],于　波[1,2]
(1.中国科学院大学,北京 100049;　2.中国科学院 沈阳计算技术研究所,沈阳 110168)
(*通信作者电子邮箱1280371035@qq.com)

摘　要:计算卸载作为移动边缘计算(MEC)中降低时延与能耗的手段之一,通过合理的卸载决策能够降低工业成本。针对工业生产线中部署MEC服务器后时延变长和能耗增高的问题,提出了一种基于粒子群优化(PSO)算法的计算卸载策略PSAO。首先,将实际问题建模为时延模型与能耗模型。由于是针对时延敏感型的应用,因此将模型转化为在能耗约束条件下的最小化时延问题,使用惩罚函数来平衡时延与能耗。其次,根据PSO算法优化后得到计算卸载决策向量,通过集中控制的方式使每一个计算任务合理分配到对应的MEC服务器。最后,通过仿真实验,对比分析了本地卸载策略、MEC基准卸载策略、基于人工鱼群算法(AFSA)的卸载策略以及PSAO的时延数据,PSAO的平均总时延远远低于其他三种卸载策略,PSAO比原来系统总代价降低了20%。实验结果表明,PSAO策略能够降低MEC中的时延,均衡MEC服务器的负载。

关键词:移动边缘计算;计算卸载;增强现实;粒子群优化算法;工业生产线
中图分类号:TP391.9　　**文献标志码:**A

Computation offloading strategy based on particle swarm optimization in mobile edge computing

LUO Bin[1,2*], YU Bo[1,2]
(1. *University of Chinese Academy of Sciences*, *Beijing* 100049, *China*;
2. *Shenyang Institute of Computing Technology*, *Chinese Academy of Sciences*, *Shenyang Liaoning* 110168, *China*)

Abstract: Computation offloading is one of the means to reduce delay and save energy in Mobile Edge Computing (MEC). Through reasonable offloading decisions, industrial costs can be greatly reduced. Aiming at the problems of long delay and high energy consumption after the deployment of MEC servers in the industrial production line, a computation offloading strategy based on Particle Swarm Optimization (PSO) was proposed, namely PSAO. First, the actual problem was modeled to a delay model and an energy consumption model. Since it was targeted at delay-sensitive applications, the model was transformed into a delay minimization problem under the constraints of energy consumption, and a penalty function was used to balance delay and energy consumption. Second, according to the PSO, the computation offloading decision vector was obtained, and each computation task was reasonably allocated to the corresponding MEC server through the centralized control method. Finally, through simulation experiments, the delay data of local offloading strategy, MEC baseline offloading strategy, Artificial Fish Swarm Algorithm (AFSA) based offloading strategy and PSAO were compared and analyzed. The average total delay of PSAO was much lower than those of the other three offloading strategies, and PSAO reduces the total cost of the original system by 20%. Experimental results show that the proposed strategy can effectively reduce the delay in MEC and balance the loads of MEC servers.

Key words: Mobile Edge Computing (MEC); computation offloading; Augmented Reality (AR); Particle Swarm Optimization (PSO) algorithm; industrial production line

0　引言

5G具有近邻性、低延迟、高带宽等特性,作为关键技术的移动边缘计算(Mobile Edge Computing, MEC)也逐渐成熟。MEC将无线网络和互联网技术融合在一起,工业作为5G的一个应用方向,可以将MEC应用于现场设备实时控制、远程维护及操控、工业图像处理等方面[1]。任务计算量的提升导致本地设备自身无法处理相应的计算任务,而云计算的提出则很好地解决了算力不足的问题;但是云计算也带来了数据传输成本、云存储花费、互联网访问管理和安全性等一系列问题[2]。MEC将端云架构转变为端边云架构,在一定程度上降低了时延,并通过合理的计算卸载策略提高了计算任务的吞吐量,能更好地满足用户体验质量需求,使经济利益最大化。计算卸载作为MEC的研究方向之一,自2014年提出以来,工业界与学术界已经对其进行了大量的研究。目前的卸载策略控制方式主要是集中式控制与分布式协作控制,在建模场景方面主要包括单用户单MEC、多用户单MEC和多用户多MEC,优化目标主要包括能耗约束条件下最小化时延、时延约

收稿日期:2019-12-31;**修回日期:**2020-03-29;**录用日期:**2020-04-07。
作者简介:罗斌(1995—),男,甘肃兰州人,硕士研究生,主要研究方向:计算机网络、机器学习;　于波(1980—),男,辽宁沈阳人,博士,主要研究方向:计算机网络、IP通信。

束下最小化能耗以及平衡能耗与时延的卸载策略。目前研究的问题大部分是对于通用场景下的问题建模，针对工业场景下计算卸载策略的研究比较少。考虑工业生产线中实际的卸载条件，工业场景相较于通用场景下的计算卸载策略存在以下不同：1）工业场景下接入MEC服务器的设备数目多，需要考虑一种总体最优的策略；2）工业场景下对于时延特性的要求会更加严格，主要关注如何最小化时延；3）工业场景下MEC服务器会处理大量特定（图像识别、设备控制等）的任务，需要选择合适的计算模型以及通信方式。

本文主要考虑了工业生产线中的计算卸载策略，在多用户多MEC场景下，研究在能耗约束条件下时延最小化的问题，通过集中控制卸载高计算量的任务到MEC服务器。工业生产线中可以降低传输能耗与计算能耗，选择合适的通信方式以及数据压缩来降低传输能耗[3]；通过提高MEC服务器的处理性能来降低计算能耗。当前卸载策略应用场景是通过虚拟现实（Virtual Reality，VR）眼镜来实时识别和控制工业生产线中的设备，对于时延要求较高，因此选择最小化时延的卸载策略。具体的工作如下：

1）为工业场景构造时延模型与能耗模型，在MEC系统中将计算卸载问题转化为时延最小化时延模型，其中能耗模型作为约束条件；

2）提出通过智能优化算法中的粒子群优化（Particle Swarm Optimization，PSO）算法来解决最小化问题，对于离散粒子群算法进行了一定修改来适应问题；

3）通过仿真实验表明完全本地卸载策略与MEC基准卸载策略都存在各自的优缺点，而本文提出的基于PSO算法的计算卸载策略PSAO能够在能耗与时延上达到相对平衡，基于人工鱼群算法（Artificial Fish Swarm Algorithm，AFSA）的卸载方案前期优化效果与PSAO相差不大，但当设备数过多或者任务量过大时，鱼群卸载策略远不如PSAO卸载策略。

1 相关工作

欧洲电信标准协会（European Telecommunications Standards Institute，ETSI）提出了移动边缘计算（MEC）的概念，该概念将边缘计算集成到移动网络架构中[4]。2016年在原有的网络基础上接入了Wi-Fi，将MEC的中文解释修改为“多接入边缘计算”。计算卸载是MEC中重要的研究方向，主要包含卸载决策和资源分配，本文主要关注计算卸载决策，也就是研究用户中要不要卸载、卸载什么和卸载到哪里的问题[5]。文献[6]中提出了动态卸载方案，将卸载任务看作是两级随机优化问题，通过任务排队模型，由移动设备来决定任务在本地执行还是在MEC服务器上执行，采用马尔可夫决策过程来处理该问题，提出一种有效的一位搜索算法来解决功率受限的时延问题。文献[7]中以时延为优化目标，在文献[6]的基础上提出了自带能量收集的MEC系统，选择在线计算卸载策略，卸载过程中考虑了数据在信道中传递与CPU上计算的各个因素，使用Lyapunov优化来得到最优解。文献[6-7]虽然对于单个任务卸载到MEC问题给出了解决方案，但是在工业生产线中往往有成百上千个移动设备和传感器需卸载任务，此时需要分布式的策略来进行卸载。文献[8]中从另一方面来考虑卸载任务，主要是针对一系列具有顺序关系的任务，通过拓扑图的优化方式来卸载，使用负载平衡试探法将任务卸载到MEC中，最大限度提高移动设备与云之间的并行性。其他研究中也提出了通过匈牙利算法来解决一些非顺序的任务，一般是需要本地设备进行初步处理，MEC完成后续高计算量的任务[9]，在移动设备与云之间的并行方面也提供了新的思路。文献[10-11]中将多用户多MEC场景建模为时延约束条件下的能量优化问题，通过AFSA来求解决策向量。文献[12]中基于云计算提出了一种快速混合多站点计算卸载解决方案，可根据移动应用程序的大小实现最佳和接近最优的卸载分区，当规模扩大时提出使用粒子群优化（PSO）的优化版本来获得非常合适的卸载优化解决方案。文献[13]中结合人工智能技术，基于长短期记忆（Long Short-Term Memory，LSTM）网络预测计算任务，提出基于任务预测的移动设备计算卸载策略。目前各个技术趋向于相互融合来达到更优的结果，通过神经网络或者机器学习来辅助优化边缘计算卸载模型。文献[14]中对近几年关于计算卸载策略的研究做了汇总，主要根据计算卸载的卸载类型、优化目标提出解决方案、场景、评估方式进行分类总结。MEC未来可能的应用方向主要集中在动态内容优化、物联网、移动大数据分析和智能交通[15]。

结合以上的研究不难发现，虽然上述研究提出的系统或者算法在能耗和时延优化方面都有着较优的性能，但是都很难符合工业生产线场景下的计算卸载低时延目标，其中文献[12]中提出的快速混合多站点计算卸载能够适应工业场景，但是由于云计算带来的成本问题以及传输时延增高等一系列问题，需要提出一种符合多用户多MEC场景下，在能耗约束的条件下时延最小化的完全卸载策略。

2 系统模型

MEC系统的部署场景如图1所示，MEC服务器主要面向时延敏感型和计算量巨大的任务，核心任务包含工业设备的控制与工业设备的识别。本地设备（移动设备、传感器等）主要执行数据收集或者是前期数据处理，在脱离MEC服务器的情况下只能进行简单任务的处理，需要将计算任务传输到对应的MEC服务器进行计算。MEC服务器部署的位置比较灵活，本文选择将MEC计算服务器部署于微蜂窝基站上，在微蜂窝基站上实现具体的计算功能，本地设备通过无线的方式与服务器之间进行交互。针对工业生产线中的计算卸载问题：首先，在工业生产线中将会接入各种类型的移动设备（全景摄像头、智能手机、AR眼镜等），移动设备也具有一定的计算能力，即“多用户”；其次，在工业生产线中往往需要多个MEC服务器来处理任务，即“多MEC”。PSAO卸载策略应用于AR眼镜与工业设备的实时交互，需要保证低延迟。

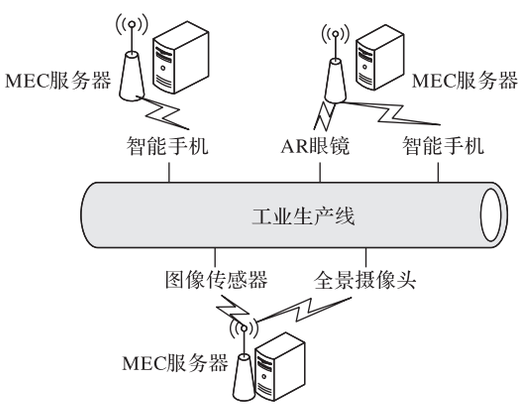

图1 多设备多MEC系统模型

Fig. 1 Model of multi-device multi-MEC system

假定在当前网络中包括K个移动设备，N个MEC服务器。每一个移动设备产生任务后指定一个服务器（或本地）来执

行,最后找到所有任务最优的执行位置,即计算卸载向量$\boldsymbol{V}$,卸载结果会有$N+1$种可能性,任务可能会在本地执行或者将任务装载到N个MEC服务器中的其中一个,并由装载服务器执行。

2.1　时延模型

根据任务在不同设备上运行,时延模型也会存在差别:如果当前任务在本地设备执行并且每个任务都没有排队延迟,此时只需要考虑本地计算时延;若将任务卸载到MEC服务器执行,完成当前任务所需要的时延主要包括传输时延、MEC计算时延、反馈时延,其中反馈时延远远小于传输时延,因此在本文中将忽略反馈时延。表1详细介绍了系统建模中使用的符号含义和在仿真实验中的取值,其中:下标i表示当前当前移动设备编号,j表示当前MEC服务器编号,u表示移动设备标识,s表示服务器标识。

表1　系统建模中使用的符号含义

Tab. 1　Meaning of symbols used in system modeling

参数	参数符号含义
B_i	用来处理任务的数据量
f_i	每比特数据量需要多少个时钟周期来处理
$f_{\mathrm{u},i}$	本地设备的CPU周期频率
$f_{\mathrm{s},i}$	MEC服务器CPU时钟频率
W	传输带宽
P_i	每一个本地设备的发射功率
$H_{i,j}$	信道增益
N_0	噪声功率频谱密度
$r_{i,j}$	本地设备i到第j个MEC的传输速率
$E_{\max}$	最大能耗约束
k_{u}	本地设备CPU固有系数
k_{s}	MEC服务器CPU固有系数
g	惩罚因子
T_i	表示每一项任务所需要的总时延
E_i	第i个任务的传输能耗

1)本地计算时延。

$$T_{\mathrm{local}} = B_i * f_i / f_{\mathrm{u},i} \tag{1}$$

式(1)表示本地计算时延与本地CPU周期频率的倒数成正比,$B_i * f_i$表示当前任务计算量。

2)MEC传输时延。

在考虑计算MEC计算时延之前,需要去定义通道传输速率,根据香农定义可知:

$$r_{i,j} = W * \mathrm{lb}\left(1 + \frac{P_i * H_{i,j}}{W * N_0}\right) \tag{2}$$

已知当前任务在通道中的传输速率为式(2),当前任务计算量为$B_i * f_i$,则传输时延为:

$$T_{t,i,j} = B_i * f_i / r_{i,j} \tag{3}$$

3)MEC计算时延。

$$T_{\mathrm{s},i} = B_i * f_i / f_{\mathrm{s},i} \tag{4}$$

式(4)表示MEC计算时延与MEC服务器周期频率的倒数成正比。

2.2　能耗模型

1)计算能耗。

CPU的算力与芯片架构有着密切的关系,每一代CPU架构都有固定的能耗比,在同架构前提下,功耗与电压和频率都成正比[16],设备之间的能耗也会受到计算任务量的影响:

$$E_i = C_i * L^2 * f_{\mathrm{u},i} * B_i * f_i \tag{5}$$

其中:电容C_i取决于有效开关电容;L表示电压。

2)传输能耗。

传输能耗针对的是卸载到MEC服务器上的任务,在时延模型中已经计算出传输时延,则传输能耗为:

$$E_{t,i,j} = P_i * T_{t,i,j} \tag{6}$$

2.3　计算模型

2.1节和2.2节对工业生产线问题抽象建模:首先当前MEC的垂直应用场景主要是针对计算密集型和延迟敏感型计算任务,其中包含工业中的增强现实图像识别;其次需要确保每一项本地设备的平均能耗都小于最大能耗,因此实际建模为在能耗约束条件下的最小化时延问题。

完成一个卸载任务的总时延为传输时延与服务器计算时延的和:

$$T_i = T_{\mathrm{t},i,j} + T_{\mathrm{s},i} \tag{7}$$

本地执行一个任务的总时延为:

$$T_i = T_{\mathrm{local}} \tag{8}$$

由式(7)与式(8),可以得到针对某一个计算任务,在本地设备计算与MEC服务器计算时会有不同的时延求解公式,可以将当前问题联合表达为:

$$\mathrm{P1}:\ \min_{v_i} \sum_{j=1}^{n}\sum_{i=1}^{k} T_i \tag{9}$$

$$\text{s. t.}\quad E_i < E_{\max} \tag{10}$$

$$0 \leqslant P_i \leqslant P_{\max} \tag{11}$$

$$0 \leqslant f_{\mathrm{u},i} \leqslant f_{\mathrm{u},\max} \tag{12}$$

$$0 \leqslant f_{\mathrm{s},i} \leqslant f_{\mathrm{s},\max} \tag{13}$$

其中:式(9)表示找到使得满足时延最小的卸载决策向量,v_i表示任务i具体分配到的服务器编号;式(10)表示能耗约束,每个任务所耗费的平均能耗小于最大能耗;式(11)表示本地设备的CPU周期频率小于最大CPU周期频率;式(12)表示MEC服务器的CPU周期频率小于最大CPU周期频率。P1更加关注在整个卸载过程中的时延问题。

再考虑这样一种情况:其中一个MEC服务器性能更优,按照式(9)的建模方式,因为没有考虑到排队时延,导致大多数任务会卸载到性能更优的MEC服务器,性能优秀的MEC服务器的平均功耗会高于最大能耗,如果能将当前任务卸载到其他MEC服务器则可以缓解负载压力。因此选择将P1问题转化为增加惩罚函数的P2问题:当设备平均能耗大于最大能耗时,通过增加惩罚函数的形式,来使得任务卸载到当前MEC服务器的系统总成本增高。

$$\mathrm{P2}:\ \min_{v_i} \sum_{j=1}^{n}\sum_{i=1}^{k} T_i + penalty(\boldsymbol{V}) \tag{14}$$

其中:

$$penalty(\boldsymbol{V}) = g * \sum_{j=1}^{n}\sum_{i=1}^{k} (E_i - E_{\max}) \tag{15}$$

$$\text{s. t.}\quad 0 \leqslant P_i \leqslant P_{\max}$$

$$0 \leqslant f_{\mathrm{u},i} \leqslant f_{\mathrm{u},\max}$$

$$0 \leqslant f_{\mathrm{s},i} \leqslant f_{\mathrm{s},\max}$$

其中:$\boldsymbol{V}$表示需要求解的卸载向量;g表示惩罚系数,随着计算消耗的能耗越多,则惩罚项越大,如果不超过最大能耗则不惩罚,增加惩罚函数的意义在于平衡能耗与时延。P2更加关注在整个卸载过程中的时延与能耗的平衡。

3　基于PSO算法的优化

PSO算法受鸟类捕食行为的启发并对这种行为进行模仿,将优化问题的搜索空间类比为鸟类的飞行空间,将每一只

鸟抽象为一个粒子，粒子无质量无体积，用以表征问题的一个可行解，优化问题所要搜索到的最优解等同于鸟类寻找的食物源[17]。

3.1 粒子编码

假设粒子群规模为 U，K 表示当前本地任务数量，N 表示 MEC 服务器的数量，所有任务的集合可以表示为 $M=\{m_1, m_2, \cdots, m_k\}$。首先对任务进行定量描述，假设编号为 i 的移动设备产生 $m_i(B_i, f_i, E_{\max})$ 个任务待计算。其中：B_i 表示输入数据量，f_i 表示计算密度，$E_{\max}$ 表示任务的能耗约束。所有 MEC 服务器周期频率集合 $S=\{s_1, s_2, \cdots, s_n\}$，包含了当前系统中所有服务器的 CPU 周期频率。信道增益矩阵 $\boldsymbol{H}=\begin{bmatrix} H_{1,1} & H_{1,2} & \cdots & H_{1,n} \\ H_{2,1} & H_{2,2} & \cdots & H_{2,n} \\ \vdots & \vdots & & \vdots \\ H_{K,1} & H_{K,2} & \cdots & H_{K,n} \end{bmatrix}$ 用来计算任务传输到 MEC 服务器的最大传输速率，$H_{i,i}$ 的信道增益为 0，$H_{i,j}(1 \leqslant i \leqslant K, 1 \leqslant j \leqslant N, i \neq j)$ 表示移动设备 i 产生的任务传输到编号 j 服务器所需要的信道增益。

粒子个体采用整数编码，每一个粒子的元素可以取 1 到 N 之间的任意整数，粒子编码的维度与任务集合的个数相同。假定当前任务集合包含 5 个任务 $M=\{m_1, m_2, m_3, m_4, m_5\}$，粒子编码为[0, 1, 2, 5, 1, 2]表示所有任务的执行位置，其中第一个元素 0 表示 m_1 任务直接在本地进行计算，第二个元素 1 表示 m_2 任务将要装载到编号为 1 的 MEC 服务器执行，以此类推。

粒子表示当前所有任务将要卸载到的具体某个 MEC 服务器。每一个粒子的卸载决策向量 $\boldsymbol{V}=\{v_1, v_2, \cdots, v_k\}$ 表示所有任务最优的执行位置，其中 v_i 在 0 到 N 中随机取值：$v_i=0$ 表示当前任务直接在本地执行；$v_i=j(1 \leqslant j \leqslant N)$ 表示当前任务装载到第 j 个 MEC 服务器，并由第 j 个服务器执行。

粒子的速度表示当前任务分配到其他 MEC 服务器的趋势快慢，记作 $\boldsymbol{X}_i=\{x_1, x_2, \cdots, x_k\}$，粒子速度随机初始化为整数值并在更新过程中进行取整，粒子速度的编码维度与任务集合的数目相同。假定当前任务集合包含 5 个任务 $M=\{m_1, m_2, m_3, m_4, m_5\}$，粒子速度为[0, 3, 1, 4, 1, 2]，其中第一个元素 0 表示 m_1 任务依旧由原来的 MEC 服务器处理，第二个元素 3 表示 m_2 任务由原来下移 3 位的 MEC 服务器处理，以此类推。

每一个粒子在所有迭代过程中的最优位置记作 $\boldsymbol{P}_{\text{best}}=\{p_1, p_2, \cdots, p_n\}$，表示当前粒子找到的使得系统总代价最小的任务分配方式；记录所有粒子中出现的最优位置，记作 $\boldsymbol{G}_{\text{best}}=\{g_1, g_2, \cdots, g_n\}$，表示所有粒子中系统代价最小的分配方式。

3.2 适应度函数

粒子的适应度表示所有任务分配到不同 MEC 服务器的系统总代价，计算公式为：

$$Fitness(\boldsymbol{V})=\sum_{j=1}^{n}\sum_{i=1}^{k}T_i+penalty(\boldsymbol{V})=\sum_{j=1}^{n}\sum_{i=1}^{k}T_i+g*\sum_{j=1}^{n}\sum_{i=1}^{k}(E_i-E_{\max}) \tag{16}$$

3.3 计算卸载流程

粒子群速度更新的决定因素主要包含三部分：第一部分主要是自身原来的速度，也可以称作“惯性影响”；第二部分主要是自身记忆的最优位置，也可以称作“认知影响”；第三部分主要是全部粒子记忆的最优位置，也可以称作“社会影响”。最后用粒子群算法解决计算卸载问题。

算法 1 基于 PSO 算法的卸载流程。

输入：

1）本地任务集合 $M=\{m_1, m_2, \cdots, m_k\}$，MEC 服务器集合 $S=\{s_1, s_2, \cdots, s_n\}$，信道增益矩阵 $\boldsymbol{H}$。

2）算法控制参数：粒子群规模 $N=25$，迭代代数 $maxGen=100$，速度边界 $v_{\max}$ 为 MEC 服务器个数的 2 倍，位置边界 $p_{\max}$ 为 MEC 服务器的个数，学习因子 $c_1=1.5$，$c_2=1.5$ 惯性因子 $W_{\text{p}}=0.4$，惩罚因子 $g=1\times10^{-2.5}$。

输出：

最优分配向量 $\boldsymbol{V}_i=\{v_1, v_2, \cdots, v_n\}$ 与最小延迟 $Fitness(\boldsymbol{V})$。

初始化：

1）初始化每一个粒子的随机位置 $\boldsymbol{V}_i$ 与速度 $\boldsymbol{X}_i$，其中 i 表示第 i 个粒子。

2）初始化适应度值：通过本机任务集合 M、MEC 服务期集合 S 和信道增益矩阵 $\boldsymbol{H}$ 计算时延与能耗，按式(16)来计算每一个粒子的适应度。

3）初始化粒子最优分配与全局最优分配：将粒子当前位置设置为个体最优任务分配方案 $\boldsymbol{P}_{\text{best}}$，并将适应度值最小的粒子的位置设置为群体最优分配方案 $\boldsymbol{G}_{\text{best}}$。

迭代计算：

4）令迭代次数 $t=0$

5）while $t \leqslant maxGen$

a）更新速度。粒子中的每一个维度独立去更新速度 $\boldsymbol{X}[i]$，如果速度大于 $v_{\max}$，则令 $\boldsymbol{X}[i]=v_{\max}$。粒子速度的更新公式为：

$$\boldsymbol{X}[i]=W_{\text{p}}*\boldsymbol{X}[i]+c_1*\text{rand}()*(\boldsymbol{P}_{\text{best}}-\boldsymbol{V}[i])+c_2*\text{rand}()*(\boldsymbol{G}_{\text{best}}-\boldsymbol{V}[i]) \tag{17}$$

其中：c_1 和 c_2 为学习因子；W_{p} 为惯性权重：rand()为 0~1 的随机数。

b）更新位置。粒子中的每一个维度独立去更新位置 $\boldsymbol{V}[i]$，如果 $\boldsymbol{V}[i]$ 位置大于 $p_{\max}$，则令 $\boldsymbol{V}[i]=p_{\max}$。粒子位置的更新公式为：

$$\boldsymbol{V}[i]=\boldsymbol{V}[i]+\boldsymbol{X}[i] \tag{18}$$

c）更新粒子最优分配与全局最优分配。通过本机任务集合 M、MEC 服务期集合 S 和信道增益矩阵 $\boldsymbol{H}$ 计算时延与能耗，按式(16)来计算每一个粒子的适应度。如果更新后的适应值小于当前适应值，则更新粒子最优分配方案 $\boldsymbol{P}_{\text{best}}$、群体最优分配方案 $\boldsymbol{G}_{\text{best}}$ 和系统总代价 $Fitness(\boldsymbol{V})$。

d）$t=t+1$

6）End

输出结果：

7）得到最佳分配向量 $\boldsymbol{V}[i]=\boldsymbol{G}_{\text{best}}$ 和最小延迟 $Fitness(\boldsymbol{V})$。

8）如果最优解不满足，则转到步骤 1）；否则输出最优分配向量 $\boldsymbol{V}_i=\{v_1, v_2, \cdots, v_n\}$ 与最小延迟 $Fitness(\boldsymbol{V})$。

最后通过集中控制的方式，$v_i(i=1, 2, \cdots, K)=0$ 将该任

务放入本地设备中执行,$v_i = j(j = 1, 2, \cdots, n)$将该任务放入对应编号$j$的MEC服务器中执行。

4　仿真与结果分析

4.1　仿真环境以及参数设置

通过Java语言实现了PSO算法,从而具体地评估所提出的计算卸载策略的性能。在工业生产线场景中,设备的个数K=[50,100,150,200,250],MEC服务器的个数为10,分别有K个移动设备向10个服务器发起卸载请求。仿真用到的主要参数取值与符号含义[10]如表2所示。

4.2　结果分析

根据表2的参数取值进行仿真实验,本文选择四种卸载策略进行对比:本地卸载策略、MEC基准卸载策略、基于人工鱼群算法(AFSA)的卸载策略和PSAO;同时也通过PSO算法优化基准卸载策略在卸载过程中的平均总时延。在实际生产环境中本地设备只能进行简单任务的处理,仿真实验中假设本地设备也能够处理高计算量的任务,假设每一项任务到达即执行,不会存在任务排队导致的时延增加。

表2　主要参数取值与符号含义

Tab. 2　Main parameter values and symbol meanings

参数	参数取值	参数	参数取值
B_i	7~40 Mb 随机选择	$H_{i,j}$	2×10^{-10}~2×10^{-6}
f_i	800~1 200 cycle/b 随机选择	$W*N_0$	1×10^{-9} W
$f_{u,i}$	1~3 GHz 随机选择	E_{max}	1 000 J
$f_{s,i}$	4~8 GHz 随机选择	k_u	10^{-24}
W	1 MHz	k_s	10^{-26}
P_i	100~500 mW	g	$10^{-2.5}$

图2为不同能耗约束下不同卸载策略的系统总代价对比,设置设备数K = 50,MEC服务器的个数N = 10,每一项任务数据量大小B_i = 3 Mb。可以观察到,随着能耗约束的增大,四种卸载策略系统总代价降低。当能耗(单位:J)约束为0时,本地卸载策略的系统总代价远远高于其他三种卸载策略,这是由于本地设备处理任务将会产生较高的能耗;能耗约束为750 J时,本地卸载策略代价开始低于MEC基准卸载策略,这是由于能耗约束的增大,模型对能耗约束的敏感度降低,因此需要选择合适的能耗约束。PSAO卸载策略在不同的能耗约束下都优于其他MEC基准卸载策略和本地卸载策略,当能耗约束大于750 J时,基于AFSA的卸载策略与PSAO卸载策略优化效果相差不大。

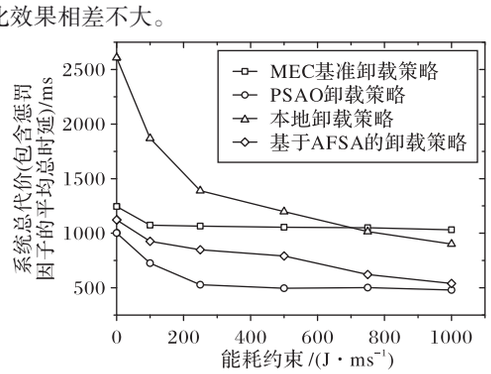

图2　能耗 vs. 系统总代价

Fig. 2　Energy consumption vs. total system cost

图3对PSAO卸载策略的收敛性进行评估,设置设备数K = 50,MEC服务器的个数N=10,每一项任务数据量大小B_i =3 Mb,通过改变迭代次数来观察所有任务的平均总时延。

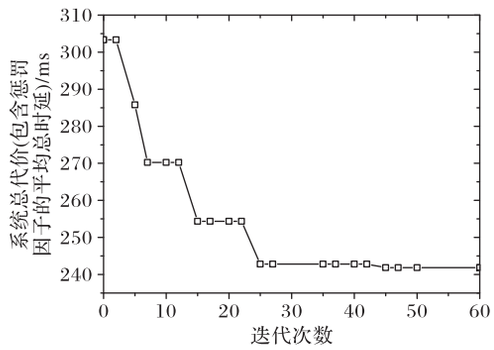

图3　收敛性分析

Fig. 3　Convergence analysis

由图3可以观察到,算法在前25次迭代过程中收敛较快,迭代50次后系统总代价保持不变,找到全局最优解。PSAO具有很强的全局优化能力,在算法的前期不断寻找全局的最优解,且在后期具有良好的全局搜索能力。PSO算法优化后平均总时延从原来的303.33 ms降低至241.824 ms,并在后期迭代过程中保持不变,相比原来系统的总时延降低了20%。

图4为不同设备个数的情况下,不同卸载策略的系统总代价对比。MEC服务器的个数N = 10,每一项任务数据量大小B_i = 3 Mb。可以观察到在设备数分别为50,100,150,200,250时,平均总时延是处于增长的趋势,这是由于随着设备数的增加,所需要的处理时间与传输时间也随之增加;但是,随着任务数目的增加,PSAO卸载策略总时延增长幅度远远小于MEC基准卸载策略与本地卸载策略。基于AFSA的卸载策略在设备数小于200时,优化程度与PSAO卸载策略相差不大;但当设备数目大于200时,系统总代价开始急速上升,远远高于PSAO卸载策略。

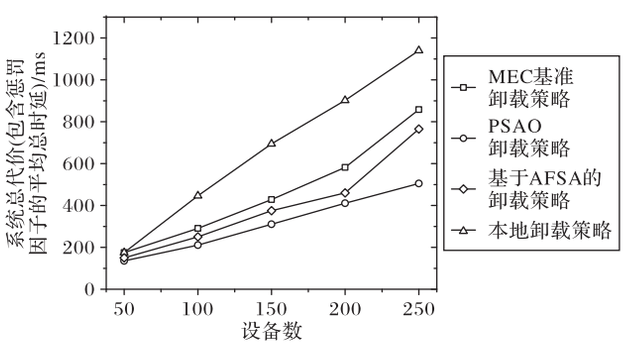

图4　设备数 vs. 系统总代价

Fig. 4　Number of devices vs. total system cost

图5为不同任务量的情况下,不同卸载策略的平均总时延对比。设置任务数K = 50,MEC服务器的个数N = 10,可以观察到:在任务量分别为3、10、15、25、40 Mb时,随着任务量的增加,平均总时延也逐渐增加。本地卸载策略比其他策略增长速度更快,表明随着任务量的增加会导致更多的时间与能量消耗。其中,基于AFSA的卸载策略在任务量小于15 Mb时,与PSAO卸载策略优化效果相差不大;但是在任务量大于15 Mb时,容易收敛于局部最优,并且在后期搜索过程

中盲目性比较大,卸载效果不理想。PSAO卸载策略增幅程度远远小于MEC基准卸载策略与本地卸载策略,并且在任务量过大时可以取得最佳效果。

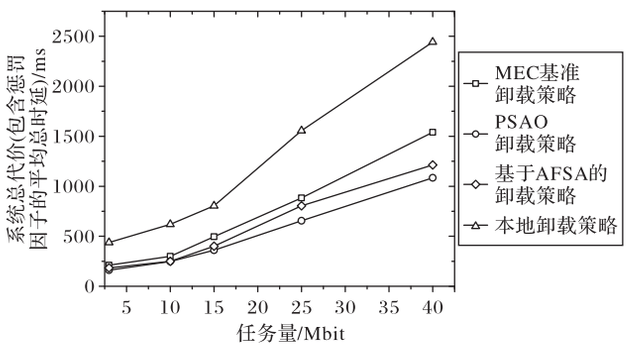

图5 任务量 vs. 系统总代价

Fig. 5 Task size vs. total system cost

5 结语

本文将工业生产线中的计算卸载问题建模为多用户多MEC时延优化问题,为了消除本地设备使用能耗来换取时延的问题,通过惩罚函数平衡时延与能耗,并提出基于PSO算法优化的卸载策略PSAO。基于分析结果可知,本地卸载策略与MEC基准卸载策略都有一定的局限性,本文选择PSAO卸载策略让本地设备和MEC服务器协作计算,使原本系统总代价降低了20%,并且随着设备数目和任务量的增加,它的增幅程度远小于本地卸载策略和MEC基准卸载策略,满足延迟敏感型应用的服务质量要求;基于AFSA的卸载策略虽然能够起到一定的优化作用,但在设备数增加和任务量增大的情况下,无法找到任务最佳分配方案。工业系统对于可靠性要求非常高,PSAO卸载策略属于启发式卸载策略,大部分启发式算法能够生成高质量的近似最优解,由于启发式方法的结果是随机产生的近似解,因此缺少可靠性[18]。未来将会考虑粒子群与其他优化算法相结合,提高寻找到最优解的鲁棒性,从而在降低时延的前提下,提高整个卸载策略的可靠性。

参考文献(References)

[1] 陆平,李建华,赵维铎. 5G在垂直行业中的应用[J]. 中兴通讯技术, 2019(1):67-74. (LU P, LI J H, ZHAO W D. Applications of 5G in vertical industry[J]. ZTE Technology Journal, 2019(1): 67-74.)

[2] GU Y, CHANG Z, PAN M, et al. Joint radio and computational resource allocation in IoT fog computing[J]. IEEE Transactions on Vehicular Technology, 2018, 67(8):7475-7484.

[3] 杨海波,马荣荣,张伟,等. 面向移动互联网的"SIP over MQTT"优化传输机制研究[J]. 小型微型计算机系统, 2017, 38(4): 776-780. (YANG H B, MA R R, ZHANG W, et al. Research of the optimization of "SIP over MQTT" transmission mechanism for mobile network[J]. Journal of Chinese Computer Systems, 2017, 38(4): 776-780.)

[4] HU Y C, PATEL M, SABELLA D, et al. ETSI White Paper 11: Mobile edge computing — a key technology towards 5G[R/OL]. [2019-12-15]. https://www. etsi. org/images/files/ETSIWhitePapers/etsi_wp11_mec_a_key_technology_towards_5g. pdf.

[5] 谢人超,廉晓飞,贾庆民,等. 移动边缘计算卸载技术综述[J]. 通信学报, 2018, 39(11): 138-155. (XIE R C, LIAN X F, JIA Q M, et al. Survey on computation offloading in mobile edge computing [J]. Journal on Communications, 2018, 39 (11): 138-155.)

[6] LIU J, MAO Y, ZHANG J, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]// Proceedings of the 2016 IEEE International Symposium on Information Theory. Piscataway: IEEE, 2016:1451-1455.

[7] MAO Y, ZHANG J, LETAIEF K B. Dynamic computation offloading for mobile-edge computing with energy harvesting devices [J]. IEEE Journal on Selected Areas in Communications, 2016, 34 (12): 3590-3605

[8] JIA M, CAO J, YANG L. Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing [C]// Proceedings of the 2014 IEEE Conference on Computer Communications Workshops. Piscataway: IEEE, 2014: 352-357.

[9] 张海波,李虎,陈善学,等. 超密集网络中基于移动边缘计算的任务卸载和资源优化[J]. 电子与信息学报, 2019, 41(5):1194-1201. (ZHANG H B, LI H, CHEN S X, et al. Computing offloading and resource optimization in ultra-dense networks with mobile edge computation[J]. Journal of Electronics and Information Technology, 2019, 41(5): 1194-1201.)

[10] 周晓敏. 面向节能的移动边缘计算的卸载策略研究[D]. 北京:北京邮电大学, 2019: 41-67. (ZHOU X M. Research on offloading strategy in energy-saving mobile edge computing system [D]. Beijing: Beijing University of Posts and Telecommunications, 2019: 41-67.)

[11] 郭俊. 超密集网络中基于移动边缘计算的卸载策略研究[D]. 北京:北京邮电大学, 2018: 19-45. (GUO J. Research of offloading scheme in ultra-dense networks integrated with MEC[D]. Beijing: Beijing University of Posts and Telecommunications, 2018: 19-45.)

[12] GOUDARZI M, ZAMANI M, HAGHIGHAT A T. A fast hybrid multi-site computation offloading for mobile cloud computing[J]. Journal of Network and Computer Applications, 2017, 80: 219-231.

[13] MIAO Y M, WU G X, LI M, et al. Intelligent task prediction and computation offloading based on mobile-edge cloud computing[J]. Future Generation Computer Systems, 2020, 102:925-931.

[14] MACH P, BECVAR Z. Mobile edge computing: a survey on architecture and computation offloading [J]. IEEE Communications Surveys and Tutorials, 2017, 19(3):1628-1656.

[15] AHMED A, AHMED E. A survey on mobile edge computing[C]// Proceedings of the 10th IEEE International Conference on Intelligent Systems and Control. Piscataway: IEEE, 2016:1-8.

[16] Wikipedia contributors. CPU power dissipation[EB/OL]. [2019-12-15]. https://en. wikipedia. org/w/index. php? title=CPU_power_dissipation&oldid=916130921.

[17] 包子阳,余继周,杨杉. 智能优化算法及其Matlab实例[M]. 2版. 北京:电子工业出版社,2018:112-134. (BAO Z Y, YU J Z, YANG S. Intelligent Optimization Algorithm and Its MATLAB Examples[M]. 2nd ed. Beijing: Publishing House of Electronics Industry, 2018: 112-134.)

[18] 祁晓峰,张兴明,高彦钊. 基于离散粒子群优化的可重构系统任务调度算法[J]. 小型微型计算机系统,2018,39(3):556-561. (QI X F, ZHANG X M, GAO Y Z. Discrete particle swarm optimization-based task scheduling algorithm in reconfigurable system[J]. Journal of Chinese Computer Systems, 2018, 39(3): 556-561.)

LUO Bin, born in 1995, M. S. candidate. His research interests include computer network, machine learning.

YU bo, born in 1980, Ph. D. His research interests include computer network, IP communications.