计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210100191

基于 PSO-GA 的多边缘负载均衡方法

姚泽玮　林嘉雯　胡俊钦　陈　星
福州大学数学与计算机科学学院　福州 350108
福建省网络计算与智能信息处理重点实验室(福州大学)　福州 350108
(zeweiy7052@163.com)

摘　要　移动边缘计算(Mobile Edge Computing,MEC)作为一种新的范式,可以解决移动设备的计算资源、存储资源短缺的问题。通过无线网络,它将移动设备上的密集型任务迁移到用户附近的边缘上执行,最后把运行结果传回给移动设备。由于用户移动的随机性,部署在城市的每个边缘的负载情况通常是不一致的。针对多边缘的负载均衡问题,考虑通过任务调度来最小化边缘集合中最大的任务响应时间,从而提高移动设备的性能。首先,对多边缘负载均衡问题进行形式化定义;其次,提出粒子群遗传算法(Particle Swarm Optimization-Genetic Algorithm,PSO-GA)来解决多边缘负载均衡问题;最后通过仿真实验,将该算法与随机迁移算法和贪心算法进行对比与分析。实验结果表明,PSO-GA 得到的结果最高分别优于随机迁移算法和贪心算法 51.58%和 26.34%。因此,PSO-GA 在缩短边缘的任务响应时间、改善用户体验方面具有较好的潜力。

关键词:移动边缘计算;任务响应时间;负载均衡;粒子群遗传算法
中图法分类号　TP301

PSO-GA Based Approach to Multi-edge Load Balancing

YAO Ze-wei,LIU Jia-wen,HU Jun-qin and CHEN Xing
College of Mathematics and Computer Science,Fuzhou University,Fuzhou 350108,China
Fujian Key Laboratory of Network Computing and Intelligent Information Processing (Fuzhou University),Fuzhou 350108,China

Abstract　As a new paradigm,mobile edge computing (MEC) can provide an efficient method to solve the computing and storage resource constraints of mobile devices. Through the wireless network,it migrates the intensive tasks on mobile devices to the edges near the users for execution,then the edges transmit the execution results back to mobile devices. Due to the randomness of users' movement,the load on each edge which deployed in the city is usually inconsistent. To solve the problem of multi-edge load balancing,the task scheduling is considered to minimize the maximum response time of tasks in the edge set,thereby improving the performance of mobile devices. Firstly,the multi-edge load balancing problem is formally defined. Then particle swarm optimization-genetic algorithm(PSO-GA) is proposed to solve the multi-edge load balancing problem. Finally,the performance of the algorithm is compared and analyzed with the random migration algorithm and the greedy algorithm through simulation experiments. The experimental results show that PSO-GA is superior to random migration and greedy algorithm by 51.58% and 26.34%,respectively. Therefore,PSO-GA has a better potential for reducing task response time of the edges and improving user experience.

Keywords　Mobile edge computing,Task response time,Load balancing,Particle swarm optimization-genetic algorithm

1　引言

近年来,随着移动通信技术和物联网的不断发展,移动设备的数量呈井喷式增长。通过移动设备上的各种应用,人们的生活越发便捷。但是,由于移动设备本身的计算能力、电池和数据存储容量有限,因此难以满足用户对低延迟与高可靠性的要求。而移动云计算[1](Mobile Cloud Computing,MCC)中的云具有丰富的计算资源和数据存储空间,并且能够高效地处理移动应用程序的任务请求,解决了便携式移动设备的计算资源受限和电池能耗的问题。然而,将移动设备上的任务迁移到远程云,会造成较长的通信延迟,从而降低用户的服务质量(Quality of Service,QoS)。

为了解决上述挑战,移动边缘计算[2-13]作为一种新的范式被提出。它将小型数据中心(即边缘)通过无线网络部署在靠近用户的网络边缘节点。在三层 MEC 框架中(分别为移动设备、边缘和远程云),边缘处于中间层,这样移动设备就可以通过无线网络以较低延时访问边缘,进而实现实时交互响应。此外,当边缘处于负载状态时,可以将延时容忍的任务迁移到远程云执行。边缘基础设施的部署方式与无线访问接入点相似。近年来,许多研究人员讨论如何在公共无线城域网

基金项目:国家自然科学基金项目(62072108);福建省自然科学基金杰青项目(2020J06014);福建省自然科学基金项目(2018J07005)
This work was supported by the National Natural Science Foundation of China(62072108),Natural Science Foundation of Fujian Province for Distinguished Young Scholars(2020J06014) and Natural Science Foundation of Fujian Province(2018J07005).
通信作者:林嘉雯(ljw@fzu.edu.cn)

中部署边缘,使得公众能享受边缘提供的高效服务[9-11]。因为大城市中的人口密度很大,所以不乏移动用户访问边缘。

但是,边缘亟需处理的一个问题是如何分配移动用户迁移的任务请求,使得边缘之间的负载能够达到均衡。通常的做法是将用户的任务请求分配给最近的边缘上去执行,但是当一个区域的用户数量过多时,对应的边缘负载就会过大。如果边缘上突然接受大量的用户任务请求,则会导致任务的响应时间变长,甚至导致任务执行失败。因此,有必要平衡每个边缘的负载,这样可以有效减少迁移任务的响应时间。

在很多研究中,都提出了能够解决分布式计算环境下的负载均衡问题的算法,其中最常见的属于启发式算法,如遗传算法[16]、粒子群优化算法[17]、人工神经网络[18]等。但是这些算法各有优缺点,例如遗传算法需要设置大量的参数,导致计算比较复杂,并且由于遗传算子之间的依赖性,因此收敛过程较慢。粒子群算法则容易陷入局部最优解而造成早熟收敛的现象。但是,可以将上述算法相结合,利用各自的优点来弥补彼此的不足。受文献[14]、文献[19]的启发,本文提出了一种混合粒子群遗传算法来解决多边缘的负载均衡问题。本文的主要贡献如下:

(1)使用二维的连续粒子来对边缘之间的任务调度方案进行编码;

(2)将遗传算法的变异算子引入到算法中,以解决粒子群算法陷入局部最优的问题;

(3)从边缘的平均任务响应时间和程序的运行时间方面对提出的算法进行评估,并与随机迁移算法和贪心算法进行比较、分析。

2 相关工作

随着移动通信技术的发展,5G 网络逐渐走进了人们的生活,它具有更高的带宽、更低的延时,并且可以大规模连接物联网。这为虚拟现实技术、无人驾驶与智慧城市等业务的发展提供了强有力支撑。虽然 5G 网络在数据传输速度上有极大的提升,但是由于移动设备本身的局限性,还是无法满足计算密集型任务的需求。作为云计算的一个分支,MCC 为移动设备提供了数据存储和计算服务,从而解决了移动设备的缺陷[1-2]。但是将移动设备上的任务迁移到离自身较远的云端上,将产生额外的网络延迟,当网络质量不好时,会大大降低移动用户的 QoS。

因此,Satyanarayanan 等[4]在 MEC 的基础上,提出了一个新的模型,称为边缘。边缘是一个受信任的计算机集群,部署在移动用户附近并提供了丰富的资源。它解决了移动设备的资源有限和无线网络中通信延时较长这两个基本问题。由于每个部署着边缘的区域的用户密度都不相同,因此到达每个边缘的任务请求数量也会有所不同。这可能会导致某些边缘的负载较大,而有些边缘的负载较小,从而导致资源的不合理分配。因此,边缘任务调度问题成为了一个主要的话题,受到了广泛学者的研究。

Wang 等[5]考虑了迁移任务和边缘的动态性,将边缘间的任务调度问题视为整数规划问题,并且提出了基于加权二部图模型的动态任务调度算法,结合 Kuhn Munkras(KM)搜索算法解决了二分图的匹配问题。Ramasubbareddy 等[6]提出多个边缘的环境中基于响应时间感知的任务调度,即通过一个边缘控制器计算每个边缘服务器的响应时间,如果有用户请求,则把具有最小响应时间的边缘服务器推送给用户。Somula 等[7]提出了一种基于用户到边缘的距离和边缘负载度感知的边缘负载均衡算法,用于找到最佳的边缘进行调度。Zhang[8]提出了一种在边缘环境下基于服务效用的公平传输调度算法,在每个时刻中从多个用户中选取服务效用值最大的用户进行任务调度。上述大多数研究通过选择目标边缘(从边缘的响应时间与负载水平等关键问题入手)来平衡边缘的负载,但是这些研究中的边缘都为独立的个体,并没有考虑到边缘之间的协同合作。

Jia 等[9]考虑了无线城域网中边缘的部署策略,提出了基于用户密度的放置算法,以此来优化移动用户和边缘之间的平均响应时间。虽然该研究关注边缘的负载均衡,但是没有考虑通过平衡边缘的负载来最大程度地缩短边缘集合的最大任务响应时间。Ma 等[10]和 Xu 等[11]分别提出了粒子群优化算法和贪心启发式算法,用于解决一样的边缘放置问题,从而缩短任务响应时间。尽管他们考虑了边缘处理用户迁移任务的平均响应时间,但是忽略了任务的排队时间。

Yao 等[12]将任务调度问题转化为整数线性规划问题,然后使用遗传算法去搜索最优解,以此在边缘负载均衡条件下尽可能缩短任务的平均响应时间。文献[13]采用遗传算法解决有优先关系约束的任务调度问题。遗传算法是大多数人解决 NP 问题首先考虑的解决办法之一,但是遗传算法计算复杂,收敛速度较慢。因此,本文在考虑了边缘之间的协同、任务的响应时间以及排队时间的前提下,提出了一种粒子群遗传算法,用于寻找边缘之间的任务调度方案,以最小化边缘集合中的最大任务响应时间。这样在保证移动用户 QoS 的同时,提升了边缘的资源使用率。

3 模型及问题定义

3.1 系统模型

假设边缘服务提供商已经在某个区域建立了 K 个边缘 $C=\{c_1,c_2,\cdots,c_K\}$,这些边缘与无线访问接入点均部署在固定的位置,边缘之间通过无线网络连接,且每个边缘至多与 E 个边缘相连。这样可以在相互连接的两个边缘间进行数据传输,而设备可以通过无线或蜂窝网络访问边缘,并使用其服务资源,如图 1 所示。假设应用程序在运行时可以被动态地任意划分成多个任务,并且可以将这些任务迁移到 K 个边缘中的任意一个执行(移动用户的任务会优先迁移到最近的边缘中)。

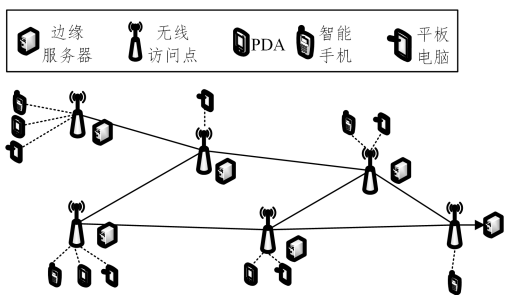

图 1 边缘部署模型

Fig. 1 Edge deployment model

由于边缘在为用户提供服务时符合排队现象,因此可以使用 $M/M/n$ 排队模型[15]对边缘进行建模。其中 $M/M/n$ 分

别代表用户迁移的任务流相继到达的时间间隔服从负指数分布、边缘上的任务服务时间服从负指数分布以及单个边缘拥有 n 台服务器。每个边缘 i 都由一个四元组表示：

$$c_i=\{n_i,\mu_i,\lambda_i,L_i\},i\in\{1,2,\cdots,K\} \tag{1}$$

其中，n_i 表示边缘 i 拥有的服务器数目；μ_i 表示边缘 i 每台服务器的服务速率；λ_i 表示边缘 i 的初始任务到达率，即单位时间内到达边缘的任务平均数(为了方便，之后将移动设备到达边缘的任务称为初始任务)；$L_i=\{c_{a_1},\cdots,c_{a_s}\},1\leqslant s\leqslant E$ 表示边缘 i 的邻接边缘集合，其中 c_{a_s} 表示在边缘集合 C 中与边缘 i 相连的第 a_s 个边缘。

假设每个边缘都可以将自身的一部分任务重定向到与其相连的任意边缘，使用 $f(i,j)$ 表示边缘 i 迁移到边缘 j 的任务流，并且边缘 i 只能考虑将自身的初始任务迁移到其邻接边缘。对于 $f(i,j)$，有以下约束：

$$\sum_{i=1}^{K}\sum_{j\in L_i}f(i,j)=\sum_{i=1}^{K}\sum_{j\in L_i}f(j,i) \tag{2}$$

$$\sum_{j\in L_i}f(i,j)\leqslant\lambda_i,\forall i\in\{1,\cdots,K\} \tag{3}$$

其中，式(2)表示边缘集合迁移的任务流总和等于边缘集合接收的任务流总和，即边缘集合的任务流守恒；式(3)表示边缘 i 发送的任务流之和必须小于等于自身的初始任务到达率 λ_i(这里忽略由边缘 i 接收的来自其邻接边缘的任务流)。

每个边缘 i 的平均任务等待时间 T_{wait}^{i} 由两部分组成，分别为任务排队时间和任务服务时间，其计算式如式(4)[20]所示：

$$T_{wait}^{i}(\lambda)=\frac{C(n_i,\frac{\lambda}{\mu_i})}{n_i\mu_i-\lambda}+\frac{1}{\mu_i} \tag{4}$$

其中，式(5)是 Erlang C 公式[15]，它根据边缘的服务器数量 n 和流量强度 ρ 来计算用户任务在边缘中不能马上接受处理而需要等待的概率。

$$C(n,\rho)=\frac{\left(\frac{(n\rho)^n}{n!}\right)\left(\frac{1}{1-\rho}\right)}{\sum_{k=0}^{n-1}\frac{(n\rho)^k}{k!}+\left(\frac{(n\rho)^n}{n!}\right)\left(\frac{1}{1-\rho}\right)} \tag{5}$$

因为边缘在重定向一部分任务流到邻接边缘时，需要通过无线网络进行传输，这将产生对应的网络延迟。假设所有迁移的任务可以被任意划分为大小相同的数据包，这样在一对边缘之间传输任意数据包都会产生一样的网络时延。边缘之间的网络延时矩阵使用 $D\in R^{K\times K}$ 来表示：

$$D=\begin{bmatrix}d_{11}&\cdots&d_{1K}\\\vdots&\ddots&\vdots\\d_{K1}&\cdots&d_{KK}\end{bmatrix} \tag{6}$$

其中，$d_{i,j}$ 表示单位任务从边缘 i 传输到边缘 j 所花费的网络延迟，且 $d_{i,j}=d_{j,i}$。特别地，当 $i=j$ 时，$d_{i,j}=0$；当边缘 i 与边缘 j 不存在连接时，$d_{i,j}=\infty$。

这样边缘 i 接收到的来自相邻边缘的重定向任务流 $f(j,i)$所产生的网络延时就等于 $f(j,i)\cdot d_{j,i}$。因此，每个边缘 i 接收重定向任务所需要的总网络延时的计算式如下：

$$T_{net}^{i}=\sum_{j\in L_i}f(j,i)\cdot d_{j,i},i\in\{1,2,\cdots,K\} \tag{7}$$

最后，使用 T_{total} 来表示在边缘 i 上执行所有任务的平均响应时间，它由边缘的任务等待时间和对应的网络延时组成，计算式如下：

$$T_{\text{total}}^{i}=T_{\text{wait}}^{i}(\bar{\lambda}_i)+T_{net}^{i} \tag{8}$$

其中，$\bar{\lambda}_i$ 表示边缘 i 的实际负载。由于边缘可能会将自身的一部分任务流重定向到邻接边缘，或者从邻接边缘接收一部分任务流，因此每个边缘的实际负载是动态变化的，在计算每个边缘对应的任务响应时间时，需要重新计算它的实际负载，如式(9)所示：

$$\bar{\lambda}_i=\lambda_i-\sum_{j\in L_i}f(i,j)+\sum_{j\in L_i}f(j,i) \tag{9}$$

相应地，邻接边缘集合 L_i 中的每个边缘的实际负载也要进行更新。边缘 j 的实际负载由边缘 i 迁移过来的任务流以及自身重定向到邻接边缘的任务流决定，如式(10)所示：

$$\bar{\lambda}_j=\lambda_j+f(i,j)-\sum_{k\in L_j}f(j,k),j\in L_i \tag{10}$$

表 1 问题定义的相关符号

Table 1 Symbols related to problem definition

符号	定义
K	部署的边缘个数
C	边缘集合
c_i	边缘节点
n_i	边缘 i 的服务器数目
μ_i	边缘 i 的服务器速率
λ_i	边缘 i 的初始任务到达率
L_i	边缘 i 的邻接边缘集合
d_{ij}	边缘 i 与边缘 j 的单位任务网络延时
T_{wait}^{i}	边缘 i 的任务等待时间
T_{net}^{i}	边缘 i 接收任务流产生的总网络延时
T_{total}^{i}	边缘 i 的任务响应时间
f	边缘之间传输的任务量

3.2 问题定义

因此，将多边缘的负载均衡问题定义为：在城市中的 K 个固定位置上分别部署边缘，每个边缘 i 上拥有 n_i 台服务器，服务器的服务速率为 μ_i，并且边缘 i 上的初始任务到达率为 λ_i，记为 c_i。我们的目标是能够找到一组任务流 $f=\{f(i,j)\mid i\in\{1,\cdots,K\},j\in L_i\}$(该任务流必须满足式(2)、式(3)的要求)，所有边缘通过该组任务流进行调度，使得所有边缘的负载处于一个相对平衡的状态，进而最小化边缘集合中的最大任务响应时间 T_{total}^{i}，如式(11)所示：

$$\min\max_{f}T_{\text{total}}^{i} \tag{11}$$

4 粒子群算法与遗传算法

4.1 粒子群算法

粒子群优化(Particle Swarm Optimization，PSO)算法是由 Eberhart 博士与 Kennedy 博士通过研究鸟类的捕食行为而提出的一种群智能算法[16]。鸟群寻找食物是基于群体中各个成员之间的信息共享和协同合作来实现的。PSO 算法的实现相对简单且无需调整许多参数。目前，它已经广泛用于解决优化问题、训练神经网络等方面。

PSO 算法设计一群无质量的粒子来模拟鸟群的行为，每一个粒子都有两个重要的属性，即位置 X 和速度 V。其中，粒子的位置代表待求解问题在搜索空间的一个可行解，而粒子的速度则代表粒子所要飞行的速率与方位。每个粒子都会根据个体极值 *pbest* 与全局极值 *gbest* 来调整自身的位置，以此不断接近搜索空间中的最优位置。其中，*pbest* 是粒子本身到目前为止发现的最佳位置；*gbest* 则是粒子群到目前发现的最佳位置。另外，每个粒子都根据自己的当前位置计算对应的适应度值，并用该值来评估个体极值 *pbest* 与全局极值 *gbest*。对粒子群进行迭代操作(主要是计算粒子的速度与位

置、更新两个极值),最后所有粒子都会向最优的位置聚集,从而找到待求解问题的最优解。

4.2 遗传算法

遗传算法(Genetic Algorithm,GA)是一种进化算法,最早由美国密歇根大学的 John Holland 教授提出[17]。该算法是以生物学家达尔文的进化论为理论基础,通过模拟自然界中优胜劣汰的生存过程和遗传学中生物的进化过程来寻找最优解的方法。遗传算法具有隐式并行性与可扩展性,适用于组合优化问题、机器学习等领域。

在使用遗传算法的过程中,需要将待求解问题映射成数学模型,即将待求解问题的可行解编码成染色体(也表示个体)。一个可行解包含的元素等同于染色体上所拥有的基因,并且多个染色体组成一个种群。

种群的进化需要依靠遗传学的遗传算子。1)选择算子:它从每一代种群中选择优秀的个体(通过适应度函数可以评估种群中每一个个体的优劣程度),并淘汰掉劣势的个体,从而发挥自然选择的作用。这样在随机初始化种群后,根据选优淘劣的自然规律,每代都可以进化出更好的个体。2)交叉算子:它将两个父代个体中的遗传基因进行部分交叉互换来创造出新的个体,这样遗传算法的寻优能力可以取得显著提升。3)变异算子:它是改动染色体上的某些基因值,这样遗传算法就可以跳出局部最优解。在对种群迭代完成后,种群中适应度值最小或最大(根据问题要求)的个体就是待求解问题的最优解。

5 多边缘负载均衡算法

5.1 引入变异算子的粒子群算法

PSO 算法与 GA 算法普遍用于组合优化问题,但是 PSO 算法较易陷入局部最优,而 GA 算法花费了较长运行时间。为了结合两者的优势,将遗传算法的变异算子引入粒子群优化算法中,以解决多边缘负载均衡问题。这样可以使算法具有较快的收敛速度,又不容易陷入局部最优解[19]。

5.1.1 解的表示

这里,问题域中的解既相当于 PSO 算法中粒子的位置,又相当于 GA 算法中的染色体(以下统称为粒子)。由于要表达边缘之间的任务流的调度情况,可以借鉴有向带权图的邻接矩阵的思想,通过使用二维矩阵 $f^{K\times K}$ 来对粒子进行编码。图中的权值就相当于边缘之间传输的任务量。用 X_t^k 表示第 t 次迭代中种群的第 k 个粒子,如式(12)所示:

$$X_t^k=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1K} \\ x_{21} & x_{22} & \cdots & x_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ x_{K1} & x_{K2} & \cdots & x_{KK} \end{bmatrix} \tag{12}$$

其中,$x_{i,j}$ 表示从边缘 i 到边缘 j 的任务流。特别地,如果边缘 i 和边缘 j 之间不存在连接,则 $x_{i,j}=0$,并且有 $x_{i,i}=0$。对于任务流矩阵 $\boldsymbol{X}$,有以下限制:

$$\sum_{j\in L_i} x_{i,j}\leqslant\lambda_i,\ \forall\, i\in\{1,2,\cdots,K\} \tag{13}$$

$$\sum_{i\in L_j} x_{i,j}<n_j\mu_j-\lambda_j,\ \forall\, j\in\{1,2,\cdots,K\} \tag{14}$$

式(13)表示边缘集合中的任意一个边缘 i 重定向到其邻接边缘集合的任务流之和都不能超过其初始任务到达率 λ_i;式(14)表示边缘集合中的任意一个边缘 j 接收的总任务流必须小于边缘服务器的总服务速率减去本身的初始任务到达率 λ_j。

5.1.2 适应度函数

选取合适的适应度函数 F,用于评估每个粒子的优劣程度。本文的目标是通过平衡边缘集合的负载,来最小化边缘集合中的最大任务响应时间,因此可以使用式(15)作为适应度函数。

$$F(X)=\max(T_{\text{total}}^i),i\in\{1,2,\cdots,K\} \tag{15}$$

计算粒子的适应度值的流程如下:1)对于 X_t^k,使用式(9)、式(10)来更新每个边缘及其邻接边缘的任务到达率 λ;2)计算每个边缘的任务等待时间 $T_{\text{wait}}^i(\bar{\lambda}_i)$ 和对应的网络延时 T_{net}^i;3)选取此时边缘集合中的最大任务响应时间作为该个体的适应度值。

如果粒子的适应度值越小,边缘集合对应的最大任务响应时间就越短。因此,当执行完种群的迭代,就可以把适应度值最小的粒子作为 K 个边缘的最优任务调度方案。

5.1.3 种群初始化

首先确定 PSO-GA 的种群大小和迭代次数 P_{num} 和 N,然后通过均匀分布随机生成初始种群 $G_1=\{X_1^1,X_1^2,\cdots,X_1^{P_{num}}\}$ 来进行迭代。

粒子的位置 X:对于粒子位置的第 i 行 $X_1^k(i)$(对应于边缘集合的第 i 个边缘),根据均匀分布 $uniform(0,\lambda_i)$ 选取随机数进行填充。如果随机生成的粒子 X_1^k 违反了式(13)、式(14),则选取 X_1^k 的相关行或列的值进行随机减小,直到该粒子的位置符合要求。

粒子的速度 V:对于粒子速度的第 i 行 $V_1^k(i)$,使用均匀分布 $uniform(-\lambda_i\cdot 0.1,\lambda_i\cdot 0.1)$ 来进行指定。这里,均匀分布的范围选择边缘 i 初始任务到达率的 10%。

5.1.4 粒子位置与速度的更新

在种群的迭代过程中,需要不断更新粒子的速度和位置,以此来搜索解空间中的最优解,更新方式如下:

$$V_{t+1}^k=\omega V_t^k+c_1r_1(pbest^k-X_t^k)+c_2r_2(gbest-X_t^k) \tag{16}$$

$$X_{t+1}^k=X_t^k+V_{t+1}^k \tag{17}$$

其中,式(16)为粒子的速度更新公式,而 c_1 和 c_2 为种群的学习因子(通常取 2),r_1 和 r_2 是介于(0,1)之间的随机数,ω 为惯性权重。此外,式(16)的 ωV_t^k 为粒子的“惯性部分”,表示粒子的上一次迭代的速度 V_t^k 对本次迭代的速度 V_{t+1}^k 大小和方向的影响;$c_1r_1(pbest^k-X_t^k)$ 记为粒子的“自身认知项”,是粒子从当前位置到自身最优位置的一个矢量,表示粒子的移动部分是依据自己之前的经验;$c_2r_2(gbest-X_t^k)$ 记为“群体认知项”,是粒子从当前位置到种群中最优位置的矢量,体现了粒子之间的协作与经验共享。式(17)则表示在解空间中搜索时,每个粒子的位置更新方式。

为了使本文算法在迭代初期具有较强的全局寻优能力,在解空间中能尽可能探索更多的情况;而在迭代后期,算法有更好的局部寻优能力,能更快地逼近最优解。本文在式(16)中增加了惯性权重线性递减策略,这样可以调整算法的全局和局部搜寻能力,以大大提升其性能,具体表达式如下:

$$\omega=\omega_{\max}-t\cdot(\omega_{\max}-\omega_{\min})/N,\omega\in[\omega_{\min},\omega_{\max}] \tag{18}$$

在计算完粒子的位置与速度后,需要更新粒子的个体最优位置 pbest、种群最优位置 gbest。式(19)表示获取第 k 个粒子自第 t 次迭代以来的自身最优位置,arg min 函数用于返回从第 1 次到第 t 次迭代具有最小适应度值的自身位置 X。式(20)则表示在第 t 次迭代时种群中的最优位置,使用

arg min 函数取得种群中适应度值最小的粒子对应的个体极值 *pbest*。

$$pbest^k = \arg\min_{X}(F(X_1^k), F(X_2^k), \cdots, F(X_t^k)) \quad (19)$$

$$gbest = \arg\min_{pbest}(F(pbest^k), k = 1, 2, \cdots, P) \quad (20)$$

5.1.5 变异算子

本文考虑将 GA 算法中的变异算子引入到 PSO 算法中。在每次迭代过程中，都会根据变异概率 R_{mut} 从种群中随机选取一部分粒子。变异操作的主要流程如下：1)对于第 i 个边缘(对应到粒子位置的第 i 行 $X(i)$)，先获取其初始任务到达率和邻接边缘集合；2)从邻接边缘集合随机选取 e 个边缘，记为 *rand_list*；3)通过 *rand_list* 中边缘的下标确定粒子第 i 行 $X(i)$需要进行操作的位置；4)根据边缘 i 的初始任务到达率对这些位置上的值进行重新的初始化。

变异算子操作过程如算法 1 所示。

算法 1 粒子的变异算法

输入：X，C

输出：X

1. for each i in C do
2. 获取边缘 i 的 λ_i 和 L_i；
3. 从 L_i 中随机选取 e 个边缘，记为 rand_list；
4. for each j in rand_list do
5. X[i][j] = uniform(0, λ_i)；
6. end for
7. end for

5.1. 算法流程

PSO-GA 的算法流程如下：

(1)初始化算法的主要参数，包括边缘的个数 K，种群的大小 P_{num}，种群迭代次数 N，粒子的变异概率 R_{mut}，惯性权重 ω 以及学习因子 c_1 和 c_2 等；

(2)生成初始迭代种群 G，并计算每个粒子的适应度值。对于每个粒子的个体极值都设置为粒子当前的位置，种群的全局极值设置为所有粒子中适应度值最小的粒子位置；

(3)先根据式(18)，计算惯性权重 ω，接着使用式(16)、式(17)更新粒子的速度与位置，然后检查其合法性(粒子的速度与位置的值可能超出边界范围)；

(4)对种群 G 中的部分粒子根据算法 1 进行变异操作；

(5)重新计算所有粒子的适应度值，并根据式(19)、式(20)更新粒子的 pbest 以及种群的 gbest；

(6)判断算法是否达到终止条件，即是否执行到最大迭代次数 N。若满足条件，则返回种群的全局极值 gbest 作为最优的任务调度方案；否则返回步骤 3。

基于 PSO-GA 的多边缘负载均衡算法的流程如算法 2 所示。

算法 2 PSO-GA 算法

输入：N，P_{num}，R_{mut}，C

输出：gbest

1. 初始化 P_{num} 个粒子作为初始种群 G；
2. 计算所有粒子的适应度值并更新个体极值 pbest 和全局极值 gbest；
3. for n=1 to n=N do
4. 计算惯性权重 ω；
5. for each X in G do
6. 更新所有粒子的速度 V 和位置 X；
7. 检查粒子速度 V 和位置 X 的合法性；
8. end for
9. for each X in G do
10. if rand()<R_{mut} then
11. 调用算法 1；
12. end if
13. end for
14. 计算种群 G 中所有粒子的适应度值；
15. 更新每个粒子的个体极值 pbest；
16. 更新种群 G 的全局极值 gbest；
17. end for
18. 返回全局极值 gbest.

5.2 随机迁移算法

本文使用了两种对比算法，其中一个是随机迁移算法(Random Migration Algorithm，RMA)。该策略随机确定每个边缘迁移的任务量范围，以及迁移的任务量。

算法 3 的主要流程如下：1)初始化边缘集合及算法参数；2)随机生成 N_{rma} 个任务流矩阵 $\boldsymbol{s}$，并进行合法性检查；3)使用任务流矩阵 $\boldsymbol{s}$ 计算边缘集合的最大任务响应时间，最后选出具有最小任务响应时间的任务调度方案 $\boldsymbol{s}$。

随机迁移算法的具体流程如算法 3 所示。

算法 3 随机迁移算法

输入：N_{rma}，C

输出：s

1. for n=1 to n=N_{rma} do
2. 生成零矩阵 s∈$R^{K\times K}$；
3. 生成一个随机数 ratio∈[0,1]；
4. for each i in C do
5. 获取边缘 i 的 L_i 并设置其迁移的任务量范围 arr=λ_i · ratio；
6. for each j in L_i do
7. s[i][j]=uniform(0,arr)；
8. end for
9. end for
10. 检查方案 s 的合法性并保存到 solutions 集合；
11. end for
12. 计算 solutions 中每个方案对应的最大任务响应时间；
13. 返回具有最小任务响应时间的方案 **s**。

5.3 贪心算法

另一种算法为贪心算法(Greedy Algorithm)。贪心算法对问题进行求解时，每一步都选择当前的最优方案，以期望得到全局最优解。在本文的贪心策略中，每次选取任务响应时间最大的边缘进行任务迁移，边缘的任务响应时间越长，其优先级越高。

贪心算法的具体流程如下：

(1)将边缘集合及所需参数进行初始化；

(2)选取一个划分指标 rt，并计算边缘集合的任务响应时间以及剩余可接收的任务量；

(3)若边缘集合的任务响应时间均小于划分指标 rt，则跳转到步骤 7；否则获取集合 *rt_set* 中任务响应时间最大的边缘对应的下标 i；

(4)获取边缘 i 的邻接边缘集合及该集合对应的任务响应时间，然后获取该集合中任务响应时间最小的边缘对应的下标 j；

(5)计算边缘 i 迁移的任务量 *delta*，若边缘 j 剩余可接收

的任务量小于 *delta*,则跳转到步骤 7,否则继续执行;

(6)将任务从边缘 i 迁移到边缘 j,并更新边缘集合的任务响应时间以及剩余可接收的任务量,并回到步骤 3;

(7)保存当前的调度方案到 *solutions*。若达到终止条件,则从 *solutions* 中返回最优的调度方案;否则回到步骤 2。

贪心算法的具体过程如算法 4 所示。

算法 4 贪心算法

输入:C,rt_min,rt_max

输出:s

1. rt=rt_min;
2. while rt≤rt_max do
3. 生成零矩阵 $\mathbf{s}\in R^{K\times K}$;
4. 计算边缘集合的任务响应时间,记为 rt_set;
5. 计算每个边缘剩余可接收任务量,记为 lefts;
6. while True do
7. if each val∈rt_set≤rt do
8. break;
9. end if
10. 设置 i=集合 rt_set 中最大值对应的边缘的下标;
11. 获取边缘 i 的 L_i 及 L_i 对应的任务响应时间集合,记为 l_rt_set;
12. 设置 j=集合 l_rt_set 中最小值对应的边缘的下标;
13. 设置迁移的任务量 delta=边缘 i 总服务器速率的 3%;
14. if left[j]<delta then
15. break;
16. end if
17. s[i][j]+=delta;
18. 重新计算边缘 i,j 的任务响应时间;
19. 更新集合 rt_set,lefts;
20. end while
21. 保存方案 s 到 solutions 集合;
22. rt=rt+0.1;
23. end while
24. 计算 solutions 中每个方案对应的最大任务响应时间;
25. 返回具有最小任务响应时间的方案 s。

6 实验与仿真

本节通过模拟和仿真实验对本文所提出的 PSO-GA 进行性能评估。

PSO-GA 和对比算法均在 Python3.7 的环境下实现,并在搭载了 3.00GHz Intel(R) Core(TM) i5-9500 CPU 和 8GiB RAM 的系统上运行。

PSO-GA 的参数设置如下:$P_{num}=100$,$N=1\,000$,$R_{mut}=0.05$,$c_1=c_2=2$,$\omega_{\min}=0.3$,$\omega_{\max}=0.8$。对比算法的主要参数为:$N_{rma}=1000$,$min\_rt=1.5$,$max\_rt=3.5$。

6.1 实验环境

与文献[20]一样,假设边缘之间的网络延时与它们之间的物理距离成正比。各个边缘的任务到达率、服务器速率和边缘之间的网络延时均服从正态分布(注意边缘的任务到达率不能超过边缘总的服务器速率,否则会造成边缘的排队等待时间无限延长),边缘的服务器数则采用泊松分布,如表 2 所列。

表 2 边缘参数的设置

Table 2 Setting of edge's parameters

名称	分布
服务器数 n	$Poisson(3)$
服务器速率 μ	$N(5,2)>0$
任务到达率 λ	$0<N(15,6)<\mu\cdot n-0.25$
网络延时 d	$0.1\leqslant N(0.15,0.05)\leqslant 0.2$

本文中考虑边缘个数 $K=15$ 的情况,并在 5 个场景下进行算法的对比实验。每个场景的边缘拓扑图,均选取部分北京市的联通 5G 基站的真实坐标,然后根据就近原则,限制每个节点至多与 3 个节点相连。具体边缘拓扑图如图 2 所示,其中边上的权值表示节点之间的单位网络延时。

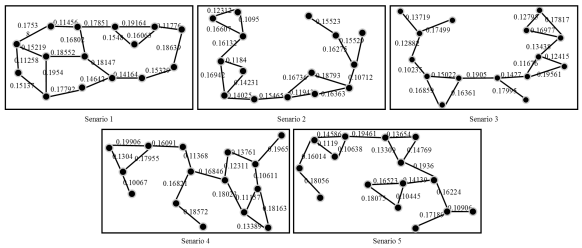

图 2 边缘的拓扑图

Fig. 2 Topological diagram of the edges

6.2 实验结果

为了比较 PSO-GA、随机迁移算法与贪心算法的任务调度性能,在 5 种不同的场景下,分别进行 10 组独立重复实验,然后从任务响应时间和程序执行时间两方面进行分析。

6.2.1 任务响应时间

表 2 列出了 PSO-GA、随机迁移算法和贪心算法在任务调度前后边缘集合的最大任务响应时间。

表 2 不同场景下 3 种算法的最大任务响应时间

Table 2 Maximum task response time of three algorithms in different senarios

	None	PSO-GA	RMA	Greedy
场景 1	7.821 18	1.373 44	2.081 86	1.497 25
场景 2	4.041 87	1.305 73	1.892 21	1.474 92
场景 3	12.279 59	2.268 75	3.001 9	2.866 33
场景 4	37.787 52	2.331 12	3.301 56	2.677 57
场景 5	13.115 69	1.964 19	2.865 55	2.341 85

图 3 更为直观地反映了在 5 种场景下 3 种算法所能达到的最小任务响应时间。可以看出,本文算法能够求得比另外两种算法更短的任务响应时间。这是因为 PSO-GA 算法采用了“开发”和“探索”机制,在种群迭代初期,算法会更注重随机探索,以找到较好可行解的大致区间。然后在种群迭代后期,则注重种群的开发,即算法会不断地朝着搜索空间中的最优解进行收敛。此外,PSO-GA 的结果优于 RMA 的比例最高达到 51.58%,且 PSO-GA 的结果优于 Greedy 的比例最高达到 26.39%。

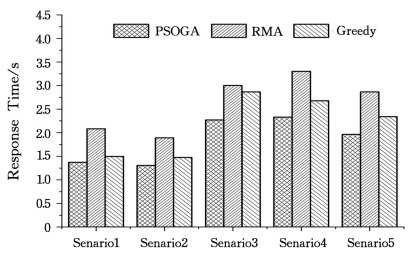

图 3 不同算法的最大任务响应时间

Fig. 3 Maximum task response time of different algorithm

对于贪心算法,它会自适应地选择一个较好的划分指标

rt。该算法得到的结果更多的是取决于实验中所设置的划分指标 rt。如果划分指标设置得过高，算法得到的任务响应时间会继续增加；如果设置得过低，则可能会找不到对应的调度方案。

而对于随机迁移算法，它计算出的任务响应时间有较大的不确定性，这是因为随机迁移算法依靠随机生成的解集。虽然随机迁移算法可以在一定程度上缩小最大的任务响应时间，但是效果却没有 PSO-GA、贪心算法的好。

6.2.2 程序执行时间

图 4 给出了在 5 种场景中 PSO-GA、随机迁移和贪心算法需要运行的时间。可以看出，与随机迁移和贪婪算法相比，PSO-GA 花费的时间最长。这是因为 PSO-GA 需要执行多次的迭代和计算操作，其中包含粒子的速度、位置更新与变异操作。而随机迁移算法没有 PSO-GA 类似的复杂计算，时间更多地花费在可行解的生成以及可行解的最大任务响应时间的计算上，因此其运行时间相比 PSO-GA 算法更小。对于贪心算法，它每次将任务响应时间最大的边缘进行任务迁移，并能够在较少的次数内达到终止条件而不需要进行迭代操作，所以花费的时间最短。

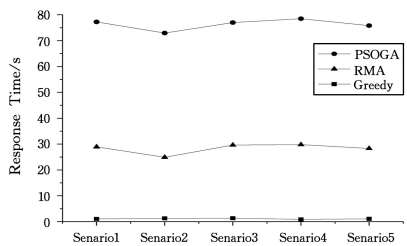

图 4 不同算法的运行时间

Fig. 4 Processing time of different algorithms

因此，贪心算法运行速度较快，可以适合用于有大量延迟敏感类型任务的边缘集合；而 PSO-GA 可以用于有更多延迟容忍类型任务的边缘集合。

7 讨论

在实际场景中，如果两个边缘相距太远，它们之间的传输延迟会很大，因此距离较远的边缘往往是不互相连接的。为了更加符合实际场景，本文设定 K 个边缘组成的网络拓扑结构中边缘之间是部分连通的。当用户通过无线网络访问边缘资源，由于不同区域的用户密度不同，会导致每个边缘节点上的负载不一致。通过合理地调度边缘的任务，可以使得边缘集合的负载相对均衡，进而优化整个边缘网络的最大任务响应时间。

通过 6.2.1 节中的实验表明，PSO-GA 能够为部分连通的边缘拓扑结构提供较好的任务调度方案。本文只考虑将边缘的初始任务迁移到与其邻接的边缘节点上执行，其邻接边缘节点不能把这部分任务进行再次迁移，即只考虑了边缘任务的单跳计算。为了进一步优化 PSO-GA 算法的结果，在未来的工作中，我们将考虑边缘任务的多跳计算，即允许边缘的初始任务进行多次迁移，到达处理能力更强的边缘节点来缩短任务的响应时间。

在 6.2.2 节的实验中，可以得知 PSO-GA 需要花费较长的时间来获得合理的任务调度方案。因此，为了使 PSO-GA 算法更好地应用于即时交互环境中，该算法运行效率的优化也考虑作为未来工作之一。考虑结合深度学习的算法，训练出一个能够应用于实际环境中的模型，让边缘节点能够进行实时的调度决策。

结束语 为了保证使用边缘服务的移动用户有较好的 QoS，并减少无线网络的时延，提高边缘的资源利用率，本文提出了一种基于 PSO-GA 的多边缘负载均衡算法。该算法将适合于求解 NP 问题的粒子群优化算法与遗传算法相结合，进而提高算法的搜索能力和运行效率，最终得到满足条件的最优边缘任务调度方案。最后，通过仿真实验证明了所提算法的有效性。实验结果表明，与随机迁移算法和贪心算法相比，PSO-GA 算法可以获得更小的任务响应时间。

参 考 文 献

[1] FAN X, CAO J, MAO H. A survey of mobile cloud computing [J]. zTE Communications, 2011, 9(1): 4-8.

[2] 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.

[3] 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.

[4] SATYANARAYANAN M, BAHL P, CÁCERES R, et al. The Case for VM-Based Cloudlets in Mobile Computing [J]. Pervasive Computing, IEEE, 2009, 8(4): 14-23.

[5] WANG T, WEI X, LIANG T, et al. Dynamic tasks scheduling based on weighted bi-graph in Mobile Cloud Computing [J]. Sustainable Computing: Informatics and Systems, 2018, 19: 214-222.

[6] RAMASUBBAREDDY S, SASIKALA R. RTTSMCE: a response time aware task scheduling in multi-cloudlet environment [J]. International Journal of Computers and Applications, 2019 (1): 1-6.

[7] SOMULA R, SASIKALA R. A load and distance aware edge selection strategy in multi-cloudlet environment [J]. International Journal of Grid and High Performance Computing (IJGHPC), 2019, 11(2): 85-102.

[8] ZHANG Q. Research on Task Offloading Technology in Mobile Cloud Computing [D]. Harbin: Harbin Institute of Technology, 2016.

[9] JIA M, CAO J, LIANG W. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks [J]. IEEE Transactions on Cloud Computing, 2015, 5(4): 725-737.

[10] MA L, WU J, CHEN L, et al. Fast algorithms for capacitated cloudlet placements [C]// 2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD). IEEE, 2017: 439-444.

[11] XU Z, LIANG W, XU W, et al. Capacitated cloudlet placements in wireless metropolitan area networks [C]// 2015 IEEE 40th Conference on Local Computer Networks (LCN). IEEE, 2015: 570-578.

[12] YAO D, GUI L, HOU F, et al. Load balancing oriented computation offloading in mobile cloudlet [C]// 2017 IEEE 86th Vehicular Technology Conference (VTC-Fall). IEEE, 2017: 1-6.

[13] OMARA F A,ARAFA M M. Genetic algorithms for task scheduling problem[J]. Journal of Parallel and Distributed Computing,2010,70(1):13-22.
[14] GONG Y G. Research on The Combination of Particle Swarm Optimization and Genetic Algorithm[D]. Guangzhou:Sun Yat-sen University,2007.
[15] KLEINROCK L. Queueing systems,volume 2:Computer applications[M]. New York:wiley,1976.
[16] KENNEDY J,EBERHART R. Particle swarm optimization [C]// Proceedings of ICNN'95-International Conference on Neural Networks. IEEE,1995:1942-1948.
[17] SRINIVAS M,PATNAIK L M. Genetic algorithms:a survey [J]. Computer,1994,27(6):17-26.
[18] PRIDDY K L,KELLER P E. Artificial neural networks:an introduction[M]. SPIE Press,2005.
[19] AGARWAL M,SRIVASTAVA G M S. Genetic algorithm-enabled particle swarm optimization (PSOGA)-based task scheduling in cloud computing environment[J]. International Journal of Information Technology & Decision Making,2018,17(4):1237-1267.
[20] JIA M,LIANG W,XU Z,et al. Qos-aware cloudlet load balancing in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing,2018,8(2):623-634.

YAO Ze-wei, born in 1998, postgraduate. His main research interests include computation offloading, computational intelligence and its applications.

LIN Jia-wen, born in 1985, Ph. D, lecturer, postgraduate supervisor, is a member of CCF. Her main research interests include intelligent information processing, computer vision and medical image analysis.

(上接第 451 页)

[2] HART Field Communication Protocol Specification,Revision 7.7[OL]. https://www.fieldcommgroup.org.
[3] Wireless Systems for Industrial Automation:Process Control and Related Applications[S]. ISA-100.11a-2009 Standard,2009.
[4] IEEE Standard for Low-Rate Wireless Personal Area Networks (LRWPANs) [S]. IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011),April 2016.
[5] COSTA R,LAU J,PORTUGAL P,et al. Handling Real-Time Communication in Infrastructured IEEE 802.11 Wireless Networks:The RT-WiFi Approach[J]. Journal of Communications and Networks,2019,21(3):319-334.
[6] IEEE Standard for Low-Rate Wireless Personal Area Networks (LR-WPANs)[S]. IEEE Std 802.15.4-2006,2006.
[7] Bluetooth Core Specification Version 4.2[OL]. https://www.bluetooth.com/specifications.
[8] Bluetooth Core Specification Version:5.2[OL]. https://www.bluetooth.com/specifications.
[9] KARALIS A,ZORBAS D,DOULIGERIS C. Collision-Free Advertisement Scheduling for IEEE 802.15.4-TSCH Networks [J]. Sensors,2019,19(8):1-22.
[10] ZORBAS D,PAPADOPOULOS G Z,DOULIGERIS C. Local or Global Radio Channel Blacklisting for IEEE 802.15.4-TSCH Networks? [C]// IEEE International Conference on Communications. IEEE,2018:1-6.
[11] LEENTVAAR K,FLINT J. The capture effect in FM receivers [J]. IEEE Transactions on Communications,1976,24(5):531-539.
[12] FERRARI F,ZIMMERLING M,THIELE L,et al. Efficient network flooding and time synchronization with glossy [C]// Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks. IEEE,2011:73-84.
[13] ZIMMERLING M,MOTTOLA L,KUMAR P,et al. Adaptive real-time communication for wireless cyber-physical systems [J]. ACM Transactions on Cyber-Physical Systems,2017,1(2):8:1-29.
[14] BOR M,VIDLER J E,ROEDIG U. Lora for the Internet of Things [C]// Proceedings of the 2016 International Conference on Embedded Wireless Systems and Networks. TU Graz,2016:361-366.
[15] STM32F103xC STM32F103xD STM32F103xE Datasheet production data[OL]. https://www.st.com.
[16] nRF24L01+Single Chip 2.4GHz Transceiver Product Specification v1.0[OL]. https://infocenter.nordicsemi.com.
[17] nRF52840 Product Specification v1.1[OL]. https://infocenter.nordicsemi.com.

QIAN Guang-ming, born in 1963, professo. His main research interests include wireless networks for embedded and real-time systems, etc.

YI Chao, born in 1995, graduate student. His main research interests include real-time wireless network.