计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200700025

# 减少核心网拥塞的边缘计算资源分配和卸载决策

李振江　张幸林
华南理工大学计算机科学与工程学院　广州 510006
(zhenjiang_li37@163.com)

**摘　要**　随着移动互联网和物联网的发展,越来越多的智能终端设备投入到实际使用当中,大量计算密集型和时间敏感型应用被广泛应用,如 AR/VR 、智能家居、车联网等。因此,网络中的数据流量激增,使得核心网络面临的压力逐渐增大,对网络时延的控制也越来越难,此时云边协同的计算范式作为一种解决方案被提出。针对云边之间的核心网流量控制问题,文中提出了关于减少云边通信流量的资源分配和卸载决策算法。首先使用设计的基于分割时间槽的资源分配算法来提高边缘处理的流量,然后使用遗传算法搜索最优卸载决策。实验结果表明,与基线方案相比,所提算法能够更好地提高边缘的资源利用率,减少云边通信流量,从而减少潜在的核心网拥塞。

**关键词**:移动边缘计算;核心网拥塞;资源分配;计算卸载
**中图法分类号**　TP393

# Resource Allocation and Offloading Decision of Edge Computing for Reducing Core Network Congestion

LI Zhen-jiang and ZHANG Xing-lin
School of Computer Science & Engineering,South China University of Technology,Guangzhou 510006,China

**Abstract**　With the development of mobile Internet and IoT,more and more intelligent end devices are put into use,and a large number of computation-intensive and time-sensitive applications are widely used,such as AR/VR,smart home,and Internet of vehicles. Thus,the traffic in the network is surging,which gradually increases the pressure of the core network,and it is more and more difficult to control the network delay. At this time,the cloud-edge collaborative computing paradigm is proposed as a solution. To solve the problem of core network traffic control between the cloud and edges,this paper proposes a resource allocation and offloading decision algorithm to reduce the traffic of cloud-edge communication. First,this paper uses the designed resource allocation algorithm based on the divided time slot to improve the processed traffic of edges. Then,it uses the genetic algorithm to search the optimal offloading decision. Experimental results show that compared with the baseline schemes,the proposed algorithm can better improve the resource utilization rate of edges,and reduces the cloud-side communication traffic,and thus reducing the potential congestion of the core network.

**Keywords**　Mobile edge computing,Core network congestion,Resource allocation,Computation offloading

## 1　引言

随着移动互联网和物联网的快速发展,越来越多的智能化应用出现在人们面前,如 AR/VR、智能家居、车联网应用等。这些应用通常属于资源密集型应用,给资源受限的终端设备带来了极大的挑战,并且需要将数据上传到云计算服务器进行处理。然而,云计算服务器往往部署在远离用户的地方,随着网络数据流量的增长,网络日益拥堵,云计算已无法满足低时延需求的应用[1]。

作为一种很有前景的新型计算模式,移动边缘计算[2]能够有效解决云计算的上述不足,与云计算形成互补。移动边缘计算通过将边缘服务器部署在基站、网关、路由等地方,把计算中心从云计算中心下沉到靠近用户的地方,提供低时延服务以及可观的算力支持。因此,移动边缘计算作为缓解网络拥塞、提高用户服务质量的一种策略方案,已经成为解决物联网和本地化计算需求的新范式。在边缘处进行数据处理/预处理还可以有效减轻云端的流量压力,并减少潜在的核心网拥塞。

到稿日期:2020-07-03　返修日期:2020-08-26　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61872149);广东省杰出青年基金(2018B030306010);广东特支计划科技创新青年拔尖人才(2017TQ04X482);广州市珠江科技新星(201806010088);中央高校基本科研业务费专项资金
This work was supported by the National Natural Science Foundation of China(61872149),Natural Science Foundation of Guangdong Province for Distinguished Young Scholar (2018B030306010),Guangdong Special Support Program (2017TQ04X482),Pearl River S&T Nova Program of Guangzhou (201806010088) and Fundamental Research Funds for the Central Universities.
通信作者:张幸林(zhxlinse@gmail.com)

但是,边缘节点的计算资源有限,往往不足以处理所有的卸载任务,而用户任务有不同的时延要求,因此考虑在满足用户时延要求的前提下,采用边云合作的模式,将边缘节点的部分任务再卸载到云端。如何高效地利用边缘节点的计算资源,同时合理地将边缘节点的部分任务卸载到云端,既发挥边缘节点的低时延优势,又能降低边云通信的核心网络的带宽压力,成为边缘计算的重要研究课题。

目前的边缘计算资源分配策略从时间角度可以分为静态和动态两种。静态计算资源分配不考虑分配的时间长度。如 Wang 等[3]提出在用户层-接入层-云端三层模型中联合处理用户数据,对每一个设备的数据从用户层到云端都定义了所分配的计算资源和传输资源,并通过这些定义的资源量来构建一个凸优化问题,以求解得到最优的资源分配策略。其不足之处在于,这些已分配的计算资源在该任务集合的所有任务完成之前都被占用,即使任务集合中的某个任务提前完成,其占用的计算资源也不会被同任务集合中的其他任务所使用,从而造成空闲计算资源的浪费。动态计算资源分配则考虑了一个比较长的时间长度,并将该时间长度划分为很多等长的时间帧。如 Zhang 等[4]利用网络功能虚拟化技术,在密集云无线接入网中在线分配资源,并利用李雅普诺夫优化理论在平均能耗与平均时延之间进行权衡。可以看出,这类方法是通过优化较长的时间长度累积的平均指标,如平均时延、平均能耗、平均支出成本等,来优化每个时间帧的计算资源分配。相较于静态方法,动态方法可以保证性能指标在较长的时间内保持稳定。其不足之处在于,若单独取出一个时间帧来看,其仍然是一个静态计算资源分配,即在一个时间帧内,固定的任务集合所分配到的计算资源不会因为某个任务已经完成就提前释放给同任务集合中的其他任务使用,资源利用率仍然有待进一步提高。其动态性主要表现在不同时间帧的任务集合的不同,当考虑一个长期的系统效益时,在计算资源分配上依然是静态计算资源分配模式。

另外,任务集合利用计算资源的方式可以分为并行使用和串行使用两种。并行使用计算资源是指对于一个任务集合,每一个任务都被分配到各自的计算资源。在运行阶段,所有任务使用各自的计算资源同时执行。Yu 等[5]借助无人机的通信能力,联合优化无人机的位置、通信和计算资源分配以及任务分配决策,构造了物联网设备的服务延迟最小化和无人机的能源效率最大化问题。Zhang 等[6]借助分支限界法来改进搜索树算法,求解联合考虑计算卸载、内容缓存和资源分配的时延最小化问题。在上述两个模型中,所有用户使用的计算资源之和不能超过系统的计算资源总量,所有用户同时使用所分配的计算资源,即使某个用户使用结束,其释放的资源也不再分配给其他用户。串行使用方式则常见于不考虑单个边缘服务器的计算资源分配,如 Ma 等[7]建立的云边协作模型,不同的移动设备请求按时间顺序通过接入点,该模型根据最优决策将请求转移到边缘服务器或者云端服务器串行处理;或是预先分配好计算资源块作为分配单位,如 Bahreini 等[8]在边缘服务器上使用虚拟机技术预先分配固定资源量大小的多个虚拟机,然后使用拍卖机制将这些资源块分配出去,在每个虚拟机上数据都是串行处理。严格来说,串行使用中并没有考虑资源分配问题的,多个任务分配到同一台边缘服务器或者虚拟机后将按照串行方式使用所有计算资源,然后逐一处理。在并行使用方式中,所有计算资源的系统分配周期是从系统做出决策并将任务分配给用户开始,到整个用户集合中所有用户都完成任务为止,期间即使某个用户提前完成了任务并释放了所占用的资源,系统也不会将这部分资源回收并重新分配给还在处理中的任务,从而造成空闲资源的浪费。串行使用方式中,由于每一个任务都是使用所有的计算资源,一个任务结束,下一个任务就会接着使用,因此其计算资源利用率在理论上能达到百分之百。但是这种方式对于有延迟要求的任务集合来说,会存在执行顺序的问题,顺序分配不当会导致某些任务超时。

不管是静态分配、动态分配还是并行处理方式,都没有考虑用户集合内的空闲资源回收再利用的问题,而串行处理方式需要考虑任务的先后顺序。一个理想的计算资源分配情况是:在一个用户集合中,计算资源被分配出去之后,当其中的某个用户提前完成任务并释放了计算资源,系统能够将这部分空闲资源很快收集并重新分配给其他仍在处理任务的用户,这样在任一时刻的资源利用率都是百分之百。在这个设想中,最重要的步骤就是计算资源分配和预测任务结束时刻,以便在任务一结束就将资源回收并重新分配计算资源。这两步在系统运行时不断循环,直到所有任务在规定时限内完成。但是由于任务开始的时刻和所分配的计算资源会影响任务的完成时刻,导致分配计算资源和预测任务完成时间这两个步骤相互耦合,想要精确地在所有任务完成时间点上回收并重新分配计算资源,同时保证所有任务都按时完成是非常困难的。

为了在处理问题的复杂程度和计算资源利用率之间取得平衡,我们考虑将整条用户任务处理时间轴等分成连续的时间槽。由于任务需要在给定的时延内完成,因此用户集内可能的最长任务处理时间是可知的。在每一个时间槽开始时刻收集空闲资源并重新分配给仍在处理或还未处理的任务,以达到计算资源的有效利用。通过缩短时间槽的长度可以提高系统重分配资源的密度,当时间槽的长度无限小时,就逼近前文所述的理想情况。实际上,从后文的实验中可以看出,当时间槽的长度缩短到一定程度后,对系统收益的提高并不显著,因此使用这种等分时间槽的方式来代替理想情况下的计算资源分配和预测任务结束时刻是合理的。

本文的主要贡献如下:1)从边缘节点分流减压的角度提出了任务卸载和资源分配问题,目的是提高边缘节点的资源利用率,同时最小化云边之间的数据流量,从而达到为云边之间的核心网分流减压的作用。2)提出一个分割时间槽的资源分配算法(Slotted Time Resource Allocation Algorithm, STRA),在传统的并行一次性资源分配和精确调控之间进行平衡,有效提高边缘节点对任务的处理效率,并且应用遗传算法的思想,结合问题自身特点设计了任务卸载算法。3)通过将现有算法分为并行处理和串行处理两种类型,抽象出核心的分配方式来适应本文场景,并设计了多个模拟实验,展示了本文算法的高效性和可收敛性。

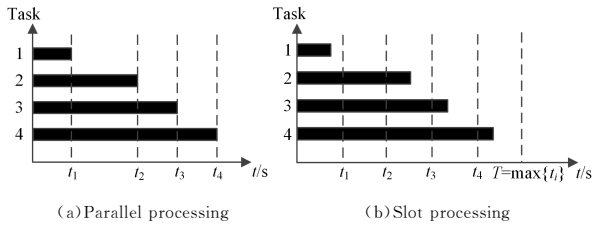
## 2 相关工作

在边缘计算的研究问题中,现有工作一般将计算卸载和资源分配进行协同处理,以优化任务时延、系统能耗和系统开

销等。Zhou 等[9]考虑最小化任务的完成时间，在工业物联网的背景下考虑通过计算资源限制来进行计算任务跨域卸载。他们将问题解耦并使用凸优化理论以及启发式模拟退火算法分别进行资源分配和任务卸载。Dong 等[10]考虑最小化计算能耗和时延，在实际用户请求具有优先级的情境下，在面向优先级业务的移动边缘计算中使用加权线性规划算法得到资源分配策略。另外，从服务提供商的角度，Jin 等[11]最小化所有提供商的部署开销，提出了基于精英保留策略的遗传算法，满足了第三方服务提供商的部署需求。而 Yu 等[12]考虑了传输功率这一条件，采用二分搜索法优化传输功率来降低传输时延和能耗，并采用非合作博弈论来解决多用户卸载决策问题，以降低系统开销。

对于不同场景下的资源分配问题，目前也有诸多研究。基于车联网场景，Wang 等[13]考虑了车辆的流动性和资源约束，将能耗和时延加权，并利用强化学习方法进行优化。Castellano 等[14]基于网络虚拟化技术，考虑了边缘部署应用的异构性，进而提出使用投票机制，通过节点之间的选举和协商来分配资源，使得系统效用最大化。Dab 等[15]基于多用户无线 wifi 场景，基于分支切割算法来联合优化任务和资源分配，使得在时间约束下系统能耗最小。考虑到单个边缘服务器独立操作的能力有限，Ndikumana 等[16]提出联合多个边缘服务器一起进行缓存分配和计算卸载，以最大化资源利用率。此外，Jošilo 等[17]基于斯塔克伯格博弈理论（Stackelberg Game Theory），让用户设备可以根据运营商和其他设备的决策来决定是否卸载，从而使自身收益达到最优，同时运营商也可以根据用户设备的决策进行最优资源分配而最小化成本。Tong 等[18]考虑了在多个用户和多个计算节点之间进行以用户为中心的公平资源分配问题，通过对用户的延迟容忍、比特错误率、能耗以及雾节点的能耗加权得到以用户为中心的效用函数，并以此为标准进行计算资源的分配。Jia 等[19]在用户、雾节点和云三者之间利用网络虚拟化技术，使用户向云请求虚拟资源，云再通过雾节点向用户分发实体资源。在这个过程中，他们研究了向用户分配有限的雾节点的资源问题，将其建模为双匹配问题，并提出了一个基于 DA（Deferred Acceptance）算法的双匹配策略。在物联网场景中，Yao 等[20]考虑将计算和存储资源打包成一个个虚拟机进行分配，在为任务租借更多的虚拟机来提高可靠性和租借尽可能少的虚拟机来降低成本之间取得平衡。Qian 等[21]利用非正交多路访问技术，将一个智能终端的数据分割并卸载到多个边缘服务器，同时考虑边缘服务器上分配的计算资源成本，使得延迟和成本的加权和最小，得到卸载和资源分配决策。Yu 等[22]考虑了一个允许 D2D 模式的互助场景，通过网络负载类型为设备选择合适的蜂窝模式、专用模式或重用模式来分配功率和信道资源，实现更高的吞吐量。Liu 等[23]考虑了 AR 场景中移动边缘计算对网络环境的适应问题，通过研究帧率和计算模型对提高服务质量和服务延迟的影响，得到使得整体平均服务质量最大的 AR 重配置和资源分配方案。

上述研究是在多个场景中进行计算资源的分配和任务卸载，但是本质上对相同任务集上的资源都是一次性分配的，如图 1(a)所示，较早完成计算任务的计算资源得不到有效的再利用，从而导致空闲资源的浪费。另外，目前还没有考虑时间延迟的云边协同的流量优化研究，因此，本文设计了分割时间槽的计算资源分配算法，在图 1(b)所示的时间轴中，每一个时间槽结束后将空闲资源重新分配，并且设计了基于遗传算法的计算任务卸载算法，以最小化云边之间的通信流量，缓解云端的流量压力。




(a)Parallel processing (b)Slot processing

图 1 任务的资源占用时间

Fig. 1 Resource occupancy time of tasks

## 3 减少核心网拥塞的边缘计算资源分配和卸载决策

### 3.1 系统模型

如图 2 所示，考虑一个独立的小型基站，基站内部署了一台小型边缘服务器。小型边缘服务器的总计算能力为 $F^e$ CPU 周期每秒，可以为用户提供卸载计算服务。在该基站的覆盖范围内有若干用户设备与覆盖用户设备的基站通过无线局域网连接，用户设备集合定义为 $\mathcal{N}=\{1,2,\cdots,N\}$。同时，基站还与远程云端服务器通过核心网连接，可以将超出边缘服务器承受能力的用户任务卸载到云端协作处理。考虑到云端的计算资源远远大于边缘服务器，所以假设云端计算资源为无限多，同时核心网的通信总带宽为 $B$ MHz。假设在边缘处理的用户集合为 $\mathcal{N}^e$，其数量为 $|\mathcal{N}^e|=N^e$；在云端处理的用户集合为 $\mathcal{N}^c$，其数量为 $|\mathcal{N}^c|=N^c$。每个用户设备都会产生一个任务，任务集合定义为 $\mathcal{A}=\{a_1,a_2,\cdots,a_N\}$。每个任务 $a_i$ 具有 3 个属性：任务本身的数据量大小 $d_i$、完成任务所需的总 CPU 周期数 $c_i$，以及完成任务的时间限制 $t_i$。这里，完成任务的时间限制不包括用户设备传输任务到边缘服务器所花费的时间。我们使用三元组来表示任务 $a_i=(d_i,c_i,t_i)$。

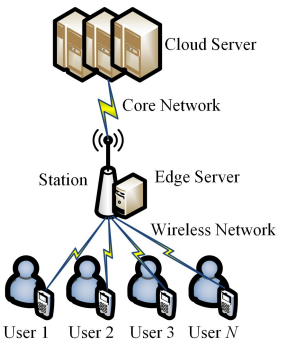



图 2 系统层次图

Fig. 2 System hierarchy diagram

整个系统的处理流程为：用户设备产生用户任务，这些任务被全部卸载到边缘服务器进行处理。边缘服务器根据一定的决策方法做出卸载决策，决定哪些用户任务可以在边缘处进行处理，哪些用户任务需要被再卸载到云端处理。做出卸载决策后，边缘服务器为保留在边缘处的每个用户任务分配合理的计算资源，为需要卸载的每个用户任务分配带宽资源。

### 3.2 问题建模

#### 3.2.1 计算资源分配

如果在边缘处需要满足所有用户任务的时延要求，则在边缘处理的所有用户任务的处理时间长度必定不会超过该用户集合中的最长时延要求。设这个时间长度为：

$$T=\max\{t_i\}, i\in\mathcal{N}^e \tag{1}$$

则 $T$ 可以作为用户任务分配到的计算资源的最长占用时间。这里的占用时间包括用户任务使用分配给它的计算资源的处理时间加上用户任务完成后释放计算资源但是还未被利用的空闲时间。如图 1(b)所示，为了定期收集空闲资源并重新分配，对 $T$ 进行等分，假设等分为 $W$ 段，每一段为一个时间槽，时间槽索引即为 $\mathcal{W}=\{1,2,\cdots,W\}$。在每个时间槽开头收集空闲计算资源并重新分配，从而达到及时再利用空闲资源的目的。则每一段时间槽的长度为：

$$\tau=\frac{T}{W} \tag{2}$$

在边缘处要保证用户任务 $a_i$ 在规定的时间限制 $t_i$ 内完成，则需要满足在时间限制内使用所分配的计算资源实现的计算量不小于所要求的计算量，即：

$$\sum_{j=1}^{T_i-1} f_{i,j}^e\times\tau+f_{i,T_i}^e\times(t_i-(T_i-1)\times\tau)\geqslant c_i, \forall i\in N^e \tag{3}$$

其中，$f_{i,j}^e$ 表示用户任务 $a_i$ 在第 $j$ 个时间槽在边缘处所分配到的计算资源量，$T_i=\lceil t_i/\tau\rceil$ 表示用户任务 $a_i$ 一共占用的时间槽数量。$\sum_{j=1}^{T_i-1} f_{i,j}^e\times\tau$ 表示任务 $a_i$ 在前 $T_i-1$ 个时间槽内的计算量，$f_{i,T_i}^e\times(t_i-(T_i-1)\times\tau)$ 表示在其时间约束下占用的最后一个时间槽内的计算量。

边缘处的计算资源总量总是有限的，因此需要满足计算资源量约束：

$$\sum_{i=1}^{N_e} f_{i,j}^e\leqslant F^e, \forall j\in\mathcal{W} \tag{4}$$

即在一个时间槽内，分配出去的计算资源量之和不能超过边缘所拥有的最大计算资源总量 $F^e$。

对于卸载到云端的任务，由于云端的计算资源远远大于边缘的计算资源，因此简化云端的计算时延，所有在云端处理的任务的计算时延都设为 $t^{c,comp}$。

#### 3.2.2 带宽资源分配

本文假设云边核心网使用的是正交频分多路复用技术，总的可用带宽为 $B$。假设边缘服务器的发射功率为 $p$，信道增益为 $h$，信道的噪声功率为 $\sigma$，则根据香农定理(Shannon's Theorem)可以得到用户任务 $a_i$ 的数据传输速率为：

$$r_i=b_i\log(1+\frac{ph}{\sigma}), \forall i\in\mathcal{N}^c \tag{5}$$

其中，$b_i$ 为用户任务 $a_i$ 分配到的带宽资源量，需要满足带宽总量的约束：

$$\sum_{i=1}^{N^c} b_i\leqslant B \tag{6}$$

则每个用户任务卸载到云端所需要的传输时间为：

$$t_i^{c,trans}=\frac{d_i}{r_i}, \forall i\in\mathcal{N}^c \tag{7}$$

用户任务 $a_i$ 需要在规定的时间限制内完成，而完成时间为任务卸载到云端所需要的传输时间、任务在云端的计算时间以及云端结果回传到边缘的时间之和。假设回传结果数据量很小，则回传结果所用的时间可以忽略不计，可以得到：

$$t^{c,comp}+t_i^{c,trans}\leqslant t_i, \forall i\in\mathcal{N}^c \tag{8}$$

#### 3.2.3 卸载决策问题

定义卸载变量为 $\boldsymbol{X}=[x_1,x_2,\cdots,x_N]$，其中 $x_i=1$ 表示将用户任务 $a_i$ 卸载到云端进行处理，而 $x_i=0$ 表示将用户任务 $a_i$ 留在边缘处进行处理。本文的目的是在计算资源、带宽资源以及用户任务时间限制的约束下得到最优的卸载决策和最优决策的资源分配变量，以最小化卸载的数据量，即：

$$P1: \min_{\{\boldsymbol{X},\{f_{i,j}^e\},\{b_i\}\}} \sum_{i=1}^{N} x_i d_i \tag{9}$$

$$\text{s.t.}\begin{cases} x_i=\{0,1\}, & \forall i\in\mathcal{N} \quad (9a)\\ \sum_{i=1}^{N}(1-x_i)f_{i,j}^e\leqslant F^e, & \forall j\in\mathcal{W} \quad (9b)\\ (1-x_i)(c_i-\sum_{j=1}^{T_i-1} f_{i,j}^e\times\tau-f_{i,T_i}^e\times(T_i-1)\times\tau)\leqslant 0, & \forall i\in\mathcal{N} \quad (9c)\\ x_i(t^{c,comp}+\frac{d_i}{r_i})\leqslant t_i, & \forall i\in\mathcal{N} \quad (9d)\\ \sum_{i=1}^{N} x_i b_i\leqslant B & (9e)\\ f_{i,j}^e\geqslant 0, & \forall i\in\mathcal{N}^e, j\in\mathcal{W} \quad (9f)\\ b_i\geqslant 0, & \forall i\in\mathcal{N}^c \quad (9g)\end{cases}$$

其中，约束(9a)规定了卸载变量必须为 0 或者 1。约束(9b)表示边缘服务器的总计算资源约束，在每一个时间槽中，在边缘处所有用户任务分配到的计算资源总量不超过边缘服务器本身所拥有的计算资源总量。约束(9c)表示在边缘节点处计算的用户任务在时间约束内需要完成对应的 CPU 周期数。约束(9d)表示卸载到云端的用户任务的时间约束，其计算时间和传输时间之和不得超过本身的时间限制。约束(9e)表示带宽资源约束，即需要卸载的用户任务所分配到的带宽总量不超过信道的带宽总量。约束(9f)，(9g)表示计算资源和带宽资源变量必须是非负的。可以看出，P1 问题中需要求解的变量既有整数变量，又有非整数变量，是一个混合整数规划问题，其求解是 NP 难的，一般很难求出精确解，或者求出精确解所要花费的时间和空间代价太高，因此我们考虑求出满意的近似最优解。

观察 P1，目标函数只与卸载决策变量 $\boldsymbol{X}$ 有关，且当卸载决策变量 $\boldsymbol{X}$ 已知后，约束可以分解为两个部分，一部分与边缘处的用户任务集合有关，另一部分与卸载到云端的用户任务集合有关，因此我们可以在假定卸载决策变量已知的条件下，定义以下两个子问题。

1)与边缘处的用户任务集合相关的子问题 P2：

$$\text{P2}: \min_{\{f_{i,j}^e\}} \sum_{i=1}^{N^e}\sum_{j=1}^{W} f_{i,j}^e \tag{10}$$

$$\text{s.t.}\begin{cases} \sum_{i=1}^{N^e} f_{i,j}^e\leqslant F^e, & \forall j\in\mathcal{W} \quad (10a)\\ c_i-\sum_{j=1}^{T_i-1} f_{i,j}^e\times\tau-f_{i,T_i}^e\times(t_i-(T_i-1)\times\tau)\leqslant 0 & (10b)\\ f_{i,j}^e\geqslant 0, & \forall i\in\mathcal{N}, \forall j\in\mathcal{W} \quad (10c)\end{cases}$$

即在满足边缘计算资源约束和用户任务时间约束的前提下，最小化所有用户任务所分配到的总计算资源量。其中约束

(10a)表示边缘计算资源总量约束，约束(10b)表示在边缘的计算量不能小于任务的计算量，约束(10c)为分配的计算资源量非负。

2)与卸载到云端的用户任务集合相关的子问题 P3：

$$\mathrm{P3}:\min_{\{b_i\}}\sum_{i=1}^{N^c}\left(t^{c,comp}+\frac{d_i}{r_i}\right) \tag{11}$$

$$\text{s.t.}\begin{cases} t^{c,comp}+\dfrac{d_i}{r_i}\leqslant t_i, & \forall i\in\mathcal{N}^c \quad (11a)\\ \sum\limits_{i=1}^{N^c}b_i\leqslant B & (11b)\\ b_i\geqslant 0, & \forall i\in\mathcal{N}^c \quad (11c)\end{cases}$$

即在满足带宽约束、云端可用计算资源约束、用户任务时间约束的前提下，最小化所有卸载的用户任务的完成时间。其中约束(11a)表示时间约束，约束(11b)表示带宽总量约束，约束(11c)表示分配的带宽量非负。

## 4 算法

### 4.1 资源分配问题

通过计算目标函数和每个约束的海森矩阵可以证明两个资源分配问题 P2 和 P3 都是凸优化问题。凸优化问题可以借助拉格朗日乘子法和 KKT 条件来求解析解。但是由于求解析解的过程比较困难，因此可以考虑求数值解。借助现有的凸优化求解器，可以求得这两个凸优化问题的近似最优解。因此，对于资源分配问题的算法设计，即子问题 P2，P3 的求解算法，在给定卸载决策变量的前提下，采用现有的凸优化求解器求解，可以分别得到两个子问题的目标函数最优值以及对应的求解变量的值。

### 4.2 卸载组合优化问题

对于文中的卸载组合优化问题，其搜索空间大小为 $2^N$，随着用户任务数量的增加决策变量的搜索空间大小呈现指数级增加，在如此大的搜索空间中，穷举算法、分支限界法等无法高效求解。因此，我们基于启发式的遗传算法[24]进行算法设计。

要使用遗传算法就必须先解决遗传算法的编码和适应度函数的问题。由于在本文的卸载组合变量问题中，卸载组合变量是可被遗传算法处理的二进制串，不需要进行额外的编码操作，因此，本文的优化问题中的编码就是求解的卸载组合变量本身。其次，关于适应度函数的定义，考虑到有些卸载变量会使得 P2 或者 P3 问题无解，即在给定的时间约束下，边缘或者云端没有足够的计算资源或带宽资源来及时处理分配到的用户任务，使得用户任务集中的某些任务超时，此时我们将对应的染色体的适应度值设置为 0，剩余的合法染色体(所谓合法，就是 P2 和 P3 问题都能得到满足)的适应值就是保留在边缘处进行处理的数据量大小，因为边缘处理的数据量越多，卸载到云端的数据量就越少。则适应度函数可定义为：

$$f(\boldsymbol{X})=\begin{cases}0, & \text{P2 or P3 unsolvable}\\ \sum\limits_{i=1}^{N}(1-x_i)d_i, & \text{otherwise}\end{cases} \tag{12}$$

得到适应度函数后，我们设计的卸载组合优化问题算法的伪代码如算法 1 所示。

**算法 1** 卸载组合优化问题算法

输入：任务集合 $A=\{a_1,a_2,\cdots,a_N\}$，边缘计算资源总量 $F^e$，带宽资源总量 B，分割时间槽数量 W，种群数 K，迭代次数 G

输出：卸载变量 $\mathbf{X}_{opt}=[x_1,x_2,\cdots,x_N]$，边缘计算资源分配矩阵 $\mathbf{F}_{opt}^{e,alloc}=\{f_{i,j}^e\}$，$i\in\mathcal{N}^e$，$j\in\mathcal{W}$，带宽资源分配向量 $\mathbf{B}_{opt}=\{b_i\}$，$i\in\mathcal{N}^c$

1. $\text{bestvalue}=\sum\limits_{i=1}^{N}d_i$
2. 初始化种群 $\mathbf{G}_1=\{\mathbf{X}_1^1,\mathbf{X}_2^1,\cdots,\mathbf{X}_K^1\}$
3. for g=1 to G do
4. 使用式(9)计算每个个体的适应值 $f_k^g=f(\mathbf{X}_k^g)$，$k\in\{1,2,\cdots,K\}$；
5. 从种群中选出适应值最高的 $f_{best}^g$，将其置于种群第一位；
6. if $f_{best}^g==\text{bestvalue}$ then
7. break；
8. end if
9. 使用轮盘赌＋随机竞争的方式选择父本 $\mathbf{X}_{father}^g$ 和母本 $\mathbf{X}_{mother}^g$；
10. for k=2:K
11. 使用父本和母本通过两点交叉、变异操作繁殖出两个后代，两个后代进行竞争，取适应值最高的一个作为子代 $\mathbf{X}_k^{g+1}$。
12. end for
13. end for
14. 从种群中选出适应值最高的 $f_{best}^g$，$\mathbf{X}_{opt}=\mathbf{X}_{best}^g$，根据这个最优卸载决策分别求解 P2 和 P3，得到 $\mathbf{F}_{opt}^{e,alloc}$ 和 $\mathbf{B}_{opt}$。

算法 1 的第 1 行计算了可能的最大适应值，这个值在第 6 行中用于判断算法是否可以提前结束，若能达到这个最大的适应值，则说明所有的用户任务都可以在边缘处进行处理，不需要卸载到云端，从而卸载到核心网的数据量为 0，这也是 P1 可能的最小值。另外，由算法的第 9 行可知，本文的选择操作是基于轮盘赌和随机竞争相结合的方式，即先使用轮盘赌算法选择 2 条染色体，然后按照一定的概率选取适应值最大的染色体，以较小的概率选择另一条染色体，这样的操作有助于加快遗传算法的收敛。至此，经过该算法，可以求得 P1 的近似最优解。

## 5 实验

### 5.1 实验环境

本文使用 matlab 编写算法程序来模拟系统的运行。模拟程序运行的系统为 64 位 Windows 10，使用的处理器为 Inter(R) Core(TM) i5-7200U CPU @ 2.50 GHz 2.7 GHz。考虑到遗传算法的可并行性能够加快结果的输出，通过在 matlab 中编程实现了双核并行处理。

实验模拟场景中有一个边缘服务器，该边缘服务器与一个云计算中心通过无线网连接。同时，假设每个用户会向边缘卸载任务。遗传算法的运行时间与遗传算法的各项参数均有关系，如种群数量大小、遗传算法迭代次数等，种群数量越大，算法迭代次数越多，就越能逼近全局最优解，但与此同时运行时间会成倍增加。考虑到实验算法所运行的电脑的性能以及结果的精确度，在本实验中遗传算法的种群数量设定为 10 个，遗传算法迭代次数设定为 50。若无特殊说明，本文的实验参数如表 1 所列。

表 1 参数设定

Table 1 Parameter setting

| parameter | value |
|---|---|
| $p$/dBm | 40 |
| $h$ | $0.2\times10^{-6}$ |
| $\sigma$/mW | $3.423\times10^{-8}$ |
| $F^e$/CPU cycle/s | $1\times10^3$ |
| $t^{c,comp}$/ms | 10 |
| $B$/MHz | 20 |
| $d_i$/Mbits | [1,5] |
| $c_i$/CPU cycles | [20,50] |
| $t_i$/s | [0.5,1] |
| $N$ | 30 |
| $K$ | 10 |
| $G$ | 50 |
| $W$ | $T\times200$ |

### 5.2 实验对比算法

由于现有研究中的算法不能直接适用于本文的场景，我们将现有研究中的计算资源分配算法按照引言所述分类后根据本文场景进行适配，得到以下算法与本文所提算法(STRA)进行比较。

(1)并行资源分配型算法(Para)[4-6]。不考虑计算资源的回收再利用，根据资源约束和任务时间约束，最小化总分配的资源，通过建模为凸优化问题解得最优的资源分配策略。

(2)串行处理型算法(Serial)[7-8]。将任务按照其规定时延从小到大排列，使用边缘所有的计算资源逐个任务进行处理，当处理到某个任务超时时，表示该任务集无法全部满足时延要求。

(3)按比例资源分配算法(Ratio)。将总计算资源按照任务所需最小计算资源(即刚好满足延迟时所需的计算资源量)比例进行分配，当所有任务的最小计算资源之和大于总计算资源时表示该任务集中存在某个任务无法满足时延要求。

### 5.3 实验对比变量

(1)卸载率，定义为卸载到云端处理的用户任务总数据量与全部用户任务总数据量的比值。该比值越小，说明留在边缘处进行处理的用户数据量越多，算法对计算资源的利用效率越高。

(2)平均资源利用率，定义为单位时间内使用的资源占总资源的平均比值。由前文所述可知，串行处理算法每个时刻都是使用所有的计算资源进行处理，因此其平均资源利用率恒为 1。对于传统的并行资源分配算法和按比例的资源分配算法，其平均资源利用率设为$\bar{\delta}$，推导过程如下。

如图 1(a)所示，将所有任务按照规定的时延从小到大排序，设任务$a_i$分配到的计算资源为$f_i$，总资源量为 $F$。

在$[0,t_1]$内，$\delta_1=\frac{f_1+f_2+\cdots+f_N}{F}\times100\%$；

在$[t_1,t_2]$内，$\delta_2=\frac{f_2+f_3+\cdots+f_N}{F}\times100\%$；

…

在$[t_{k-1},t_k]$内，$\delta_k=\frac{f_k+f_{k+1}+\cdots+f_N}{F}\times100\%$；

…

在$[t_{N-1},t_N]$内，$\delta_N=\frac{f_N}{F}\times100\%$。

根据时长加权得到：

$$\bar{\delta}=\frac{\delta_1t_1+\delta_2(t_2-t_1)+\cdots+\delta_N(t_N-t_{N-1})}{t_1+(t_2-t_1)+\cdots+(t_N-t_{N-1})}$$

$$=\frac{1}{t_N}\times\left[\frac{f_1+f_2+\cdots+f_N}{F}\times t_1+\frac{f_2+f_3+\cdots+f_N}{F}\times(t_2-t_1)+\cdots+\frac{f_N}{F}\times(t_N-t_{N-1})\right]$$

$$=\frac{1}{Ft_N}\left[(f_1t_1+f_2t_1+\cdots+f_Nt_1)+(f_2t_2+\cdots+f_Nt_2-f_2t_1-\cdots-f_Nt_1)+\cdots+(f_Nt_N-f_Nt_{N-1})\right]$$

$$=\frac{1}{Ft_N}\sum_{i=1}^{N}f_it_i=\frac{\sum_{i=1}^{N}c_i}{Ft_N}$$

对于本文的分割时间槽资源分配算法的平均资源利用率$\bar{\delta}$，推导过程如下：

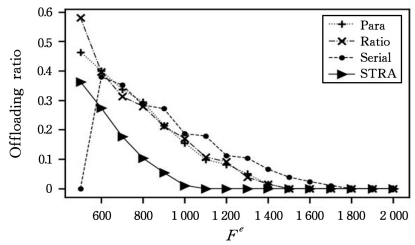
$$\bar{\delta}=\frac{(\sum_{i=1}^{N}f_{i,1}+\sum_{i=1}^{N}f_{i,2}+\cdots+\sum_{i=1}^{N}f_{i,W})\tau}{FW\tau}$$

$$=\frac{\tau\sum_{j=1}^{W}\sum_{i=1}^{N}f_{i,j}}{FW\tau}=\frac{\sum_{j=1}^{W}\sum_{i=1}^{N}f_{i,j}}{FW}$$

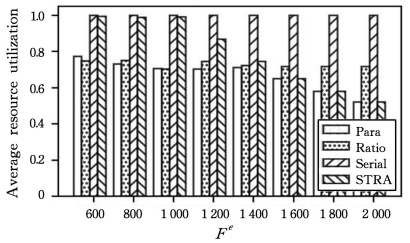
### 5.4 实验结果

#### 5.4.1 比较不同边缘计算能力下各种算法的性能

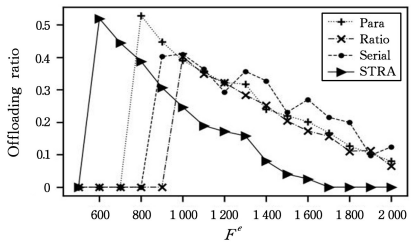
图 3 给出了不同边缘计算能力下各算法的性能。



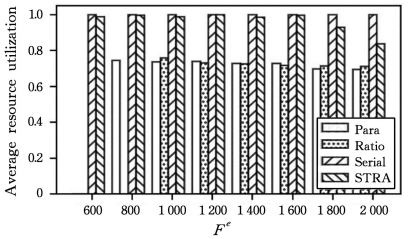

(a) $N=30$




(b) $N=30$




(c) $N=50$




(d) $N=50$

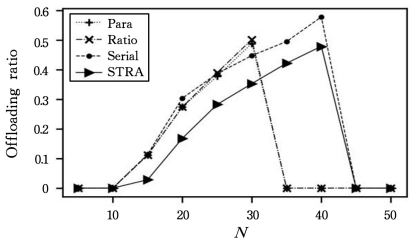
图 3 不同边缘计算能力下的性能对比

Fig. 3 Performance comparison with different edge computing capabilities

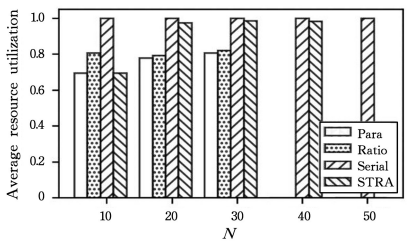
从图 3(a)、图 3(c)可以看出，随着总计算资源的增加，卸载率逐渐减少到 0，这是因为计算资源量的增加使得更多的任务能够在边缘处完成，且本文的分割时间槽分配算法有最小的卸载率，并且更早减小到 0，这说明本文算法在完成相同任务的前提下对总计算资源量的要求更小(我们注意到图 3(a)、图 3(c)一开始曲线为 0 的情况，表示该曲线对应的算法无法在规定时延内处理任务集，其对应的图 3(b)、图 3(d)的平均资源利用率缺失)。图 3(b)、图 3(d)显示，在资源紧张时，算法能够通过时间槽调整资源分配，以提高资源利用率，从而处理更大流量的任务；在资源充裕的时候，也可以通过较小的资源利用率处理跟其他算法一样多的任务，此时与并行算法效果相近。

5.4.2 比较不同用户数量下各算法的性能

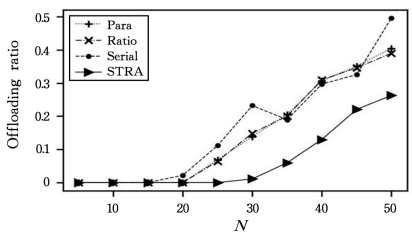
从图 4 可以看出，随着用户数量的增加，卸载率逐渐增加，这是因为用户数量的增加使得边缘计算资源无法支持所有任务，需要将任务再卸载到云端进行处理。



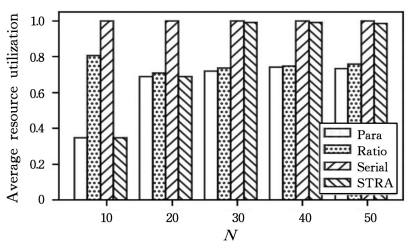

(a) $F^e=500$




(b) $F^e=500$




(c) $F^e=1\,000$




(d) $F^e=1\,000$

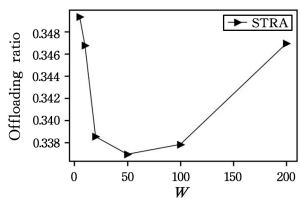
图 4 不同用户数量下的性能对比

Fig. 4 Performance comparison with different number of users

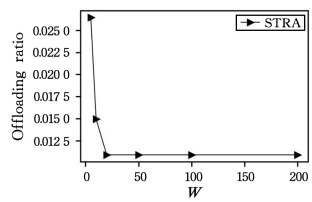
本文算法相较于对比算法而言有最小的卸载率，且在相同的计算资源下能支持更多的用户数量。这是因为在用户数少时，算法能够使用较少的资源处理一样多的任务，而当用户数增加时，所需要的计算资源逐渐逼近边缘资源所能承受的极限时，算法能够通过提高资源利用率来处理更多的任务，使得卸载率始终低于其他算法。我们注意到图 4(a)中最后为 0 的部分表示云端和边缘合作都无法在规定时间内完成任务集，即超出系统的处理能力，其所对应的图 4(b)的平均资源利用率数据缺失。

5.4.3 比较不同时间分割段数下本文算法的性能

从图 5 可以看出，不同槽数量对卸载率有所影响，槽数量越大，对资源重新分配的频率就越高，对资源分配的掌控就越精确，越有利于完成更多的任务。但是由于边界效应的存在，以及分槽越细凸优化问题求解的规模也越大，使得计算时间增加，因此并不是槽数量越多越好。在图 5(a)中资源量较小的情况下，分槽数量为 50 左右就能得到较好的卸载率，而对于图 5(b)资源量较大的情况，分槽数量为 20 左右时就可以收敛到满意的卸载率。




(a) $F^e=500, N=30$




(b) $F^e=1\,000, N=30$

图 5 STRA 在不同分割段数下的性能

Fig. 5 Performance of STRA with different segment numbers

5.4.4 本文算法的收敛性实验

从图 6 可以看出，本文算法的收敛是阶梯式的，一开始卸载率减小的速度很快，经过一个平缓的变化后，又有几次相对较小的减小幅度，越往后减小的幅度就越小，经过 50 次的迭代已经可以收敛到一个比较稳定的值，从而验证了本文算法的收敛性。

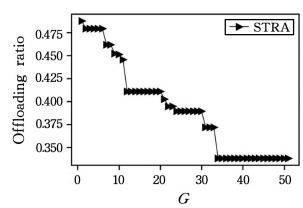



图 6 STRA 的收敛性

Fig. 6 Convergence of STRA

**结束语** 本文研究了基于云边协同的边缘计算节点的资源分配和卸载策略，提出了一种分割时间槽的计算资源分配算法。该算法能够通过细分时间长度来重新分配计算资源，从而达到提高计算资源利用率的目的。针对云边之间的通信流量问题，本文提出了在计算任务延迟约束下最小化云边通信流量的问题，并提出了一种基于遗传算法的卸载策略，得到全局近似最优解。本文所提方法的使用场景为计算能力较低的边缘设备与云端共同进行计算任务处理的情况，且边缘设备的计算能力越小，本文算法的优势就越大，这使得一些中小型，甚至微型边缘计算节点能够分散部署在任何地方，同样能够发挥其计算服务和流量处理能力，而不必考虑租借大场地

来部署大型边缘计算节点，使边缘计算节点的部署更加轻量化。但是由于使用的遗传算法需要多次迭代，导致计算时间过长，将来的工作将着重解决计算时间过长的问题。

## 参 考 文 献


[1] MAO Y, YOU C, ZHANG J, et al. A survey on mobile edge computing: The communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[2] 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.

[3] WANG P, YAO C, ZHENG Z, et al. Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems[J]. IEEE Internet of Things Journal, 2018, 6(2): 2872-2884.

[4] ZHANG Q, GUI L, HOU F, et al. Dynamic Task Offloading and Resource Allocation for Mobile-Edge Computing in Dense Cloud RAN[J]. IEEE Internet of Things Journal, 2020, 7(4): 3282-3299.

[5] YU Z, GONG Y, GONG S, et al. Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing[J]. IEEE Internet of Things Journal, 2020, 7(4): 3147-3159.

[6] ZHANG J, HU X, NING Z, et al. Joint resource allocation for latency-sensitive services over mobile edge computing networks with caching[J]. IEEE Internet of Things Journal, 2018, 6(3): 4283-4294.

[7] MA X, WANG S, ZHANG S, et al. Cost-efficient resource provisioning for dynamic requests in cloud assisted mobile edge computing[J]. IEEE Transactions on Cloud Computing, 2019: 1-1.

[8] BAHREINI T, BADRI H, GROSU D. An envy-free auction mechanism for resource allocation in edge computing systems [C]// 2018 IEEE/ACM Symposium on Edge Computing (SEC). IEEE, 2018: 313-322.

[9] ZHOU P, XU J C, YANG B. Cross-domain task offloading and computing resource allocation for edge computation in industrial Internet of things [J]. Chinese Journal on Internet of Things, 2020, 4(2): 96-104.

[10] DONGS Q, WU J H, LI H L, et al. Task scheduling policy for mobile edge computing withuserpriority[J/OL]. (2019-09-03) [2020-06-19]. https://doi. org/10. 19734/j. issn. 1001-3695. 2019. 03. 0131.

[11] JIN M, GAO S, LUO H, et al. Cost-effective resource segmentation in hierarchical mobile edge clouds[J]. Frontiers of Information Technology & Electronic Engineering, 2019, 20(9): 1209-1220.

[12] YU X, SHI X Q, LIU Y X. Joint Optimization of Offloading Strategy and Power in Mobile-Edge Computing[J]. Computer Engineering, 2020, 46(6): 20-25.

[13] WANG Y, WANG K, HUANG H, et al. Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications[J]. IEEE Transactions on Industrial Informatics, 2018, 15(2): 976-986.

[14] CASTELLANO G, ESPOSITO F, RISSO F. A distributed orchestration algorithm for edge computing resources with guarantees[C]// IEEE Conference on Computer Communications (IEEE INFOCOM 2019). IEEE, 2019: 2548-2556.

[15] 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 (CCNC). IEEE, 2019: 1-2.

[16] NDIKUMANA A, ULLAH S, LEANH T, et al. Collaborative cache allocation and computation offloading in mobile edge computing[C]// 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS). IEEE, 2017: 366-369.

[17] JOŠILO S, DÁN G. Wireless and computing resource allocation for selfish computation offloading in edge computing[C]// IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019: 2467-2475.

[18] TONG S, LIU Y, CHERIET M, et al. UCAA: User-centric user association and resourceallocation in fog computing networks [J]. IEEE Access, 2020, 8: 10671-10685.

[19] JIA B, HU H, ZENG Y, et al. Double-matching resource allocation strategy in fog computing networks based on cost efficiency [J]. Journal of Communications and Networks, 2018, 20(3): 237-246.

[20] YAO J, ANSARI N. Fog Resource Provisioning in Reliability-Aware IoT Networks [J]. IEEE Internet of Things Journal, 2019, 6(5): 8262-8269.

[21] QIAN L P, SHI B, WU Y, et al. NOMA-Enabled Mobile Edge Computing for Internet of Things via Joint Communication and Computation Resource Allocations[J]. IEEE Internet of Things Journal, 2020, 7(1): 718-733.

[22] YU G, XU L, FENG D, et al. Joint Mode Selection and Resource Allocation for Device-to-Device Communications [J]. IEEE Transactions on Communications, 2014, 62(11): 3814-3824.

[23] LIU Q, HAN T. DARE: Dynamic Adaptive Mobile Augmented Reality with Edge Computing[C]// International Conference on Network Protocols. 2018: 1-11.

[24] MITCHELLM. An Introduction to Genetic Algorithms[M]. MIT Press, Cambridge, 1998.

**LI Zhen-jiang**, born in 1997, postgraduate. His main research interests include mobile edge computing and so on.

**ZHANG Xing-lin**, born in 1987, Ph. D, associate professor, is a member of China Computer Federation. His main research interests include mobile edge computing and mobile crowdsensing.