·热点与综述· 文章编号:1000-3428(2021)02-0019-07 文献标志码:A 中图分类号:TP391

移动边缘计算中的卸载决策与资源分配策略

杨 天,杨 军
(宁夏大学 信息工程学院,银川 750021)

摘 要: 为在移动边缘计算服务器计算资源有限的情况下最小化系统总成本,提出一种多用户卸载决策与资源分配策略。优化任务执行位置选择和计算资源分配过程,对基于精英选择策略的遗传算法在编码、交叉、变异等操作方面进行改进,设计联合卸载决策与资源分配的improve-eGA算法。实验结果表明,与All_local、All_offload、RANDOM和CGA等算法相比,improve-eGA在迭代次数、任务周期数、任务传输数据量等影响因素下系统总成本均为最低,验证了所提策略的有效性。

关键词: 移动边缘计算;计算资源;卸载决策;资源分配;遗传算法

开放科学(资源服务)标志码(OSID):

中文引用格式: 杨天,杨军.移动边缘计算中的卸载决策与资源分配策略[J].计算机工程,2021,47(2):19-25.
英文引用格式: YANG Tian,YANG Jun.Offloading decision and resource allocation strategy in mobile edge computing[J].Computer Engineering,2021,47(2):19-25.

Offloading Decision and Resource Allocation Strategy in Mobile Edge Computing

YANG Tian,YANG Jun
(School of Information Engineering,Ningxia University,Yinchuan 750021,China)

【Abstract】 In order to minimize the total system cost when the computing resources of Mobile Edge Computing(MEC) servers are limited, this paper designs a multi-user offloading decision and resource allocation strategy. The strategy jointly optimizes the selection of task execution location and the allocation of computing resources, and improves the encoding, crossover and mutation parts of the Genetic Algorithm(GA) based on the elite selection strategy(e-GA).On this basis, improve-eGA algorithm is designed combining offloading decision and resource allocation.Experimental results show that, compared with the ALL_Local algorithm, ALL_Offload algorithm, RANDOM algorithm and Conventional Genetic Algorithm(CGA), etc., improve-eGA has the smallest total system cost under the influence of the iteration number, CPU working frequency, transferred data size of tasks, etc., which verifies the validity of the proposed strategy.

【Key words】 Mobile Edge Computing(MEC); computing resource; offloading decision; resource allocation; Genetic Algorithm(GA)
DOI: 10. 19678/j. issn. 1000-3428. 0058085

0 概述

随着通信技术的发展和智能化终端设备的普及,以物联网和移动互联网为核心的网络服务及应用应运而生,特别是5G网络的出现,显著提高了交互式游戏、增强现实(AR)、虚拟现实(VR)和车联网等计算密集型应用的渗透率[1]。此类应用需要在用户设备上消耗大量的计算资源和能耗来满足低时延需求,但由于用户设备计算能力和电池能量有限,因此很难满足这些要求[2]。为缓解这一矛盾,研究者提出移动边缘计算(Mobile Edge Computing,MEC)作为一种新型且可行的解决方案[3]。MEC是云化无线接入网(Cloud Radio Access Network,C-RAN)的一种新型范式,其通过在移动用户附近部署高性能服务器来增强移动网络边缘的计算能力[4],这将有助于满足5G业务的超低时延、超高能效以及超高可靠性等需求[5]。

计算卸载是MEC中的关键技术,主要包含卸载决策和资源分配两个部分[6],其通过有效的卸载决策和资源分配方案合理安排用户设备,将计算任务

基金项目: 赛尔网络下一代互联网技术创新项目(NGII20161206)。
作者简介: 杨 天(1995—),男,硕士研究生,主研方向为移动边缘计算、普适计算;杨 军(通信作者),教授。
收稿日期: 2020-04-16 **修回日期:** 2020-06-05 **E-mail:** dragon@nxu.edu.cn

卸载至MEC服务器,同时分配资源进行任务计算,以降低系统的时延和能耗。

近年来,国内外已有较多关于MEC卸载决策和资源分配的研究,研究者从不同角度提出了可行的解决方案。在卸载决策方面:文献[7]基于任务缓冲区以及本地传输模块与计算模块的状态,利用马尔科夫决策过程找到最佳的任务调度策略以降低能耗和时延;文献[8]针对绿色能源(太阳能)与电网能源双向支持的边缘计算系统,基于Lyapunov优化技术提出一种在线卸载算法,通过权衡平均响应时间与平均能耗成本生成最优卸载方案;文献[9]研究了超密集无线网络中单用户多基站情况下的MEC卸载问题;文献[10]针对MEC卸载系统,提出基于博弈论的功率分配算法,其在服务器计算资源的约束条件下,采用二分搜索法优化传输功率减小传输时延和能耗,利用非合作博弈论解决多用户卸载决策问题,降低系统开销;文献[11]通过引入移动云计算(Mobile Cloud Computing,MCC)辅助MEC,进一步降低了系统时延和能耗;文献[12]将执行任务划分为可迁移和不可迁移两个部分,结合改进的Q学习算法和深度学习算法最小化移动设备的能耗和时延;文献[13]提出一种计算切换策略以优化卸载决策。在资源分配方面:文献[14]在有执行延迟约束的条件下应用优先级队列分配边缘节点与云端节点,以满足约束,提高服务质量;文献[15]采用一种可分离的半定松弛算法联合优化了多用户情况下任务卸载的决策方案和通信资源分配方案;文献[16]提出一种基于深度强化学习算法对计算资源进行分配,以减少能量消耗;文献[17]提出一种激励拍卖机制用于移动用户(买家)和服务提供者(卖家)之间的资源交易,以有效分配计算资源,满足移动设备的服务需求。

然而,目前多数研究未考虑在MEC服务器计算资源有限的情况下如何更好地进行卸载决策和资源分配,进一步减少时延并降低能耗。本文针对此类情况提出一种多用户卸载决策与资源分配的联合优化策略。对基于精英选择策略的遗传算法(e-GA)进行改进,设计联合卸载决策与资源分配的improve-eGA算法,从而最小化系统总成本。

1 系统模型

本文考虑如图1所示的系统模型,其中包含1个eNodeB(eNB)和N个用户设备,使用eNB部署MEC服务器。假设模型中每个用户设备都有一个需要完成的计算密集型任务,每个用户设备都可以通过无线方式将任务卸载至MEC服务器或者在本地执行。但需要注意的是,MEC服务器的计算资源是有限的,可能不足以完成所有的计算任务。

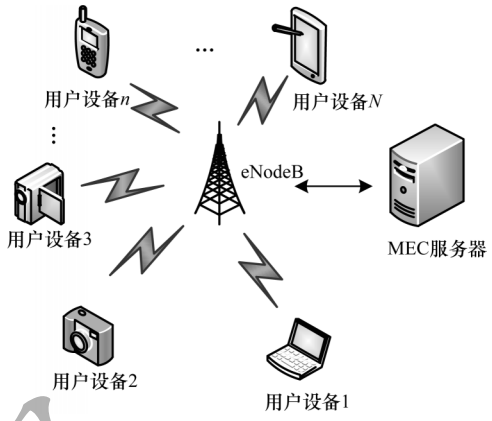

图1 多用户设备的系统模型

Fig.1 System model with multiple user devices

假设任务不能被划分,即每个用户设备只能通过本地计算或卸载计算来执行其任务。将$x_n \in \{0,1\}$表示为用户设备n的卸载决策,并将卸载决策向量定义为$\boldsymbol{X}=[x_1,x_2,\cdots,x_n,\cdots,x_N]$。如果用户设备$n$通过本地计算执行其任务,则$x_n=0$,否则$x_n=1$。

1.1 本地计算模型

若用户设备n选择在本地执行其任务W_n,则用户设备n的本地执行时延为T_n^{L},定义为:

$$T_n^{\mathrm{L}}=\frac{C_n}{f_n} \tag{1}$$

其中,C_n为完成W_n所需的CPU周期数,f_n为用户设备n的计算能力。需要注意的是,本地执行时延仅包括本地CPU的处理时延,并且不同用户设备之间的计算能力可能不同。

用户设备n在本地执行W_n的能耗定义为:

$$E_n^{\mathrm{L}}=E^{\mathrm{J}}\times C_n \tag{2}$$

其中,E^{J}为完成W_n每个CPU周期的能耗。根据文献[18],本文设定:

$$E^{\mathrm{J}}=f_n^2\times 10^{-27} \tag{3}$$

1.2 卸载计算模型

如果选择通过卸载计算来执行任务W_n,则整个卸载过程可分为以下三步:1)通过无线方式向eNB上传输入数据(即程序代码和参数),eNB将数据转发给MEC服务器;2)MEC服务器分配部分计算资源代替执行计算任务,此时用户设备n在等待计算结果;3)MEC服务器将执行结果返回给用户设备n,卸载过程结束。

根据上述过程,定义传输时延为:

$$T_n^{\mathrm{U}}=\frac{D_n}{r^{\mathrm{U}}} \tag{4}$$

其中,D_n为计算W_n所需输入数据的大小,包括程序代码和输入参数,r^{U}为系统模型无线信道中用户设备n的上行链路速率。

相应地,第1步中能耗定义为:

$$E_n^{\mathrm{U}}=T_n^{\mathrm{U}}\times P^{\mathrm{U}} \tag{5}$$

其中,P^{U}为用户设备n的上传功率。

将MEC服务器的处理时延T_n^{C}定义为:

$$T_n^{\mathrm{C}}=\frac{C_n}{f_n^{\mathrm{O}}} \tag{6}$$

其中，f_n^{O}为MEC服务器计算W_n所分配的计算资源。若F^{S}为MEC服务器全部的计算资源，则所有f_n^{O}之和不能超过F^{S}。因此，有以下限定条件：

$$\sum_{n=1}^{N}x_n\times f_n^{\mathrm{O}}\leqslant F^{\mathrm{S}} \tag{7}$$

假设用户设备n保持空闲，并将空闲状态的功率定义为P^{I}，则此时用户设备n的能耗为：

$$E_n^{\mathrm{C}}=T_n^{\mathrm{C}}\times P^{\mathrm{I}} \tag{8}$$

对于卸载过程的最后一步，所需时间是计算结果的下载时延T_n^{D}，定义为：

$$T_n^{\mathrm{D}}=\frac{D_n^{\mathrm{C}}}{r^{\mathrm{D}}} \tag{9}$$

其中，D_n^{C}是W_n处理结果的大小，r^{D}是用户设备n的下行链路速率。根据文献[19-20]可知，下载数据速率通常很高，并且处理结果的数据量远小于输入数据的数据量。因此，本文研究忽略此步中的时延和能耗。

根据式(4)~式(6)和式(8)，用户设备n卸载计算的执行时延T_n^{O}和能耗E_n^{O}分别表示为：

$$T_n^{\mathrm{O}}=T_n^{\mathrm{U}}+T_n^{\mathrm{C}} \tag{10}$$

$$E_n^{\mathrm{O}}=E_n^{\mathrm{U}}+E_n^{\mathrm{C}} \tag{11}$$

1.3　优化问题模型

本文将MEC系统的任务卸载和资源分配公式化为一个优化问题，目的是使MEC系统总成本（所有用户执行时延与能耗的加权和）G最小化。该优化问题以公式描述如下：

$$\min_{\boldsymbol{X},\boldsymbol{f}} G \tag{12}$$

$$G=\alpha\times\sum_{n=1}^{N}T_n+\beta\times\sum_{n=1}^{N}E_n \tag{13}$$

$$T_n=(1-x_n)\times T_n^{\mathrm{L}}+x_n\times T_n^{\mathrm{O}} \tag{14}$$

$$E_n=(1-x_n)\times E_n^{\mathrm{L}}+x_n\times E_n^{\mathrm{O}} \tag{15}$$

$$\boldsymbol{X}=[x_1,x_2,\cdots,x_n,\cdots,x_N] \tag{16}$$

$$\boldsymbol{f}=[f_1^{\mathrm{O}},f_2^{\mathrm{O}},\cdots,f_n^{\mathrm{O}},\cdots,f_N^{\mathrm{O}}] \tag{17}$$

s.t.

$$0\leqslant\alpha,\beta\leqslant 1$$

$$\alpha+\beta=1$$

$$x_n\in\{0,1\},\forall n\in\mathbb{N}$$

$$\sum_{n=1}^{N}x_n\times f_n^{\mathrm{O}}\leqslant F^{\mathrm{S}}$$

其中，$\boldsymbol{X}$表示卸载决策方案，$\boldsymbol{f}$表示计算资源分配方案，T_n与E_n分别是$\boldsymbol{X}$与$\boldsymbol{f}$下W_n的时延和能耗，α和β是权重系数，分别表示系统对时延与能耗的关注程度，两者取值范围为0~1且总和为1，具体数据可根据应用的实际需要或设备的实际情况进行设定。对于时延敏感性应用，可适当调高α、降低β，而对于用户设备能耗不足的情况，则可适当调高β、降低α。本文同等重视时延与能耗，因此，将α和β均设定为0.5。

2　卸载决策与资源分配策略

本文考虑MEC服务器计算资源有限的情况，对基于精英选择策略的遗传算法（e-GA）进行部分改进，优化其任务执行位置和资源分配过程，从而实现MEC系统总成本G的最小化。

2.1　编码

本文采用改进的二进制编码法，将卸载决策方案$\boldsymbol{X}$与计算资源分配方案$\boldsymbol{f}$联合表示为一个个体，具体如图2所示。需要注意的是，当$x_n=0$时，$f_n^{\mathrm{O}}=0$ GHz且$\boldsymbol{f}$应当满足式(7)所示的限制条件，以保证计算资源分配的合理性。

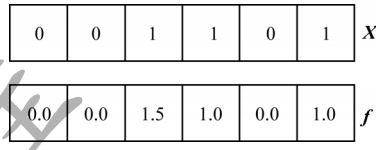

图2　编码示意图

Fig.2　Schematic diagram of encoding

2.2　适应度函数

本文提出的联合优化卸载决策与资源分配方案将适应度函数定义为：

$$F=H-G \tag{18}$$

其中，H为常数，其取值根据实际情况而确定。系统总成本越小，适应度值越大，算法通过F进行迭代，逐步找到最优解决方案。

2.3　初始化

对N个任务的执行位置进行初始化，即对每个任务随机产生0或1，表示在本地或MEC服务器上执行，然后对需要在MEC服务器上执行的计算任务进行计算资源分配。此处假设MEC服务器可分配的计算资源是预先规定的。例如，MEC服务器总的计算资源$F^{\mathrm{S}}=5$ GHz，每个任务可被分配的计算资源$f_n^{\mathrm{O}}\in\{1.0,1.5,2.0\}$ GHz，设A、B、C分别对应每种可分配资源的个数，因此，首先要统计进行卸载计算的任务个数N^{O}，然后对A、B、C进行求解，如式(19)所示，并将求解得到的结果随机分配到初始的卸载方案中。需要注意的是，本地计算任务的MEC分配资源为0 GHz。

$$\begin{cases}A+B+C=N^{\mathrm{O}}\\A+1.5B+2C\leqslant F^{\mathrm{S}}\end{cases} \tag{19}$$

2.4　种群选择

为防止当前种群的最优个体在下一代发生丢失，导致遗传算法不能收敛到全局最优解，本文采用精英选择策略，即对当前种群里的最优基因个体与下一代最优基因个体的适应度值进行比较，如果前者值大，则将当代的最优基因个体不进行交叉变异直接复制到下一代种群中。精英选择策略有两种方式：一种是将最优基因个体（elitist）直接加入到下一代种群中，扩大种群数目；另一种是将下一代种群中适应度值最小的基因个体与elitist进行代换，保持种群数目。本文策略使用第二种方式，具体算法描述如下：

算法1 种群选择算法

输入 当前种群集合M

输出 通过精英策略选择的下一代种群N

```
elitist=M[0]//最优基因个体
for i←0 to length(M)-1 do
if F(M[i])>=F(elitist) then
//比较适应度值
elitist=M[i]
end if
end for
N//通过轮盘赌进行选择的下一代种群
temp=N[0]
min=N[0]
site=0//记录最小值的位置
for i←0 to length(N)-1 do
if F(N[i])>=F(temp) then
temp=N[i]
end if
if F(N[i])<F(min) then
min=N[i]
site=i
end if
end for
//如果elitist的适应度值比temp的适应度值大
if F(elitist)>F(temp) then
//将N中适应度最小的基因个体替换为elitist
N[site]=elitist
end if
```

2.5 交叉

本文采用多点交叉法,以产生更高质量的基因个体。首先根据交叉概率随机生成一条长度为N的布尔值列表,列表中的每一项表示该点是否进行交叉,假设True代表交叉点,False代表非交叉点。卸载决策方案和计算资源分配方案都要进行交叉操作,具体如图3所示。

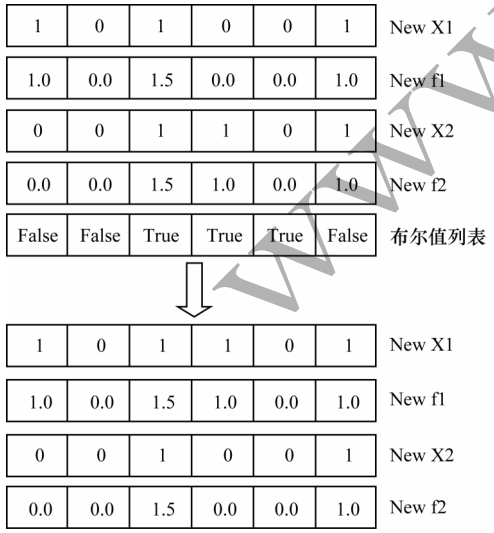

图3 交叉操作示意图

Fig.3 Schematic diagram of crossover operation

需要注意的是,若交叉后的计算资源分配方案不满足式(7)所示的限制条件,则交叉失败。交叉算法描述如下:

算法2 交叉算法

输入 基因个体$(\boldsymbol{X}_1,\boldsymbol{f}_1)$、$(\boldsymbol{X}_2,\boldsymbol{f}_2)$,根据交叉概率随机生成的长度为$N$的布尔值列表c_list

输出 交叉后的两条基因个体$(\boldsymbol{X}_1,\boldsymbol{f}_1)$、$(\boldsymbol{X}_2,\boldsymbol{f}_2)$

```
初始化:temp_X1,temp_f1←Ø
       temp_X2,temp_f2←Ø
for i←0 to length(c_list)-1 do
if c_list[i] then
temp_X2[i]=X1[i]
temp_f2[i]=f1[i]
temp_X1[i]=X2[i]
temp_f1[i]=f2[i]
else
temp_X1[i]=X1[i]
temp_f1[i]=f1[i]
temp_X2[i]=X2[i]
temp_f2[i]=f2[i]
end if
end for
temp1=0.0
temp2=0.0
for i←0 to length(temp_f1)-1 do
temp1+=temp_f1[i]
temp2+=temp_f2[i]
end for
if temp1<=F^s then
X1=temp_X1
f1=temp_f1
end if
if temp2<=F^s then
X2=temp_X2
f2=temp_f2
end if
```

2.6 变异

根据变异概率随机产生任务的变异序号n,然后对W_n的执行位置进行如下操作:

$$x_n = 1 - x_n \tag{20}$$

若变异后x_n由1变为0,则对应$f_n^{\mathrm{O}}=0$;否则f_n^{O}的值将根据F^{S}与其他已分配计算资源的差值进行确定。若差值为正,则f_n^{O}等于其计算结果;反之,则变异失败,f_n^{O}不变。变异算法描述如下:

算法3 变异算法

输入 基因个体$(\boldsymbol{X},\boldsymbol{f})$,变异序号$n$

输出 变异后的基因个体$(\boldsymbol{X},\boldsymbol{f})$

```
X[n]=1-X[n]
```

```
if X[n]==0 then
f[n]=0.0
else
temp=0.0
for i←0 to length(f)-1 do
if i!=n then
temp+=f[i]
end if
end for
if F^s-temp>0 then
f[n]=F^s-temp
else
X[n]=1-X[n]
end if
end if
```

3　仿真实验

通过仿真对比实验来验证本文策略的有效性。在仿真模拟中，假设每个任务的数据规模即传输数据量D_n（以Kb为单位）服从(300,500)之间的均匀分布，对应所需的任务周期数C_n服从(800 Megacycles,1 200 Megacycles)之间的均匀分布。MEC服务器的计算能力F^S为5 GHz，可分配计算资源与2.3节中一致，信道上传速率为2 Mb/s。为方便起见，假设N个用户设备是一样的，计算能力均为0.8 GHz，传输功率和空闲功率分别为500 mW和100 mW。模拟全部本地执行算法ALL_Local、全部卸载算法ALL_Offload、随机卸载分配算法RANDOM、标准遗传算法CGA以及文献[13]中的MET、MCT算法，并与本文提出的improve-eGA算法进行对比。在ALL_Offload算法中，MEC服务器为每个任务平均分配计算资源。

3.1　迭代次数对算法性能的影响

假设MEC系统一次处理6个任务，具体参数如表1所示。设种群数目为60，交叉概率为0.8，变异概率为0.05，迭代次数从1递增至100，迭代次数对系统总成本的影响如图4所示。可以看出：ALL_Local与ALL_Offload算法曲线不随迭代次数的增加而改变，且始终保持在较高数值；RANDOM算法曲线在整个迭代过程中震荡，不能收敛；CGA、MET、MCT与improve-eGA算法曲线随着迭代次数的增加逐步递减，逐渐收敛。CGA、MET、MCT及improved-eGA算法的具体对比如图5所示。可以看出：CGA算法曲线前期震荡，在55次迭代后逐步收敛至局部最优解，但收敛趋势缓慢，这是因为算法中交叉、变异的操作可能导致当前种群中的最优基因个体在下一代种群中发生丢失，而且这种最优基因个体丢失的现象会重复出现在进化过程中；MET、MCT和improve-eGA算法分别在38次、76次与17次迭代时收敛，三者的系统总成本相较于CGA算法分别降低1.5%、1.95%与2.86%，因为MET与MCT算法过多关注于系统任务的计算时延和完成时延，所以这两种算法的系统总成本略低于improve-eGA。综上可知，improve-eGA在迭代次数影响下系统总成本低于对比算法。

表1　模拟数据

Table 1　Simulation data

任务W_n	数据规模D_n/Kb	周期数C_n/Megacycles
W_1	420.813	1 026.202 556
W_2	388.653	1 000.232 493
W_3	383.832	950.645 076
W_4	455.276	1 057.215 149
W_5	342.818	977.907 010
W_6	469.315	955.633 363

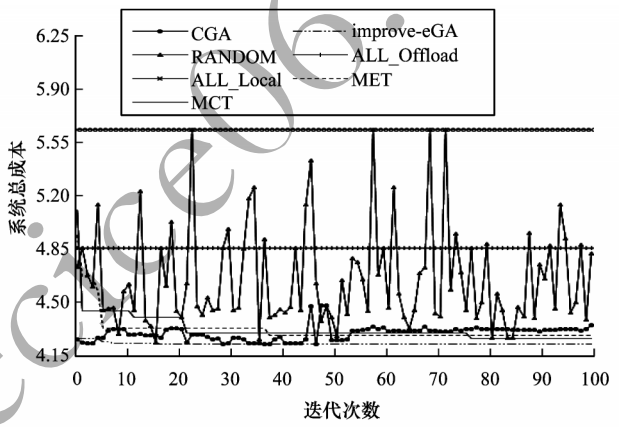

图4　迭代次数对系统总成本的影响

Fig.4　Influence of iterations on total system cost

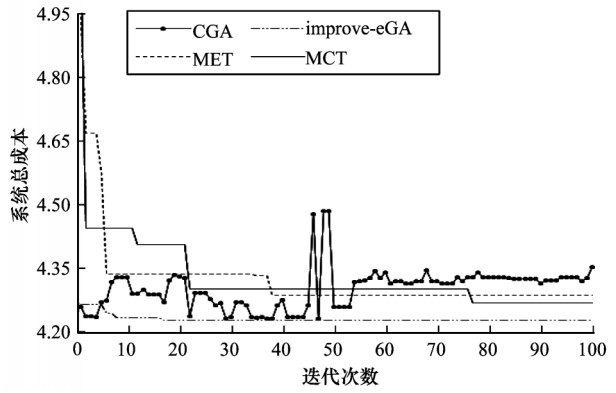

图5　CGA、MET、MCT与improve-eGA的系统总成本对比

Fig.5　Comparison of total system cost of CGA, MET, MCT and improve-eGAs

3.2 任务数量对算法性能的影响

分别模拟20~100的累计任务数量(以10递增),比较7种算法的系统总成本,其中迭代次数设定为100。如表2所示,可见系统总成本随着任务数量的增加而增加,这是由于任务数量的增大,导致系统需要更大的时延与能耗开销,从而使得系统总成本增加。从表2中可以看出:ALL_Local、ALL_Offload与RANDOM算法系统总成本较高,其余4种算法在任务数量小于70时差距较小;当任务数量大于等于70时,improve-eGA算法的实验结果更优。由此表明,在大规模任务计算卸载中,improve-eGA算法具有较大优势。

表2 7种算法在不同任务数下的系统总成本

Table 2 Total system cost of seven algorithms under different task numbers

任务数	系统总成本						
	RANDOM	ALL_Local	ALL_Offload	CGA	improve-eGA	MET	MCT
20	14.80	17.40	14.85	13.22	12.95	13.12	13.01
30	24.19	28.76	24.66	21.75	21.40	21.70	21.49
40	33.90	40.38	34.66	30.62	30.04	30.49	30.24
50	38.37	46.14	39.56	35.03	34.34	34.81	34.56
60	47.33	57.47	49.18	44.70	42.73	44.33	44.01
70	56.64	68.80	58.71	53.12	48.06	52.88	52.48
80	61.91	74.57	63.76	58.53	51.41	56.28	56.07
90	70.47	85.72	73.23	65.06	59.63	64.64	64.20
100	79.55	97.36	83.10	73.83	67.26	73.43	72.86

3.3 任务周期数对算法性能的影响

比较7种算法在任务周期数为800 Megacycles~1 200 Megacycles时(以100递增)系统总成本的变化,其中任务数量为70。如图6所示:随着任务周期数的增加,系统总成本也逐渐增加,这是因为任务周期数的增加会带来任务计算时间的增加,从而导致系统总成本增高;在不同任务周期数下,improve-eGA、MET和MCT算法系统总成本都低于其他4种算法,且improve-eGA算法结果更优。

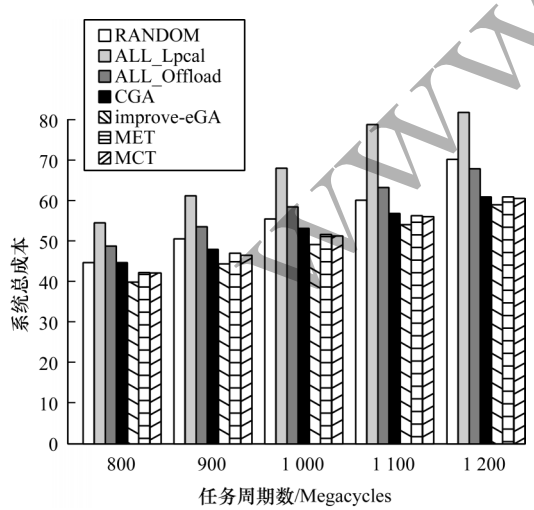

图6 任务周期数对系统总成本的影响

Fig.6 Influence of task cycle number on total system cost

3.4 任务传输数据量对算法性能的影响

比较7种算法在不同任务传输数据量下系统总成本的变化,其中任务数量为70。如图7所示:随着数据量从300 Kb增长到500 Kb,ALL_Local算法保持不变,这是因为所有任务在本地执行,不需要传输成本,但其系统总成本依然很高;其余6种算法系统总成本逐渐增高,这是因为数据量的增长,导致传输的时间成本增高,进而系统总成本增高;而improve-eGA算法在不同传输数据量下系统总成本最低。

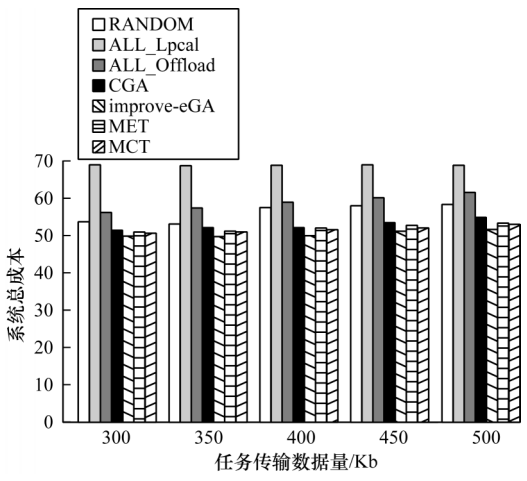

图7 任务传输数据量对系统总成本的影响

Fig.7 Influence of task transfer data size on total system cost

4　结束语

本文针对MEC计算资源有限的情况，研究多用户任务卸载决策和计算资源分配的联合优化问题。通过将研究问题转换为数学模型，提出一种MEC多用户卸载决策和资源分配策略。对基于精英选择策略的遗传算法进行改进，优化其编码、交叉、边缘等操作，在此基础上设计联合卸载决策与资源分配的improve-eGA算法。实验结果表明，该算法性能优于RANDOM、CGA、ALL_Local、AL_Offload、MET和MCT算法。下一步将设计一种考虑多个MEC服务器、任务可分割并可解决无线干扰问题的卸载决策和资源分配策略。

参考文献

[1] ZHANG Ke, LENG Supeng, HE Yejun, et al. Mobile edge computing and networking for green and low-latency Internet of things[J]. IEEE Communications Magazine, 2018, 56(5): 39-45.

[2] MAO Yuyi, YOU Changsheng, ZHANG Jun, et al. A survey on mobile edge computing: the communication perspective[J]. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322-2358.

[3] SHI Weisong, SUN Hui, CAO Jie, et al. Edge computing—an emerging computing model for the Internet of everything era[J]. Journal of Computer Research and Development, 2017, 54(5): 907-924. (in Chinese)
施巍松，孙辉，曹杰，等. 边缘计算：万物互联时代新型计算模型[J]. 计算机研究与发展，2017，54(5)：907-924.

[4] SABELLA D, VAILLANT A, KUURE P, et al. Mobile-edge computing architecture: the role of MEC in the Internet of things[J]. IEEE Consumer Electronics Magazine, 2016, 5(4): 84-91.

[5] TALEB T, SAMDANIS K, MADA B, et al. On multi-access edge computing: a survey of the emerging 5G network edge cloud architecture and orchestration[J]. IEEE Communications Surveys & Tutorials, 2017, 19(3): 1657-1681.

[6] XIE Renchao, LIAN Xiaofei, JIA Qingmin, et al. Survey on computation offloading in mobile edge computing[J]. Journal on Communications, 2018, 39(11): 142-159. (in Chinese)
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)：142-159.

[7] LIU Juan, MAO Yuyi, ZHANG Jun, et al. Delay-optimal computation task scheduling for mobile-edge computing systems[C]//Proceedings of 2016 IEEE International Symposium on Information Theory. Washington D. C., USA: IEEE Press, 2016: 1451-1455.

[8] NAN Yucen, LI Wei, BAO Wei, et al. Adaptive energy-aware computation offloading for cloud of things systems[J]. IEEE Access, 2017, 5: 23947-23957.

[9] CHEN X F, ZHANG H G, WU C, et al. Optimized computation offloading performance in virtual edge computing systems via deep reinforcement learning[J]. IEEE Internet of Things Journal, 2019, 6(3): 4005-4018.

[10] YU Xiang, SHI Xueqin, LIU Yixun. Joint optimization of offloading strategy and power in mobile-edge computing[J]. Computer Engineering, 2020, 46(6): 20-25. (in Chinese)
余翔，石雪琴，刘一勋. 移动边缘计算中卸载策略与功率的联合优化[J]. 计算机工程，2020，46(6)：20-25.

[11] WANG Yan, GE Haibo, FENG Anqi. Computation offloading strategy in cloud-assisted mobile edge computing[J]. Computer Engineering, 2020, 46(8): 27-34. (in Chinese)
王妍，葛海波，冯安琪. 云辅助移动边缘计算的计算卸载策略[J]. 计算机工程，2020，46(8)：27-34.

[12] MENG Hao, HUO Ru, GUO Qianying, et al. Machine learning-based stochastic task offloading algorithm in mobile-edge computing[J]. Journal of Beijing University of Posts and Telecommunications, 2019, 42(2): 25-30. (in Chinese)
孟浩，霍如，郭倩影，等. 基于机器学习的MEC随机任务迁移算法[J]. 北京邮电大学学报，2019，42(2)：25-30.

[13] LI Bo, HUANG Xin, NIU Li, et al. Task offloading decision in vehicle edge computing environment[J]. Microelectronics & Computer, 2019, 36(2): 78-82. (in Chinese)
李波，黄鑫，牛力，等. 车载边缘计算环境中的任务卸载决策和优化[J]. 微电子学与计算机，2019，36(2)：78-82.

[14] ZHAO Tianchu, ZHOU Sheng, GUO Xueying, et al. A cooperative scheduling scheme of local cloud and Internet cloud for delay-aware mobile cloud computing[C]// Proceedings of 2015 IEEE GLOBECOM Workshops. Washington D. C., USA: IEEE Press, 2015: 1-6.

[15] CHEN M H, LIANG B, DONG M. Joint offloading decision and resource allocation for multi-user multi-task mobile cloud[C]//Proceedings of 2016 IEEE International Conference on Communications. Washington D. C., USA: IEEE Press, 2016: 22-27.

[16] XU Jie, CHEN Lixing, REN Shaolei. Online learning for offloading and auto scaling in energy harvesting mobile edge computing[J]. IEEE Transactions on Cognitive Communications & Networking, 2017, 3(3): 361-373.

[17] JIN Along, SONG Wei, ZHUANG Weihua. Auction-based resource allocation for sharing cloudlets in mobile cloud computing[J]. IEEE Transactions on Emerging Topics in Computing, 2018, 6(1): 45-47.

[18] WEN Yonggang, ZHANG Weiwen, LUO Haiyun. Energy-optimal mobile application execution: taming resource-poor mobile devices with cloud clones[C]//Proceedings of INFOCOM'12. Washington D. C., USA: IEEE Press, 2012: 2716-2720.

[19] CHEN Xu, JIAO Lei, LI Wenzhong, et al. Efficient multi-user computation offloading for mobile-edge cloud computing[J]. IEEE/ACM Transactions on Networking, 2015, 24(5): 2795-2808.

[20] WANG C M, YU F R, LIANG C C, et al. Joint computation offloading and interference management in wireless cellular networks with mobile edge computing[J]. IEEE Transactions on Vehicular Technology, 2017, 66(8): 7432-7445.

编辑　金胡考