移动边缘计算环境下服务工作流的计算卸载

董 浩[1],张海平[2],李忠金[1],刘 辉[1]
1.杭州电子科技大学 计算机科学与技术学院,杭州 310018
2.杭州电子科技大学 信息工程学院,杭州 310018

摘 要:移动边缘计算(MEC)将计算和存储资源移动到移动网络的边缘,使其能够在满足严格的延迟要求的同时在移动设备处运行要求高处理的应用。它考虑了移动计算卸载问题,其中可以调用工作流中的多个移动服务来满足其复杂需求,并决定是否卸载工作流的服务,同时考虑了组件服务之间的依赖关系,并旨在优化执行移动服务的执行时间和能耗。针对上述问题运用了基于遗传算法(GA)的卸载方法,经过设计和实施后,部分修改传统遗传算法,以满足对所述问题的特殊需求。仿真实验表明,GA算法的实验效果都优于算法Local Execution和RANDOM得到的实验结果。

关键词:移动边缘计算;计算卸载;遗传算法;服务工作流
文献标志码:A **中图分类号:**TP391 doi:10.3778/j.issn.1002-8331.1806-0139

董浩,张海平,李忠金,等.移动边缘计算环境下服务工作流的计算卸载.计算机工程与应用,2019,55(2):36-43.
DONG Hao, ZHANG Haiping, LI Zhongjin, et al. Computation offloading for service workflow in mobile edge computing. Computer Engineering and Applications, 2019, 55(2):36-43.

Computation Offloading for Service Workflow in Mobile Edge Computing

DONG Hao[1], ZHANG Haiping[2], LI Zhongjin[1], LIU Hui[1]
1.School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou 310018, China
2.College of Information Engineering, Hangzhou Dianzi University, Hangzhou 310018, China

Abstract: Mobile Edge Computing(MEC) moves computation and storage resources to the edge of the mobile network, enabling it to run applications that require high processing at the mobile device while meeting stringent delay requirements. This paper considers the issue of mobile computing offloading, in which multiple mobile services in a workflow can be invoked to meet their complex needs and decide whether to uninstall workflow services, taking into account the dependencies between the component services, and aims to optimize the execution time and energy consumption of executing mobile services. A GA-based algorithm is applied to the above problems. After design and implementation, it partially modifies the traditional genetic algorithm to meet special requirements for the problem. Simulation experiments show that the experimental results of GA algorithm are better than Local Execution algorithm and RANDOM algorithm.

Key words: mobile edge computing; computation offloading; Genetic Algorithm(GA); service workflow

1 介绍

1.1 移动云计算与移动边缘计算

近年来,用户对数据速率和服务质量的要求呈增长状态。此外,智能手机、笔记本电脑和平板电脑的技术发展会出现新的高要求服务和应用。尽管新型移动设备在中央处理器(CPU)方面越来越强大,但即使这些设备也可能无法在短时间内处理需要大量资源的应用程序。此外,高电能消耗仍然限制用户在移动设备上充分享受高要求应用。这推动了移动云计算(MCC)概念的发展[1]。在MCC中,移动设备可利用强大的远程集中云

基金项目:浙江省高等教育教学改革项目(No.jg2015224)。
作者简介:董浩(1994—),男,硕士,研究领域为移动云计算,移动边缘计算,E-mail:1024704613@qq.com;张海平(1975—),男,副教授,研究领域为大数据,云计算,网络安全,企业信息化等方面的应用与研究;李忠金(1986—),男,博士,讲师,研究领域为云计算,工作流调度。
收稿日期:2018-06-12 **修回日期:**2018-10-22 **文章编号:**1002-8331(2019)02-0036-08
CNKI **网络出版:**2018-11-05, http://kns.cnki.net/kcms/detail/11.2127.TP.20181101.1416.033.html

(CC)的计算和存储资源,通过移动运营商的核心网络(CN)和因特网访问。移动云计算带来了几个优点[2]:(1)通过将应用卸载到云中来计算从而延长电池寿命,(2)为移动用户提供复杂的计算能力,以及(3)为用户提供更高的数据存储能力。为了解决等待时间较长的问题,应当将云服务移动到移动设备的附近,即移动到移动网络的边缘。边缘计算可以理解为移动云计算的特定情况。然而,在传统的移动云计算中,云服务通过因特网连接来访问[3],而在边缘计算的情况下,假定计算和存储资源在移动设备附近。因此,与移动云计算相比,移动边缘计算可以提供更低的延迟。而且,移动云计算通常将计算机的服务器放置在一个或几个位置,边缘计算以完全分布的方式进行部署。另一方面,边缘计算相对于移动云计算仅提供有限的计算和存储资源。

1.2　MEC的计算卸载

作为MEC的主要优势之一,计算卸载是通过将繁重计算任务迁移到边缘服务器来提高移动服务能力。在运行密集型计算服务时,计算卸载可为移动设备节省能源[4]。目前,由于虚拟化技术使移动边缘计算环境能够为移动设备远程运行服务,因此有大量关于计算卸载的研究[5]。这些研究主要是探索使计算卸载可行的想法和方法,制定最佳卸载决策[6]。有很多因素会对卸载技术的效率产生不利影响,特别是移动边缘网络中的移动设备和服务器之间的带宽限制以及必须在其间交换的数据量。当前研究已经提出了许多算法来优化卸载策略,以提高计算性能和节约能量消耗[7]。这些算法和技术主要分析一些系统参数,包括网络带宽、计算能力、可用内存、服务器负载以及移动设备和移动边缘服务器之间交换数据量。尽管上述参数对于大多数移动服务来说似乎已经足够了,但是随着移动应用程序的快速变化和计算复杂性的增加,移动服务工作流程也应该包括在未来的卸载策略中。随着用户需求变得越来越复杂,单一服务难以满足这种需求,因此需要在工作流中组合多种服务来执行复杂的任务。例如,为了满足商务旅行需求,可以生成由三个服务组成的服务工作流:(1)天气预报,(2)航班预定和(3)住宿预订,(2)和(3)可以并行执行,因为它们都取决于(1)提供的结果。这个简单的例子证明了为什么在设计卸载策略时应该考虑服务组件之间的依赖关系。当为这个工作流做出卸载决定时,应该解决以下问题:上述组件服务可以为具有数据依赖性的工作流。组合服务中的每个组件服务可以在移动设备上本地执行,也可以远程卸载到移动边缘服务器执行。由于它们的依赖关系,执行顺序在这里非常重要,在执行任何执行之前必须仔细考虑。

在这项工作中,为了解决上述问题,(1)提出了一种结合工作流调度的新型卸载策略,其组件之间具有相互依赖关系;(2)设计并实现了基于遗传算法(GA)的优化方法来做卸载决策。本文提出的解决方案可以最小化移动服务工作流程的执行时间和能耗。

2　相关工作

近些年来,随着移动智能设备的普及,用户对数据速率和服务质量的要求越来越高,这就促进了移动边缘计算的产生和发展。作为从移动云计算衍生而来的技术,移动边缘计算在计算卸载、任务迁移等方面可以借鉴移动云计算。移动云计算在考虑计算卸载问题时通常结合工作流调度来解决需求。文献[8]提出了基于延时传输机制的多目标工作流调度算法MOWS-DTM。该算法基于遗传算法,结合工作流的调度过程,在编码策略中考虑了工作流任务的调度位置和执行排序,有效地优化移动设备的能耗和工作流的完工时间。文献[9]提出了一种方法,把工作流调度分为计划和执行两个阶段,先运用改进的遗传算法对工作流系统中的任务执行顺序和资源分配做好全局优化,然后再按照计划执行,达到执行时间最短的目的。

MEC已经成为实现物联网和5G的关键技术[10-12]。近年来,来自学术界和工业界的研究人员对与MEC有关的各种问题进行了研究,包括系统和网络建模、最优控制、多用户资源分配、实施和标准化。然后,很多研究工作概述了不同侧重点的MEC领域,包括系统模型、体系结构、启用技术、边缘缓存、边缘计算卸载以及与物联网和5G的连接[13-21]。文献[13]中还从协作缓存和处理以及多层干扰消除等多个角度介绍了5G网络中几种MEC的使用场景。在文献[14]中,调查了MEC中边缘计算、缓存和通信(3C)的新兴技术。此外,还讨论了云技术、SDN/NFV和智能设备等MEC的关键功能。文献[15]中介绍了MEC平台的概述,其中讨论了不同的现有MEC框架,体系结构及其应用场景,包括Femto-Clouds、REPLISM和ME-VOLTE。文献[16]的研究侧重于MEC中的支持技术,如云计算、虚拟机、NFV、SDN,这些技术允许灵活的控制和多用户支持。在文献[17]中,作者对不同的MEC应用、服务模型、部署场景以及网络架构进行了分类。文献[18]中的研究为MEC应用程序提供了分类标准,并确定了研究和开发的潜在方向,如内容扩展、本地连接以及数据聚合和分析。文献[19]中的研究集中在MEC的计算卸载中的三个关键设计问题,即卸载决策、计算资源分配和移动性管理。此外,还有MEC在物联网中的作用,即创建新的物联网服务。在文献[20]中通过参照物联网使用案例强调了MEC部署示例。此外,从应用程序开发人员、服务提供商和网络设备供应商的角度来讨论与MEC有关的潜在商业机会[21]。文献[22]中作者设计了一种高效的分布式计算卸载算法,可以实现多用户场景下的负载均衡。

3　移动边缘计算模型

本章提供了移动服务计算的所有关键概念的明确定义。移动服务由移动用户通过设备在工作流程内调用。每个移动服务通过执行任务接收输入并产生输出;它被模拟为一个三元组 $s=(d_i,d_o,wl)$;其中 d_i 是输入数据的大小，d_o 是输出数据的大小，wl 是该服务执行所需的CPU该服务执行所需的工作量。

移动设备是手机、平板电脑或任何能够连接到互联网并从云中请求服务的便携式设备;它被建模为一个四元组 $m=(c_M,P_M,P_{\text{up}},P_{\text{down}})$;其中 c_M 是移动设备的CPU计算能力(以GHz为单位)，P_M 是本地运行服务时移动设备的功耗，P_{up}、P_{down} 是上传/下载时移动设备的功耗。卸载策略由向量表示,以确定哪些服务必须在本地运行,哪些服务必须卸载到服务器;它被建模为 $X=\{x_0,x_1,\cdots,x_i,\cdots,x_{n-1}\}$,其中 n 是移动服务工作流中服务的数量,并且 x_i 决定第 i 个服务是否将被卸载。$x_i=0$ 意味着第 i 个服务必须在本地运行；$x_i=1$ 意味着它必须被卸载到边缘服务器。

在移动边缘计算平台中,移动设备连接到基站,以从边缘服务器建立和部署服务。当移动用户执行计算工作流程时,可以通过移动网络将工作流程的服务卸载到边缘服务器。在边缘云中,边缘服务器接收来自移动用户的请求并在适当的时候处理它们。然后通过移动网络返回给移动用户。在这项工作中,假设所有服务都可以在本地运行或卸载到边缘服务器。

4　MEC卸载系统

在本章中,首先概述提出的卸载系统。图1描述了提出的卸载系统的框架。如图1所示,任务 $S1$、$S4$ 在本地执行,任务 $S2$、$S3$、$S5$ 上传至边缘服务器执行,且任务的执行存在顺序关系。本文主要关注此框架的卸载流程,旨在提出最佳卸载策略,同时最小化工作流程的执行时间和能耗。为此,目标函数被定义为运行服务的执行时间加上其能耗的加权和;这个目标函数是为每个移动设备定义的,并且被制定如下:

$$F(m)=w_m\times L_m+(1-w_m)\times E_m \tag{1}$$

其中 L_m 是由移动设备请求的工作流程的总体执行时间，E_m 是在执行工作流程期间由移动设备消耗的总能耗。权重系数 w_m 根据移动设备的状态来设置;例如,当移动设备的电池充满时,可以提供更高的 w_m 值;当其能量下降到阈值以下时,必须设置较低的值。对于本实验,设置 $w_m=0.5$ 以解决这个问题。但是,如果需要,可以根据特定用户的要求调整此权重。以下写明了所做的其他假设。

(1)移动设备一次只能将计算卸载到一个边缘服务器。

(2)在移动设备和基站之间传输数据期间的信号强度相当稳定。

(3)在本地执行移动服务期间,移动设备的能耗远高于其待机模式下的能耗。因此,移动设备在待机模式下的能耗可以忽略不计。

可以按如下方式计算在函数(1)中的主要目标函数的分量。

(1)本地执行服务的执行时间:

$$rt_l=\frac{wl_l}{c_M} \tag{2}$$

其中 wl_l 是非卸载服务在移动设备执行的工作量，c_M 是移动设备的CPU计算能力(以GHz为单位)。

(2)本地执行服务的能源消耗:

$$E_l=rt_l\times P_M \tag{3}$$

(3)卸载服务的执行时间:

$$rt_e=\frac{d_i}{Tx_i}+\frac{wl_e}{c_E}+\frac{d_o}{Tx_o} \tag{4}$$

其中 Tx_i、Tx_o 是移动用户开始向边缘服务器发送或接收卸载服务的输入和输出数据数据传输速率(以Kb/s为单位)。wl_e 是卸载的工作量，d_i、d_o 是其输入/输出数据的大小。c_E 是边缘服务器的CPU计算能力(以GHz为单位)。

卸载服务的能源消耗:

$$E_e=\begin{cases}P_{\text{up}}\times\dfrac{d_i}{Tx_i}, \text{上传}\\ P_{\text{down}}\times\dfrac{d_o}{Tx_o}, \text{下载}\end{cases} \tag{5}$$

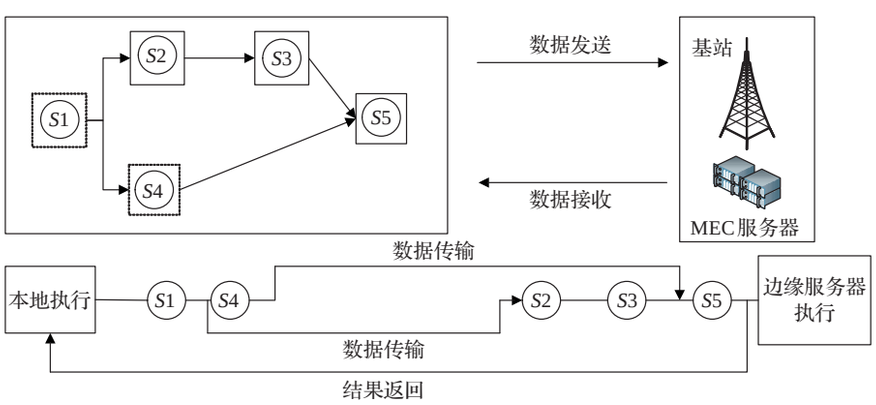

图1　卸载系统模型

其中 P_{up}、P_{down} 是上传和下载时移动设备的功耗。

基于上述计算的组件,方程(1)将是:

$$F(m)=w_m\times L(rt_s)+(1-w_m)\times\left(\sum_l E_l+\sum_e E_e\right) \quad (6)$$

其中 $L(rt_s)$ 是在本地和云资源上执行移动服务(工作流)的总体执行时间。服务的执行时间定义为用户启动移动服务的时间与请求的服务产生最终结果的时间之差。如果卸载计划可以将工作流程拆分为多个流程以在本地和云服务器上并行运行,则可以减少工作流程的执行时间。

5　卸载算法设计

在本章中,说明了基于遗传算法的计算卸载算法(GA),用遗传算法为移动服务工作流制定计算卸载计划。首先,给出基于遗传算法的解决方案的基本概述。然后,介绍算法GA,该算法针对移动环境中服务工作流程的计算卸载问题。在工作流的调度过程中,主要关注工作流任务的执行位置和执行顺序。在移动边缘计算环境中,一个任务可以在移动设备或者在边缘服务器执行,所以不同的执行位置造成不同的设备能耗和完成时间。由于移动边缘计算环境中的计算资源有限,仅有移动设备和边缘服务器两种。当两个任务同时竞争一个资源时,这就带来了资源竞争的问题。因此,通过排序任务执行的顺序,来解决任务执行过程中的资源竞争,排序中靠前的任务将具有占据资源的优先权。

下面以图2为例,阐述不同的任务排序方式对工作流调度的影响。假设一个包含5个任务的工作流,调度结果为3个任务在本地执行和2个任务在云端执行。如图3所示,当工作流任务的执行顺序是 $S1,S2,S3,S4,S5$ 时,假设每个任务执行时间相同,都为一个时间片,数据传输时间也需耗时一个时间片,任务 $S1$、$S2$ 和 $S4$ 执行各需要一个时间片,由于任务 $S3$ 在边缘服务器上执行,任务 $S2$ 完成之后,在任务 $S4$ 执行时就可以把数据传输到边缘服务器,那么总的任务执行时间为六个时间片。但是当任务的执行顺序是 $S1,S4,S2,S3,S5$ 时,如图3所示,总的任务执行时间为七个时间片。

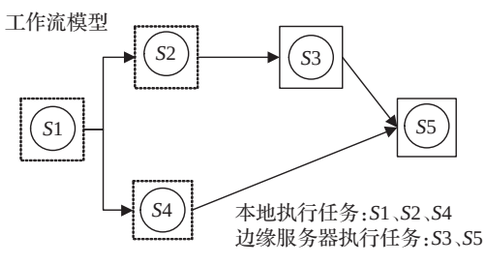

图2　工作流调度模型

5.1　编码

遗传算法是一种基于种群的优化方法,它使用一组解决方案来寻找全局优化的解决方案。编码是应用遗传算法时要解决的首要问题,也是设计遗传算法时的一个关键步骤。总的来说,这些编码方法可以分为三大类:二进制编码法、浮点编码法、符号编码法。本文采取的是二进制编码法。假设 n 是任务总数,用集合 $X=\{x_0,x_1,\cdots,x_i,\cdots,x_{n-1}\}$ 来表示工作流任务调度位置。令 $x_i=0$ 表示任务 S_i 在本地执行,$x_i=1$ 表示任务 S_i 在调度到边缘服务器执行。用集合 $Y=\{y_0,y_1,\cdots,y_i,\cdots,y_{n-1}\}$ 来表示工作流任务调度的顺序。其中 y_i 表示工作流中的某个任务。理论上来说 $y_i\in\{s_0,s_1,\cdots,s_i,\cdots,s_{n-1}\}$,但是任务的先后顺序必须受到条件限制。因此,源任务 s_0 必须排在第1个,尾任务 s_{n-1} 在集合 Y 中必须排在最后一个位置。图4中描述的是移动边缘计算环境下工作流调度的编码方案示例。

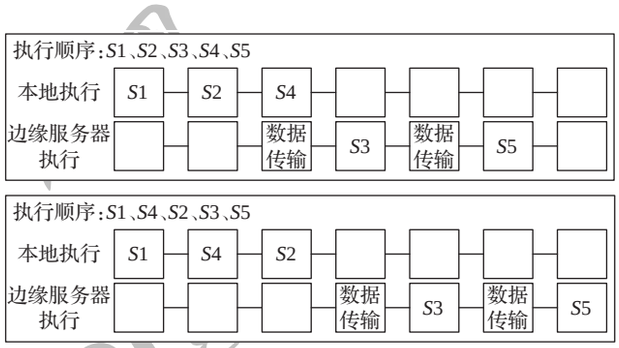

图3　任务执行顺序对工作流调度的影响

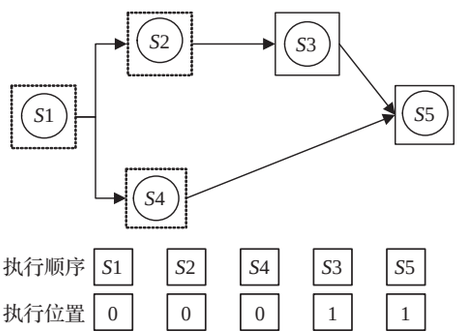

图4　基于遗传算法的编码方案

5.2　适应度函数

适应度函数用于评估可能的解决方案以找到最优解。对于GA,使用目标函数(1)计算每个任务的适应度值。这个目标函数由两部分组成:整个移动服务工作流程的总执行时间,以及执行整个移动服务工作流程的总能耗。这两个部分都必须最小化。给定适应度函数,GA将迭代地找到渐近最优解。

5.3　初始化

首先,对于一个任务 S_i 的执行位置,产生[0,1]之间的随机数 P_i,如果 $P_i\leqslant 0.5$,那么设置 $x_i=0$;否则,$x_i=1$。用同样的方法遍历所有任务,就可以得到位置集合 X 工作流任务位置初始化的伪代码如下所示,需要循环 N 次初始化种群中的每个个体,其中 N 为种群

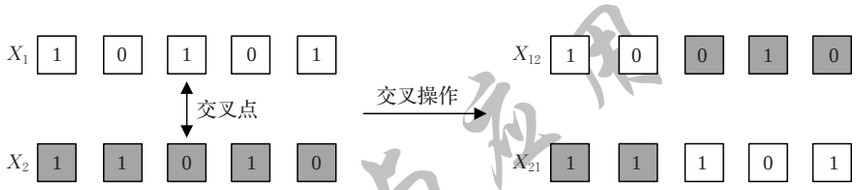

图5 任务执行位置单点交叉过程

的规模。

算法1 任务位置初始化

```
BEGIN
01. while s_i∈S do
02.     生成[0,1]之间的随机数 P_i ;
03.     if P_i≤0.5 then
04.         x_i=0 ;
05.     else
06.         x_i=1 ;
07. end while
END
```

然后,对于任务排序初始化,由于任务之间必须满足一定的顺序关系,所以使用可排序任务集合 E 来记录当前可排序的任务,并从该集合中随机选取一个任务进行排序。其中,可排序任务的定义是该任务没有前驱任务或者其前驱任务已经排序完毕按此方法继续选取下一个任务,直到形成一组可行的任务序列。任务排序初始化伪代码如下所示,同样需要循环 N 次初始化种群中的每个个体,随机产生的任务序列可能会存在重复。

算法2 任务排序初始化

```
BEGIN
01. E=∅;//待排序任务集合
02. R={s_0};//已排序任务
03. S=S-{s_0};
04. y_0=s_0;
05. index=0;//排序编号
06. for S≠∅ do
07.     for s_i∈S do
08.         if pre(s_i)⊂R then
09.             E=E+{s_i};
10.         end if
11.     end for
12.     随机从集合 E 中选取一个任务 s_i
13.     y_i=s_index;
14.     index++;
15.     S=S-{s_i};
16.     R=R+{s_i};
17. end for
END
```

5.4 选择算子

在选择阶段,选择染色体进行重组,以通过变异和交叉操作产生下一个种群。该选择基于轮盘方法。选择用于重组的任务的概率为1。

5.5 交叉算子

交叉操作是将两个染色体结合起来,以产生更高质量染色体。使用标准的单点交叉算子,通过在这点上交换基因,选择父母染色体中的单个点以产生新的个体。根据设计的编码策略,从任务执行位置和执行排序两个方面来考虑。假设有两个个体,它们表示的任务位置集合分别为 $X_1=\{x_1^0,x_1^1,\cdots,x_1^i,\cdots,x_1^{n-1}\}$ 和 $X_2=\{x_2^0,x_2^1,\cdots,x_2^i,\cdots,x_2^{n-1}\}$,经过交叉操作之后得到新的个体 X_{12} 和 X_{21} 的过程如图5所示。其算法伪代码如下所示。

算法3 位置交叉算法

```
BEGIN
01. 生成[0,n-1]之间的随机数 r
02. X_12=X_21=∅;
03. for m=0 to r do
04. X_12=X_12+x_1^m;
05. X_21=X_21+x_2^m;
06. end for
07. for m=r+1 to n-1 do
08.     X_12=X_12+x_2^m;
09.     X_21=X_21+x_1^m;
10. end for
END
```

首先,和位置交叉过程一样,任务排序交叉也需要通过生成随机数的方式设定一个交叉点。把任务排序集合的第一个任务到交叉点之间的任务序列称为匹配序列。假设有两个任务排序集合 Y_1 和 Y_2,然后将 Y_1 的匹配序列加到 Y_2 的前面,Y_2 的匹配序列加到 Y_1 前面,形成两个新的临时任务序列。最后,将两个新的临时任务序列中的重复任务移除,产生的任务序列为新个体。两个任务排序集合为 $Y_1=\{y_1^0,y_1^1,\cdots,y_1^i,\cdots,y_1^{n-1}\}$、$Y_2=\{y_2^0,y_2^1,\cdots,y_2^i,\cdots,y_2^{n-1}\}$,形成两个新个体 Y_{12} 和 Y_{21} 的过程如图6所示。算法伪代码如下所示。

算法4 排序交叉算法

```
BEGIN
01. 生成[0,n-1]之间的随机数 r
02. Y_12=Y_21=∅;
03. for m=0 to r do
04. Y_12=Y_12+y_1^m;
05. Y_21=Y_21+y_2^m;
06. end for
07. Y_12=Y_12+Y_2;
```

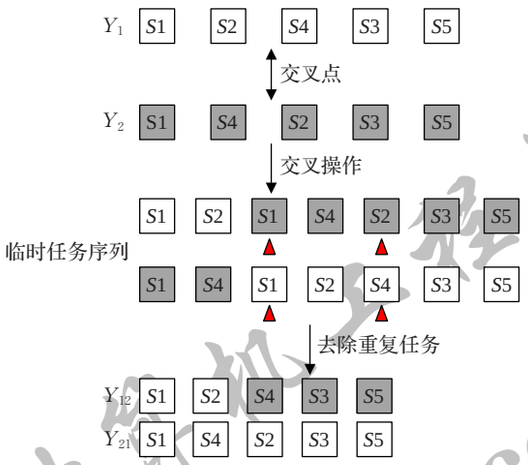

图6　任务执行排序单点交叉过程

08. $Y_{21}=Y_{21}+Y_1$;
09. 去除 Y_{12} 中的重复任务
10. 去除 Y_{21} 中的重复任务

END

5.6　变异算子

任务执行位置的变异如图7所示,每个基因只能为两个值0或1,随机选择一个基因,以突变概率 P_m 来决定其是否变异,如果产生变异,则该基因的值取反,即0变为1或1变为0,形成一个新的个体。

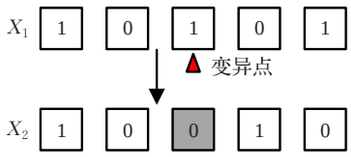

图7　任务执行位置变异过程

其算法伪代码如下所示。

算法5　位置变异算法

BEGIN

01. 生成 $[0,1]$ 之间的随机数 P_i
02. if　$P_i \leqslant P_m$　then
03. $x_i=|x_i-1|$;
04. end if

END

任务排序变异也要满足顺序限制关系,首先从集合 Y 中随机选择一个任务 y_i,任务集合中第一个任务和最后一个任务位置固定,所以变异任务不可能是第一个或最后一个任务。然后,对集合 Y 从左往右进行搜索,当搜索到某个任务 y_a 时,任务 y_i 的前驱任务都在集合 $\{y_0,y_1,\cdots,y_a\}$ 中,则停止搜索;同时,对集合 Y 从右往左进行搜索,当搜索到某个任务 y_b 时,任务 y_i 的后继任务都在集合 $\{y_b,y_{b+1},\cdots,y_{n-1}\}$ 中,则停止搜索。此时,任务 y_i 必定处于集合 $Y'=\{y_{a+1},y_{a+2},\cdots,y_{b-1}\}$ 中,而且任务 y_i 可以处于集合 Y' 中的任何一个位置。最后,任务 y_i 排除初始位置,随机选择一个新的位置进行插入操作。任务执行排序的变异过程如图8所示。

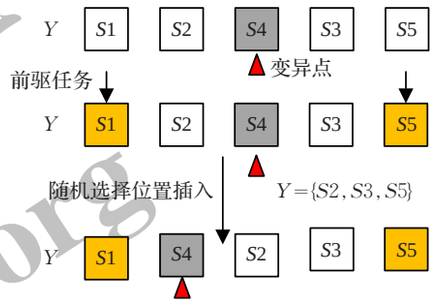

图8　任务执行排序变异过程

其算法伪代码如下所示。

算法6　排序变异算法

BEGIN

01. 生成 $[0,n-2]$ 之间的随机数 r;
02. for　$i=0$　to　$n-1$　do
03. 　　搜索任务 y_a,使得 $pre(y_r)\subset\{y_0,y_1,\cdots,y_a\}$;
04. end　for
05. for　$i=n-1$　to　0　do
06. 　　搜索任务 y_b,使得 $succ(y_r)\subset\{y_b,y_{b+1},\cdots,y_{n-1}\}$;
07. end　for
08. 得到集合 $Y'=\{y_{a+1},y_{a+2},\cdots,y_{b-1}\}$;
09. 排除 y_r 当前位置,在集合 Y' 中选择一个新的位置插入;
10. 新个体 $Y=\{y_0,y_1,\cdots,y_a\}+Y'+\{y_b,y_{b+1},\cdots,y_{n-1}\}$;

END

6　仿真实验

本章对提出的GA算法的性能进行仿真实验。采用随机生成工作流的方式,其中工作流中每个任务的工作量服从[1,100]之间均匀分布(单位为GHz);每个任务的输出数据大小服从[1,25]均匀分布(单位为Mb);并且,移动设备的计算能力、计算功率、数据发送功率、数据接收功率,以及边缘服务器计算能力如表1所示。除了GA算法,还分别仿真了Local Execution算法和RANDOM算法。

表1　移动设备和边缘服务器性能参数

设备	性能	参数
移动设备	计算能力	$c_M=1$ GHz
	计算功率	$P_M=0.5$ W
	发送数据功率	$P_{up}=0.1$ W
	接收数据功率	$P_{down}=0.15$ W
边缘服务器	计算能力	$c_E=3$ GHz

Local Execution算法:LOCAL算法就是将所有任务都放在移动设备上进行执行处理。

RANDOM算法:RANDOM算法就是对于任务的执

行位置和排序都采用随机的方式。

另外,在遗传算法中,交叉操作和变异操作的概率为1,种群中的个体数目为 $N=50$ 。在实验中,RANDOM算法和LOCAL算法设置的初始目标解的个数均为50。随机工作流的任务数为50。下面分别从工作流任务数变化、传输数据变化、工作量变化和迭代次数变化这四个方面,对GA算法的性能进行实验仿真。

6.1　任务数变化的影响

在本节主要研究GA算法、LOCAL算法和RANDOM算法在不同工作流任务数量时能耗和时间方面的比较,模拟任务数分别为50、100、150、200和250的情况,如图9所示,可以看到,随着任务数量的增加,工作流的适应度值明显增加。这是因为任务数量的增加会导致任务执行需要更多的执行时间和能耗,因此适应度值也会随之增加。而且,通过实验结果可以发现,GA算法得到的实验结果是最优的,体现了遗传算法在解决此类问题的优越性。

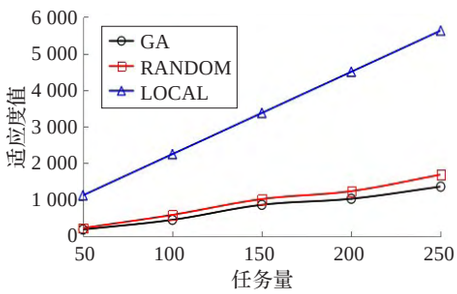

图9　任务数变化对适应度值的影响

6.2　传输数据变化的影响

本节仿真了工作流任务的输出数据大小在[5,25]时,三种算法在能耗和时间上的变化情况。如图10中所示,任务的输出数据大小和适应度值成正比。这是因为任务的输出数据大小对工作流执行过程中的数据接收和发送有较大的影响,在同样的调度策略下,较大的数据传输会产生较长的时间,会导致整个工作流的完工时间变长。同样,较多的数据传输会给移动设备带来更多的传输能耗,所以移动设备的能耗也增加了。但是,GA算法的最优解都要比LOCAL算法和RANDOM算法更优。所以,输出数据大小会对工作流的执行时间和移动设备的能耗造成一定的影响。

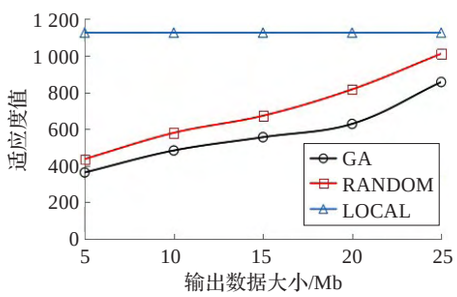

图10　传输数据大小变化对适应度值的影响

6.3　工作量变化的影响

图11研究了任务工作量大小变化对能耗与时间加权和的影响,其中仿真了工作量变化范围为[20,100]的情况。从图中可以看出,工作量越大,适应度值越大。这是因为更大的工作量带来更大任务执行时间,那么工作流的完工时间会变长。当任务在本地执行时,较大工作量的任务会消耗移动设备更多的能耗,所以在执行工作流时移动设备的能耗也增加了。同样,GA算法要比LOCAL算法和RANDOM算法可以得到更优的实验结果。

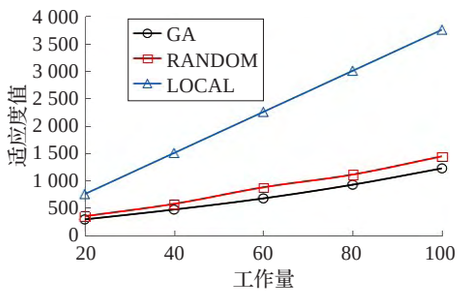

图11　工作量大小变化对适应度值的影响

6.4　迭代次数的影响

本节仿真了迭代次数在[10,100]时,迭代次数的变化对实验结果的影响。如图12所示,随着迭代次数的增加,适应度值呈下降趋势,且当迭代次数大于等于50时,适应值呈现收敛趋势。由此可以得出结论:迭代次数为50时,可以得到最优的实验结果。

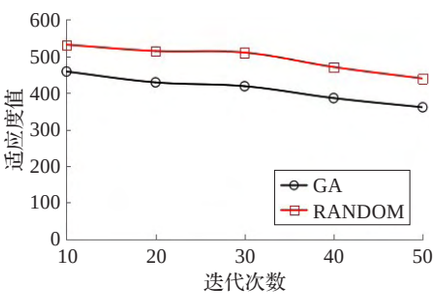

图12　迭代次数大小变化对适应度值的影响

7　总结与展望

本文针对移动边缘计算环境下工作流执行的时间和移动设备能耗问题,提出了基于遗传算法的工作流任务调度算法,设计了工作流任务调度位置和执行顺序的编码策略。基于该编码策略,在遗传算法的遗传操作中,分别对交叉和变异算子进行了详细的叙述。仿真结果表明,相对于RANDOM算法和Local Execution,遗传算法可以获得一个更好的实验结果。

在未来的工作中,将考虑多个移动设备与多个移动边缘服务器之间的任务调度问题,也可以考虑到移动边缘环境下,移动设备的移动性对能耗和时间的影响。

参考文献:

[1] Barbarossa S, Sardellitti S, Lorenzo P D.Communicating while computing: distributed mobile cloud computing over 5G heterogeneous networks[J].IEEE Signal Processing Magazine, 2014, 31(6): 45-55.

[2] Khan A U R, Othman M, Madani S A, et al.A survey of mobile cloud computing application models[J].IEEE Communications Surveys & Tutorials, 2014, 16(1): 393-413.

[3] Wang Y, Lin X, Pedram M.A Bayesian game formulation of power dissipation and response time minimization in a mobile cloud computing system[C]//IEEE Second International Conference on Mobile Services.[S.l.]: IEEE Computer Society, 2013: 7-14.

[4] Khazaei H, Misic J, Misic V B.Performance of cloud centers with high degree of virtualization under batch task arrivals[J].IEEE Transactions on Parallel & Distributed Systems, 2013, 24(12): 2429-2438.

[5] Kumar K, Lu Y H.Cloud computing for mobile users: can offloading computation save energy?[J].Computer, 2010, 43(4): 51-56.

[6] Chen Guangyu, Kang Byung-Tae, Kandemir M, et al. Studying energy trade offs in offloading computation/ compilation in Java-Enabled mobile devices[J].IEEE Transactions on Parallel and Distributed Systems, 2004, 15(9): 795-809.

[7] Chen X, Jiao L, Li W, et al.Efficient multi-user computation offloading for mobile-edge cloud computing[J].IEEE/ ACM Transactions on Networking, 2015, 24(5): 2795-2808.

[8] 周业茂,李忠金,葛季栋,等.移动云计算环境下基于延时传输机制的多目标工作流调度方法[J].软件学报,2018,29(11):1-20.

[9] 王晓军,熊潇.基于改进遗传算法的工作流调度研究[J].计算机技术与发展,2013,23(7):108-111.

[10] Hu Y C, Patel M, Sabella D, et al.Mobile edge computing—a key technology towards 5G[J].ETSI White Paper, 2015, 11(11): 1-16.

[11] Shi W, Dustdar S.The promise of edge computing[J].Computer, 2016, 49(5): 78-81.

[12] Salman O, Elhajj I, Kayssi A, et al.Edge computing enabling the Internet of Things[C]//Internet of Things, 2016: 603-608.

[13] 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.

[14] Wang S, Zhang X, Zhang Y, et al.A survey on mobile edge networks: convergence of computing, caching and communications[J].IEEE Access, 2017, 5(99): 6757-6779.

[15] Ahmed A, Ahmed E.A survey on mobile edge computing[C]//International Conference on Intelligent Systems and Control, 2016.

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

[17] Liu H, Eldarrat F, Alqahtani H, et al.Mobile edge cloud system: architectures, challenges, and approaches[J].IEEE Systems Journal, 2017, PP(99): 1-14.

[18] Beck M T, Werner M, Feld S, et al.Mobile edge computing: a taxonomy[C]//the Sixth International Conference on Advances in Future Internet, 2014: 48-54.

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

[20] 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.

[21] Ahmed E, Rehmani M H.Mobile edge computing: opportunities, solutions, and challenges[J].Future Generation Computer Systems, 2016, 70.

[22] Chen X, Jiao L, Li W, et al.Efficient multi-user computation offloading for mobile-edge cloud computing[J]. IEEE/ACM Transactions on Networking, 2015, 24(5): 2795-2808.