基于改进人工蜂群算法的边缘服务器部署策略

李 波 袁 也* 侯 鹏 丁洪伟
(云南大学信息学院 云南 昆明 650500)

摘 要 作为移动边缘计算架构部署的第一步,边缘服务器的部署是基础和关键,其部署位置与用户体验和系统性能密切相关,但是目前较少有研究关注该问题。研究无线城域网中移动边缘计算环境下的边缘服务器部署问题,以最小化响应时间为目标,将边缘服务器部署问题定义为一个优化问题,并提出基于交叉的全局人工蜂群算法求解边缘服务器部署的最优解以降低系统的平均响应时间。充分的实验结果表明,所提算法能够有效降低系统响应时间,算法性能优于其他代表性部署算法。

关键词 移动边缘计算 边缘服务器 人工蜂群算法 计算卸载 组合优化

中图分类号 TP393 **文献标志码** A DOI:10.3969/j.issn.1000-386x.2024.05.034

EDGE SERVER DEPLOYMENT STRATEGY BASED ON IMPROVED ARTIFICIAL BEE COLONY ALGORITHM

Li Bo Yuan Ye* Hou Peng Ding Hongwei
(*School of Information Science and Engineering*, *Yunnan University*, *Kunming 650500*, *Yunnan*, *China*)

Abstract As the first step in the deployment of mobile edge computing architecture, the deployment of edge servers is the foundation and key, and its deployment location is closely related to user experience and system performance, but there are currently few studies focusing on this problem. This paper studied the deployment of edge servers in the mobile edge computing environment in wireless metropolitan area networks. With the goal of minimizing response time, the edge server deployment problem was defined as an optimization problem, and a cross-global based artificial bee colony algorithm was proposed to solve the edge server deployment problem to reduce the average response time of the system. Sufficient experimental results show that the proposed algorithm can effectively reduce the average response time, and the algorithm performance is better than other representative deployment algorithms.

Keywords Mobile edge computing Edge server Artificial bee colony algorithm Computing offloading Combination optimization

0 引 言

随着通信技术、移动互联网的飞速发展,移动终端逐渐取代个人计算机成为人们日常生活的主要工具。但是由于移动终端的物理资源限制,其有限的资源(如电池续航、计算能力、存储容量等)和移动应用程序之间的矛盾给未来移动平台开发带来了巨大挑战。为了提升移动应用程序的性能,可以将移动终端产生的任务通过互联网传输到云计算中心实现远程计算,但是云计算中心通常距离用户非常远,导致移动应用程序的请求响应时间过长,为了有效解决该问题,移动边缘计算(Mobile Edge Computing, MEC)应运而生。MEC 的主要动机是将云计算能力转移到网络边缘,将 MEC 服务器部署在终端层和远程云之间,可将其视为运行于移动网络边缘的云服务器,负责网络流量控制(转发和过滤)和管控各种移动边缘服务和应用。当移动终端的计算能力不能满足其自身需求时,可以通

收稿日期:2020-12-14。国家自然科学基金项目(61562092);云南大学研究生科研创新基金项目(2020311,2020305)。**李波**,教授,主研领域:移动边缘计算,云计算。**袁也**,硕士生。**侯鹏**,硕士生。**丁洪伟**,教授。

过无线网络将计算任务和海量数据卸载至 MEC 服务器处理,以减少核心网的拥塞和降低终端应用的响应延迟。因此,MEC 服务主要是由 MEC 服务器实现的,但是目前没有关于 MEC 服务器的正式定义,也没有指定其在系统中的部署位置,因此产生了 MEC 服务器的部署问题[1]。

目前大多数现有的研究都侧重于 MEC 的计算卸载和资源分配等问题优化边缘计算服务,但是此类研究的前提是假设 MEC 服务器已经部署了,较少关注 MEC 服务器的部署问题。而在服务过程中,MEC 服务器的部署位置对实现系统的低时延、低能耗和负载均衡,提升资源利用率等问题至关重要[2]。

Cloudlet 也是作为远程云的补充而被提出,以充当移动终端的卸载目的地,据信,Cloudlet 和边缘服务器之间存在许多相似之处[3-4],因此 Cloudlet 部署和边缘服务器部署问题可以综合研究。将 Cloudlet 和 MEC 服务器的部署研究按照优化目标分类,大致可以分为优化访问时延、能耗、部署成本 3 类,每一类中代表性的工作总结如表 1 所示。

表 1　现有研究工作总结

优化目标	参考文献	部署方法	方法特点	不足之处
时延	文献[4]	整数线性规划、启发式方法、在线算法	设计了一种快速、可伸缩的启发式方法	以 AP 间的距离代替时延且未考虑排队时延
	文献[5]	近似算法、迭代算法、在线作业路由算法	重点关注优化在线作业路由以最小化平均访问时延	以 AP 间距离代替通信时延并假设网络时延固定
能耗	文献[6]	混合整数规划、基于 Benders 分解的算法	考虑能量和延迟,使得总能耗最小化并满足每个任务的延迟要求	未考虑任务排队情况
	文献[7]	粒子群优化	设计了一种基于粒子群优化的能量感知边缘服务器部署算法	以 AP 间距离代替延迟且未考虑链路传输能耗
部署成本	文献[8]	启发式算法、聚类算法、整数线性规划	研究了未指定容量和约束容量的情况下部署 Cloudlet	以 AP 间的距离代替延迟且未充分考虑其他延迟
	文献[9]	K-Medoids 聚类算法、启发式算法	延迟受限下的最优 Cloudlet 部署和用户关联	以距离替代时延且充分考虑其他延迟

如表 1 所示,首先,以往的工作大都使用基站或者接入点(Access Point,AP)之间的距离代替用户与服务器之间的访问时延,所以其本质上是在优化距离,但是访问时延还与计算任务的数据量、带宽等紧密相关;其次,以往的工作未能充分考虑计算任务服务过程中的时延,如任务在 MEC 服务器的排队时延、计算时延和下载时延等;再者,在无线城域网(Wireless Metropolitan Area Network,WMAN)环境中,由于接入点数量庞大,使用传统的算法解决大规模 MEC 服务器部署问题时可能不适用且其复杂度非常大[10]。因此,本文重点关注 WMAN 环境中 MEC 服务器的部署,从用户的角度出发,以最小化任务的响应时延为优化目标,结合计算任务特征,充分考虑了计算任务从上传到 MEC 服务器并收到计算结果全过程所经历的时间延迟。鉴于智能算法在解决优化问题方面的高效性和可靠性,结合遗传算法和人工蜂群算法特点,针对规模较大的 MEC 服务器部署问题,提出基于全局交叉操作的人工蜂群智能仿生算法(Cross-Global based Artificial Bee Colony Algorithm,CGABC)来求解 MEC 服务器的近似最优布局,以缩短移动用户的响应延迟,满足用户需求,显著提高应用程序的性能,充分的实验结果证明了该算法的有效性。

1　系统模型

1.1　网络模型

WMAN 是一种计算机网络,AP 间使用有线网络连接,并通过 AP 节点为区域中的移动用户提供无线覆盖。因此,边缘服务器可以部署到现有 WMAN 网络中,为 WMAN 网络中的移动用户提供计算服务[9]。并且与云计算相比,MEC 能提供更高的计算效率。从经济的角度来说,边缘服务器尤其适合于 WMAN 环境,首先,大城市地区用户密度高,这意味着大量用户可以访问边缘服务器,其次,WMAN 服务提供商在通过 WMAN 提供边缘计算服务时可以利用规模经济,从而使边缘计算服务对于人们来说更具性价比。

如图 1 所示,考虑由 n 个通过互联网连接的 AP($S=\{s_1,s_2,\cdots,s_n\}$)和 m 个($U=\{u_1,u_2,\cdots,u_m\}$)能够通过 AP 接入边缘计算网络的移动用户所组成的 WMAN。该网络可以使用无向图 $G=(V,L)$ 表示,其中 V 为节点集合,即 $V=S\cup U$。L 为网络中所有通信链路的集合,当且仅当相应的 s_i 和 s_j 之间通过通信链路连接时才存在边 $(s_i,s_j)\in E$,UE 与 AP 之间的通信链路

表示为$(u_i, s_j) \in E$。假设图 G 是连通的,即 G 中的每一个 AP 都可以通过互联网访问另一个 AP[3]。

理想情况下应当尽可能地多部署 MEC 服务器为移动用户提供高可靠的计算服务,但是 MEC 服务器的部署受到实际部署成本和用户计算需求的约束,因此只能战略性地选择有限个位置部署 MEC 服务器[11]。得益于部署良好的电信网络,将 MEC 服务器与 AP 等基础设施一起部署安装是一个很有前途的方案[12]。假设 k 个边缘服务器($E = \{e_1, e_2, \cdots, e_k\}$)被部署在 k 个 AP 上,与 AP 共享一个站址,这样不仅使得 AP 同时具备网络通信和计算能力,而且还能降低部署成本[6]。若移动用户位于和边缘服务器同站部署 AP 的覆盖范围内,则可以直接与边缘服务器通信,否则可以通过其他 AP 作为中继点进行任务卸载实现与边缘服务器的通信。

以图 1 为例,多个 AP 之间通过有线连接实现整个网络的连通,3 个边缘服务器分别与 s_3、s_5、s_7同站部署,使得 s_3、s_5、s_7同时具备通信和计算能力,位于 s_3、s_5、s_7覆盖范围内的移动用户可以直接向其执行计算卸载,否则将通过关联的 AP 作为中继点将计算任务卸载至边缘服务器。边缘服务器部署的位置将对服务质量、资源利用率等产生重要影响。

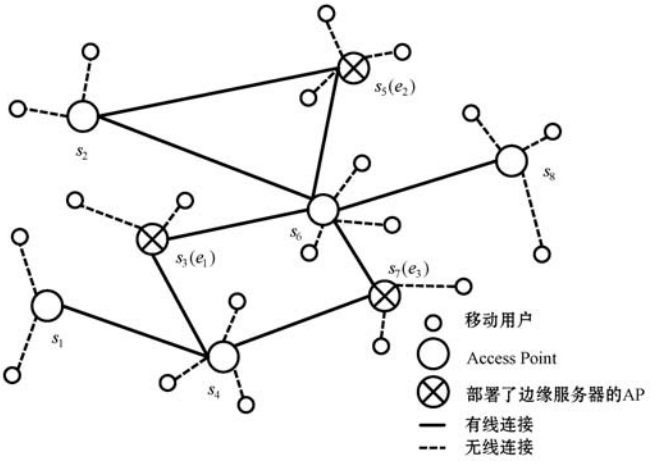

图 1 WMAN 网络示意图

使用三元组$\chi_i(L_i, X_i, \lambda_i)$描述 u_i的计算任务,其中 L_i(单位:MB)表示计算任务的通信数据量大小;X_i表示该任务的计算量(单位:CPU cycle);λ_i表示任务以卸载率为 λ_i的泊松过程进行计算卸载。

1.2 卸载模型

当计算任务执行卸载时,其需要经历上传、服务和下载三个阶段[8],因此可将计算响应延时定义为上传、服务和下载时延之和,并将其表示为:

$$T_i = t_i^{\text{up}} + t_i^{\text{sever}} + t_i^{\text{down}} \tag{1}$$

式中:t_i^{up}表示任务上传时间;t_i^{sever}表示服务时间,具体表示为排队等待时间和计算服务时间;t_i^{down}表示计算结果下载时间。

(1) 上传时延。假设每个用户与 AP 之间的关联已经通过某些用户关联策略确定[8],由于受到传输范围的限制,在一定范围内的 AP 之间是可以通过有线方式直接连通的,距离较远时 s_i将通过其他 AP 作为中继的方式实现与 s_j连通。χ_i经过 h 跳卸载到部署了边缘服务器的 AP,根据香农公式,第 j 跳的数据传输速率可表示为:

$$\begin{aligned} r_j &= B_{hj} \times \log_2\left(1 + \frac{P_{hj} \cdot H_{hj}}{\sigma_{hj}}\right) \\ h &= |d(s_i, s_j)| \end{aligned} \tag{2}$$

式中:$d(s_i, s_j)$为 s_i与 s_j之间的最短路径,h 为最短路径所经过的中继 AP 数目,B_{hj}为 AP 间的通信带宽,P_{hj}是第 j 跳相对发送节点的通信功率,H_{hj}是第 j 跳的信道增益,σ_{hj}为噪声功率。

故计算任务χ_i的上传时延可以表示为:

$$t_i^{\text{up}} = \frac{L_i}{B_s} + \sum_{h}^{j=1} \frac{L_i}{r_j} \tag{3}$$

式中:B_s为移动用户和 AP 间的传输带宽,L_i/B_s为移动用户上传任务到 AP 的传输时延。

(2) 服务时延。χ_i卸载至边缘服务器 e_k后将由 e_k启动计算服务,但是在一些用户密集区域,同一时刻可能有大量任务集中到达 e_k,且 e_k的计算能力是有限的,此时任务在 e_k处将不可避免地发生排队现象[12]。单个边缘服务器的服务模式可将其建模为 $M/M/1$ 系统,假设边缘服务器的服务率为 μ_k,每个用户的任务请求相互独立且随机发生,以到达率为 λ_i的泊松过程到达 e_k,则边缘服务器 e_k总的到达率为 $\lambda^k = \sum_{i=1}^{m} \lambda_i (u_i \in U_k)$,其中 U_k表示被调度到边缘服务器 e_k处的移动用户集合。另一方面,在 MEC 服务器中通过虚拟化技术进行并行计算是不可行的,因此需要按顺序处理计算工作负载[1],故采用 k 个 $M/M/1$ 队列,先到先服务的方式进行排队。

此外,在某些高峰时刻移动用户可能会过于集中发出卸载请求导致边缘服务器出现过载现象,当边缘服务器当前的工作负载总和超过了最大卸载率 $\lambda^{\max}$时,后续到达任务的排队等待时间将过长,严重影响计算服务质量。因此,当边缘服务器过载后,超出的计算任务将被卸载至远程云,使用 ε_k来表示在 e_k上处理的任务比例:

$$\varepsilon_k = \begin{cases} 1 & \lambda^k < \lambda^{\max} \\ \dfrac{\lambda^{\max}}{\lambda^k} & \text{其他} \end{cases} \tag{4}$$

被卸载到远程云的任务是通过互联网传输的,并将云端的服务模式建模为 $M/M/\infty$ 队列,假定该传输将产生固定的延时 ω,并且由于云计算中心具备丰富的计算资源,故任务在远程云上的排队时间可以忽略不计[17]。因此,任务的服务时延可以表示为:

$$t_i^{\text{server}} = \left[f_Q(\varepsilon_k \cdot \lambda^k) + \frac{X_i}{f_k}\right] + (1-\varepsilon_k)\left[\omega + \frac{X_i}{f_c}\right] \tag{5}$$

式中:$f_Q(\lambda) = \dfrac{\lambda}{\mu(\mu-\lambda)}$为边缘服务器不过载时的平均排队时间,$X_i/f_k$为在边缘服务器 e_k上的计算时间,f_k为 e_k的计算能力(单位:CPU cycle/s),f_c为云服务器计算能力,X_i/f_c为在云服务器上的计算时间。

(3) 下载时延。计算完成后,计算结果将由边缘服务器回传至移动用户,通常输出数据量与输入数据量是相关的,因此定义输入输出比(Input data to Output data Ratio,IOR)为输入通信数据量与输出数据量之比[18]即:

$$R_{\text{IO}} = \frac{L_i}{O_i} \tag{6}$$

式中:O_i表示任务下载通信数据量。

计算结果下载过程与上传过程类似,可以得到下载时间为:

$$t_i^{\text{down}} = \frac{O_i}{B_s} + \sum_{j=1}^{h} \frac{O_i}{r_j} \tag{7}$$

综上分析,定义系统中所有移动用户卸载任务的平均延迟为系统的平均响应时间[18],可以表示为:

$$T = \frac{1}{m}\sum_{i=1}^{m} T_i \tag{8}$$

2　问题定义

定义指标变量 $Z = \{z_{ij} \mid 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant n\}$ 用以描述边缘服务器的部署位置,当边缘服务器 e_i与 s_j共同部署时 $z_{ij}=1$,否则 $z_{ij}=0$。$Y = \{y_{ij} \mid 1 \leqslant i \leqslant k, 1 \leqslant j \leqslant m\}$ 用以描述移动用户与边缘服务器之间的关联,当 u_j将任务卸载至 e_i时 $y_{ij}=1$,否则 $y_{ij}=0$。

基于上述分析,MEC 服务器部署问题可以表示为一个优化问题:给定 $1 \leqslant k \leqslant |S|$的整数 k 和系统参数 $(G,\chi,P,H,B,\mu,f,\sigma,\omega)$,在 S 中选择 k 个 AP 部署边缘服务器,移动用户与 AP 关联后将计算任务卸载至相应的边缘服务器和远程云,以最大程度减小系统的平均响应时延,即:

$$\min \quad T$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in E \tag{9}$$

$$\sum_{i=1}^{k} x_{ij} = 1 \quad \forall j \in S \tag{10}$$

$$\lambda^k \leqslant \lambda_{\max}, \mu_k > \lambda^k \quad \forall k \in E \tag{11}$$

$$\sum_{j=1}^{k} y_{ij} = 1 \quad \forall i \in E \tag{12}$$

式(9)保证每个边缘服务器最多只能部署在一个 AP 上,式(10)保证每个 AP 上仅能部署一个边缘服务器,式(11)保证排队系统的稳定性,式(12)保证每个移动用户只能将计算任务卸载至一个边缘服务器。

3　最小化系统响应时间的部署方案

在 WMAN 环境下,由于 AP 和移动用户数量庞大,所以边缘服务器部署问题本身的规模是非常巨大的。问题规模较大时传统的整数规划等解决方法可拓展性较差和复杂性较高。因为 MEC 服务器部署问题本质上是一个优化问题,鉴于智能算法在解决优化问题的高效性和强适用性,可以设计一种智能型算法能在合理的时间内解决该问题。其中,人工蜂群算法不需要了解问题的特殊信息,只需要对问题进行优劣的比较,通过各人工蜂个体的局部寻优行为,最终在群体中使全局最优值突现出来,有着较快的收敛速度,并且在边缘计算环境中,移动用户与 AP 节点之间的分配类似于蜜群与蜜源的关系,所以在本文中选择使用改进的人工蜂群算法解决 WMAN 网络中边缘服务器的部署问题。

人工蜂群算法(Artificial Bee Colony Algorithm,ABC)首先引入蜜源,它代表解空间内的各种可能解,并使用适应度函数来衡量蜜源[14]。再引入三种蜂:采蜜蜂、观察蜂和侦察蜂。采蜜蜂同具体的蜜源联系在一起,这些蜜源是它们当前正在采集的蜜源,采蜜蜂通过生物行为与其他蜜蜂分享信息,观察蜂通过采蜜蜂的信息对食物源做出选择,采蜜蜂总是记住自己以前的最优位置,并根据记忆在邻域搜索。侦察蜂的作用是随机搜索一个新位置。通过三种蜜蜂的合作最终找到最佳蜜源,在 MEC 服务器部署问题中,最佳蜜源的位置也即 MEC 服务器部署位置。

边缘服务器部署问题作为优化问题,当 AP 数目为 n、MEC 服务器数目为 k 时解空间的大小为 C_n^k,从解空间中找到最优解的计算复杂度是非常大的。但是借

助于 ABC 算法,可以通过蜜蜂个体的局部寻优行为来凸显全局寻优,找到最合理的边缘服务器部署位置。但是基本的 ABC 算法因为基于适应值的贪婪选择策略和全局探测策略的单一性两方面原因,导致 ABC 算法缺乏全局最优值记忆和参与算法过程,致使算法因全局探测能力不足而陷入局部最优解[15]。解决收敛到局部极值问题的关键就是增加进化群体的物种多样性和提高全局搜索能力,但增加多样性的同时易导致进化系统后期的震荡,不能快速收敛到全局最优点[16]。鉴于遗传算法中的交叉运算,可以在全局最优的基础上引入交叉系数,提高局部优化能力,增强算法开发性和全局寻优能力,即基于交叉操作的全局人工蜂群算法(Crossover of the Global Artificial Bee Colony, CGABC)。

CGABC 算法结合了遗传算法的全局交叉优势和人工蜂群算法的强大局部搜寻能力,该算法涉及的关键步骤描述如下:

(1) 蜂群初始化:给定移动用户分布数据样本 $\boldsymbol{U}=\{\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_m\}$,其中 $\boldsymbol{u}_i(i=1,2,\cdots,n)$ 是 D 维向量。初始化种群为 N 个蜜蜂,其中采蜜蜂种群规模为 N_e,观察蜂种群规模为 N_u,每个蜜蜂代表一个用户区域的划分。$\boldsymbol{E}=\{\boldsymbol{e}_1,\boldsymbol{e}_2,\cdots,\boldsymbol{e}_k\}$ 为边缘服务器位置向量,其中 $\boldsymbol{e}_i$ 为 D 维向量,k 是边缘服务器数目。

(2) 用户关联:$\boldsymbol{u}_i$ 首先根据最短上传时延原则选择与边缘服务器 $\boldsymbol{e}_k$ 关联,即:

$$y_{ik}=\min\ t_i^{\mathrm{up}}\quad k\in E \tag{13}$$

得到被调度到边缘服务器 $\boldsymbol{e}_k$ 处的移动用户集合 $\boldsymbol{U}_k$,将移动用户划分为 k 个集合。

(3) 可行解与新解:对于 0 时刻,随机生成 N 个边缘服务器位置可行解,每个可行解产生方式如下:

$$e_i^j=e_{\min}^j+rand(0,1)(e_{\max}^j-e_{\min}^j) \tag{14}$$

式中:$j\in\{1,2,\cdots,D\}$,为 D 维解向量的某个分量。分别计算各向量的适应度函数值,并将适应度函数值排名前一半的向量作为初始采蜜蜂种群。对于第 n 代的采蜜蜂,若搜寻次数超过了限定次数 *Limit*,则在当前位置向量附近邻域进行搜索新的位置,搜索方法基于邻域搜索后与全局最优值进行交叉操作来提高算法开发能力,其搜索公式为:

$$new\_e_i^j=\begin{cases}e_i^j+\alpha(e_i^j-e_k^j) & rand<cr\\ e_i^{\mathrm{Global}}+\beta[e_i^{\mathrm{Global}}-(e_i^j+\alpha(e_i^j-e_k^j))] & \text{其他}\end{cases} \tag{15}$$

式中:α 为邻域搜索系数,β 为算法探索开发平衡系数,cr 为交叉系数。

(4) 适应度函数:最小化系统响应时间是部署边缘服务器的优化目标,故定义系统的适应度函数为:

$$f(e)=\frac{1}{m}\sum_{i=1}^{m}T_i \tag{16}$$

得到边缘服务器位置解后计算适应度函数值并按照贪婪准则式(17)替换当前解,然后观察蜂按照式(18)计算的跟随概率选择采蜜蜂进行跟随,转化为采蜜蜂进行邻域搜索,最后侦察蜂搜索新解替换好于最大邻域搜索的解。

$$e_{i+1}=\begin{cases}new\_e_i & f(new\_e_i^j)\leqslant f(e_i)\\ e_i & f(new\_e_i^j)>f(e_i)\end{cases} \tag{17}$$

$$p_i=\frac{f(e_i)}{\sum_{i=1}^{N}f(e_i)} \tag{18}$$

算法的详细描述如算法 1 所示。

算法 1 基于交叉的全局人工蜂群算法部署边缘服务器

输入:$G,\chi,P,H,B,\mu,f,\sigma,\omega$。

输出:Z,Y,系统响应时间。

1. 初始化:移动用户与 AP 的关联,计算任务 χ 生成;采蜜蜂、观察蜂和侦察蜂数量初始化,最大迭代次数 maxCycle,最大搜索次数 *Limit*。
2. 随机产生 N 个可行解式(14)并映射到 AP 位置,每个蜜蜂由 k 个 D 维向量构成,根据式(13)实现移动用户与 ES 之间的关联,计算适应度函数值 $f(e)$,并记录最优解。
3. for iteration = 1 to maxCycle
4. for $i=1$ to N
% 采蜜蜂
5. 按照式(15)在当前解的邻域内进行搜索,寻找新解并映射到 AP 位置。
6. 与全局最优解进行交叉。
7. 检查新解是否在优化搜索范围内。
8. 根据式(16)计算新解的适应度函数值。
9. End
10. for $i=1$ to N
11. 计算选择概率 p_i(式(18))
12. End
13. for $i=1$ to N
% 观察蜂
14. 按照 p_i 选择跟随采蜜蜂。
15. 重复采蜜蜂模式,并根据贪婪准则(式(17))进行全局最优解的更新。
16. End
17. for $i=1$ to N
% 侦察蜂
18. 根据式(15)寻找新解替换超过邻域搜索限制次数 *Limit* 的解并映射到 AP 位置。

19. End
20. End

4　实验仿真与性能分析

4.1　参数设置与对比方法

在 1 000 × 1 000 的仿真区域中,AP 和移动用户服从泊松点分布,设定计算任务通信量满足 2 ~ 5 MB 的均匀分布,计算量满足 0.2 ~ 1 G CPU cycle 的均匀分布,卸载率服从均值为 0.1,方差 0.05 的正态分布。边缘服务器的服务率为 10,其最大卸载率为其服务率的 90%,即 $\lambda_{\max} = 90\%\mu_k$。设置第 j 跳的信道增益为 $H_{hj} = l_{hj}^{-\sigma}$,其中 l_{hj} 为第 j 跳节点间距离,$\sigma = 4$ 为路径损耗因子[17]。其余仿真参数如表 2 所示。

表 2　仿真参数

参数	默认值	参数	默认值
k	[2,6]	f_k	1G CPU cycle/s
n	50	f_c	10G CPU cycle/s
m	[200,600]	IOR	[1,3]
P_{hj}	100 MW	ω	10 s
B_{hj}	15 M	N	20
B_s	10 M	Limit	50
σ_{hj}	1×10^{-10} MW	maxCycle	500
α	2	β	3
cr	0.6	仿真次数	300

为了对比 CGABC 算法的有效性,选择以下 3 种部署方法作为对比算法:

(1) Top-K(基于负载):计算每个 AP 的工作负载并根据负载大小排序,将边缘服务器部署在负载最大的 k 个 AP 上,尽可能地服务越多用户。

(2) DBC(基于密度):根据移动用户分布计算每个 AP 周围移动用户的密度,并将边缘服务器部署在周围用户密度最大的 k 个 AP 上,即将 MEC 服务器部署在计算热点。

(3) K-means(基于距离):根据移动用户分布,将边缘服务器部署在距离 k 个聚类中心最近的 AP 上。

系统的平均响应时间 T 作为优化目标,因此作为比较不同算法性能的性能指标。

4.2　实验结果

设定仿真参数的中间值($k = 4, m = 400, n = 50$)作为基准实验的仿真参数,对比不同算法的实验性能结果如图 2 所示。可以明显看出 CGABC 算法得到的系统平均响应时间是最低的,相较于其他算法,性能提升 18.91% ~ 25.84%,算法性能排序为 CGABC > K-means > Top-K > Random。

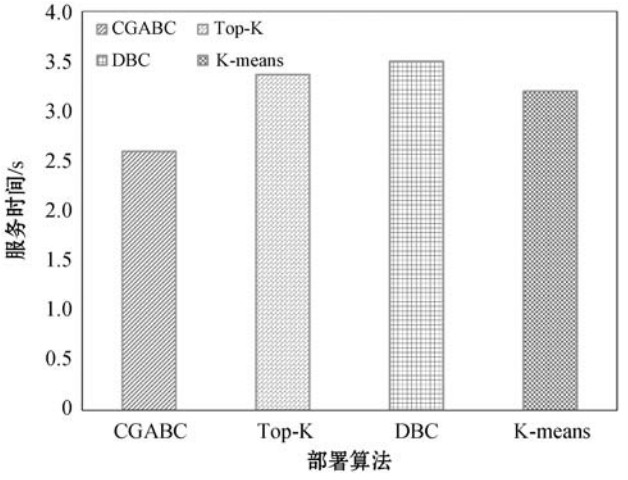

图 2　基准实验的算法性能

移动用户数目、边缘服务器数目、任务卸载率等参数都会对实验结果造成影响,为了探究在不同参数下算法性能的稳定性,分别改变不同的实验参数,得到实验结果如下。

(1) MEC 服务器变化时算法的性能。当 $m = 400$ 时,改变边缘服务器数目,系统的响应时间结果如图 3 所示。可以看出随着边缘服务器数目的增加,用户可选服务器数目增多,故每个边缘服务器的工作负载逐渐减少,系统响应时间也逐渐减少。相较于其他部署方法,在不同边缘服务器数目下,CGABC 算法得到的系统响应时间一直保持最小,平均提升性能 22.36% ~ 39.31%。同时也能明显看出,CGABC 算法的结果呈近线性变化,而其他算法会出现不同程度的波动,因此 CGABC 算法的有效性和稳定性都是优于其他算法的。

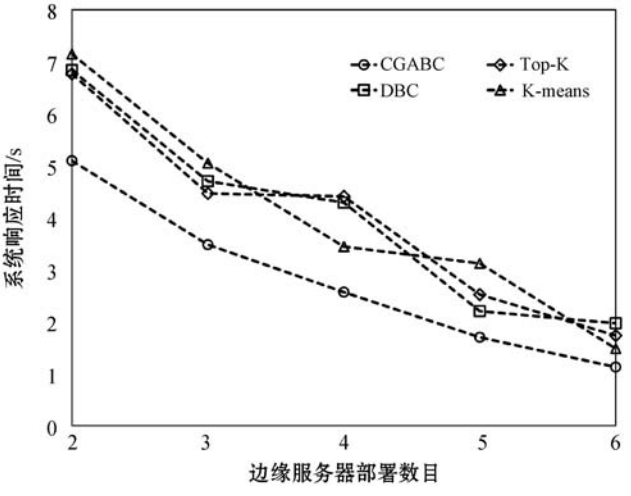

图 3　边缘服务器数目变化时的系统响应时间

(2) 移动用户数目变化时算法的性能。当边缘服务器数目不变时改变移动用户数目,系统响应时间结果如图 4 所示,系统的平均服务时间如图 5 所示。从

图 4 可以看出,随着移动用户数目的增加,边缘服务器的工作负载逐渐增大,导致系统响应时间增加,但是 CGABC 算法一直保持系统响应时间最小,并且用户数目越多,相较于其他算法的优势愈加明显(性能提升 26.31% ~35.79%)。从图 5 还可以明显观察到,当用户数目变化时,CGABC 算法得到的系统服务时间也是最低的。因为其他算法只考虑负载、密度和距离等单方面因素从而决定边缘服务器位置,没有从用户角度出发(尽可能地缩短响应时间),导致部署结果不理想。

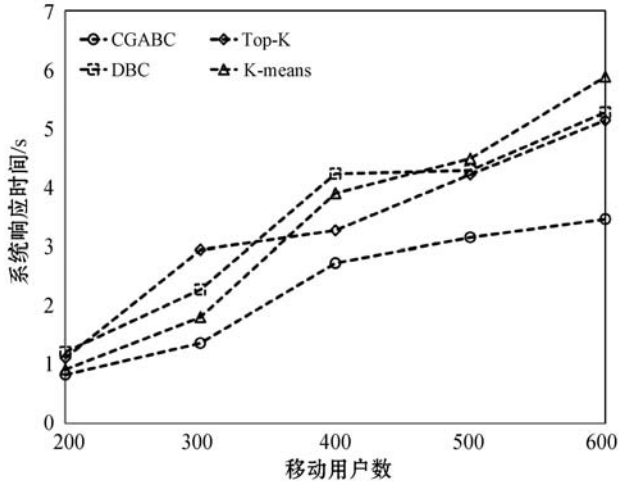

图 4 移动用户数目变化时的系统响应时间

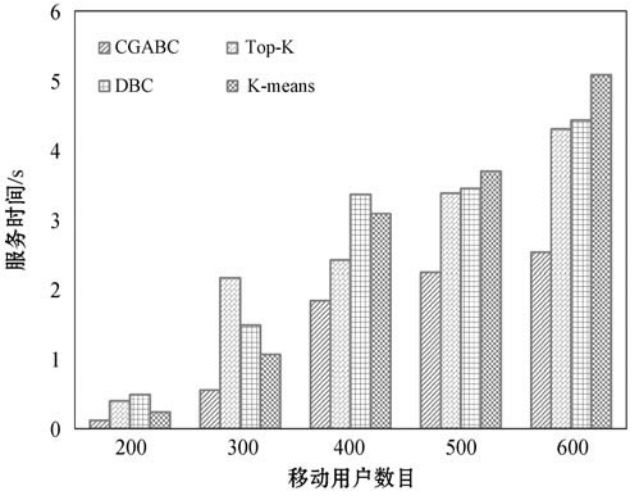

图 5 移动用户数目变化时的系统服务时间

(3) 任务卸载率变化时算法的性能。当 $k=4$, $m=400$ 时,改变计算任务的平均卸载率,可以得到系统的平均响应时间如图 6 所示。当任务卸载率逐渐增加时,系统负载也逐渐增加,计算任务的服务时间也会逐渐增大。可以明显看出,卸载率越大,CGABC 算法的优势越明显,因为 CGABC 根据系统响应时间不断迭代寻求最优解的过程中调整了每个边缘服务器的负载率,使得系统负载均衡,不易出现任务过于集中导致排队时间过大的情况。

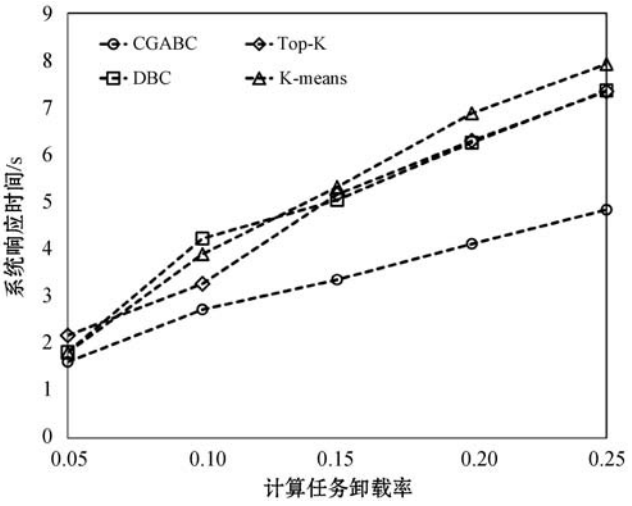

图 6 任务卸载率变化时的系统响应时间

上述实验表明,在边缘服务器数目、不同移动用户数目和不同卸载率情况下,CGABC 算法相比较于其他部署算法在优化系统响应时间方面算法性能和算法稳定性都是最优的,并且在系统负载较大时性能越明显。证明了 CGABC 算法能够优化边缘服务器部署结果。

5 结 语

本文针对移动边缘计算环境下边缘服务器的部署方法进行了研究,考虑系统的上传、服务和下载时间作为系统响应时间,提出了基于交叉的全局人工蜂群部署算法以优化边缘服务器部署的系统响应时间。充分的实验结果表明,所提算法能够最小化系统响应时间,优化边缘服务器的部署。

当前工作还未考虑到移动用户的移动性、网络的异构性对部署结果的影响,这些因素将在未来的工作中得到充分考虑。

参 考 文 献

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

[2] Zhao L, Sun W, Shi Y P, et al. Optimal placement of cloudlets for access delay minimization in SDN-based internet of things networks[J]. IEEE Internet of Things Journal, 2018, 5(2): 1334 - 1344.

[3] Jia M, Cao J N, Liang W F. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing, 2017, 5(4): 725 - 737.

[4] Xu Z C, Liang W F, Xu W Z, et al. Efficient algorithms for capacitated cloudlet placements[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 27(10): 2866 -

2880.
[5] Meng J Y, Shi W B, Tan H S, et al. Cloudlet placement and minimum-delay routing in cloudlet computing[C]//3rd International Conference on Big Data Computing and Communications,2017:297 - 304.
[6] Yang S, Li F, Shen M, et al. Cloudlet placement and task allocation in mobile edge computing[J]. IEEE Internet of Things Journal,2019,6(3):5853 - 5863.
[7] Li Y Z, Wang S G. An energy-aware edge server placement algorithm in mobile edge computing[C]//IEEE International Conference on Edge Computing,2018:66 - 73.
[8] Chen L, Wu J G, Zhou G Q, et al. QUICK: QoS-guaranteed efficient cloudlet placement in wireless metropolitan area networks[J]. The Journal of Supercomputing,2018,74(8):4037 - 4059.
[9] Ma L J, Wu J G, Chen L. DOTA: Delay bounded optimal cloudlet deployment and user association in WMANs[C]//17th IEEE International Symposium on Cluster,2017:196 - 203.
[10] Ma L J, Wu J G, Chen L, et al. Fast algorithms for capacitated cloudlet placements[C]//21st International Conference on Computer Supported Cooperative Work in Design,2017:439 - 444.
[11] Ren Y Z, Zeng F, Li W J, et al. A low-cost edge server placement strategy in wireless metropolitan area networks[C]//27th International Conference on Computer Communication and Networks,2018:1 - 6.
[12] Wang S G, Zhao Y L, Xu J L, et al. Edge server placement in mobile edge computing[J]. Journal of Parallel Distributed Computing,2019,127:160 - 168.
[13] Li B, Hou P, Wang K Y, et al. Deployment of edge servers in 5G cellular networks[J]. Transactions on Emerging Telecommunications Technologies,2020,33(8):e3937.
[14] Gao W F, Liu S Y, Huang L. A global best artificial bee colony algorithm for global optimization[J]. Journal of Computational and Applied Mathematics,2012,236(11):2741 - 2753.
[15] Gao W F, Liu S Y. Improved artificial bee colony algorithm for global optimization[J]. Information Processing Letters,2011,111(17):871 - 882.
[16] Zhu G P, Kwong S. Gbest-guided artificial bee colony algorithm for numerical function optimization[J]. Applied Mathematics and Computation,2010,217(7):3166 - 3173.
[17] 赵临东,庄文芹,陈建新,等. 异构蜂窝网络中分层任务卸载:建模与优化[J]. 通信学报,2020,41(4):34 - 44.

(上接第 217 页)

化损失率,提高网络鲁棒性,最终实现车辆、路灯、房屋、行人等多类障碍物的检测。通过实验验证,本文提出的改进后的 SegNet 神经网络算法实现了复杂道路的多类障碍物检测,障碍物识别的准确率高达 97%,具有优异的准确率和鲁棒性。

参 考 文 献

[1] Chen L C, Papandreou G, Kokkinos I, et al. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2018,40(4):834 - 848.
[2] Liu L, Ouyang W L, Wang X G, et al. Deep learning for generic object detection: A survey[J]. International Journal of Computer Vision,2020,128(2):261 - 318.
[3] Cheng X, Liu H S. A novel post-processing method based on a weighted composite filter for enhancing semantic segmentation results[J]. Sensors(Basel),2020,20(19):5500.
[4] Shelhamer E, Long J, Darrell T. Fully convolutional networks for semantic segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2017,39(4):640 - 651.
[5] Hu X G, Yang H G. DRU-Net: A novel U-net for biomedical image segmentation[J]. IET Image Processing,2020,14(1):192 - 200.
[6] Ni J, Wu J H, Tong J, et al. GC-Net: Global context network for medical image segmentation[J]. Computer Methods and Programs in Biomedicine,2020,190:105121.
[7] Badrinarayanan V, Kendall A, Cipolla R. SegNet: A deep convolutional encoder-decoder architecture for image segmentation[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2017,39(12):2481 - 2495.
[8] Li G, Liu Q W, Ren W, et al. Automatic recognition and analysis system of asphalt pavement cracks using interleaved low-rank group convolution hybrid deep network and SegNet fusing dense condition random field[J]. Measurement,2021,170:108693.
[9] 史文玲,杜慧谦,梅文波. 用于单图像超分辨率的新型通道注意力残差网络[J]. 北京理工大学学报,2020,29(3):345 - 353.
[10] Kusuma G P, Wigati E K, Chandra E. A review of recent advancements in appearance-based object recognition[J]. Procedia Computer Science,2019,157(3):613 - 620.
[11] Lin T Y, Goyal P, Girshick R, et al. Focal loss for dense object detection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence,2020,42(2):318 - 327.
[12] Eigen D, Fergus R. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture[C]//IEEE International Conference on Computer Vision,2015:2650 - 2658.