计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2023,32(4):262−267 [doi: 10.15888/j.cnki.csa.009013]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于移动边缘计算的任务卸载优化①

彭　昇[1], 赵建保[2], 魏敏捷[3], 秦伦明[1]

[1](上海电力大学 电子与信息工程学院, 上海 201306)
[2](国网信息通信产业集团有限公司, 北京 102200)
[3](上海电力大学 电气工程学院, 上海 201306)
通信作者: 魏敏捷, E-mail: weiminjie@shiep.edu.cn

摘　要: 随着智慧物联体系的发展, 物联网中应用程序的种类与数量不断增加. 在移动边缘计算 (mobile edge computing, MEC) 中, 通过允许移动用户将任务卸载至附近 MEC 服务器以加快移动应用程序的速度. 本文通过考虑不同任务属性、用户的移动性和时间延迟约束模拟移动边缘场景. 根据用户移动轨迹, 将目标建模为寻找满足时延约束条件且在卸载过程中产生最小能耗 MEC 服务器优化模型, 并提出一种最小能耗卸载算法求解该问题的最优解. 仿真结果表明, 在约束条件下, 提出的算法可以找到在用户移动轨迹中产生最小能耗的 MEC 服务器, 并显著降低任务卸载过程的能耗与时延, 提高应用程序服务质量.

关键词: 移动边缘计算; 任务卸载; 物联网 (IoT); MEC 服务器

引用格式: 彭昇,赵建保,魏敏捷,秦伦明.基于移动边缘计算的任务卸载优化.计算机系统应用,2023,32(4):262–267. http://www.c-s-a.org.cn/1003-3254/9013.html

Task Offload Optimization Based on Mobile Edge Computing

PENG Sheng[1], ZHAO Jian-Bao[2], WEI Min-Jie[3], QIN Lun-Ming[1]

[1](College of Electronic Information Engineering, Shanghai University of Electric Power, Shanghai 201306, China)
[2](State Grid Information and Telecommunication Group Co. Ltd., Beijing 102200, China)
[3](College of Electrical Engineering, Shanghai University of Electric Power, Shanghai 201306, China)

Abstract: With the development of the smart Internet of things (IoT) system, the type and number of applications in the IoT continue to increase. In mobile edge computing (MEC), mobile applications are accelerated by allowing mobile users to offload tasks to nearby MEC servers. This study simulates mobile edge scenarios by analyzing task attributes, user mobility, and delay constraints. In addition, according to the moving trajectory of users, the goal is modeled. Specifically, it aims to find an optimization model for MEC servers that satisfies the delay constraints and generates the minimum energy consumption during the offloading. The study also proposes a minimum energy offloading algorithm to find the optimal solution to this problem. The simulation results show that under the constraints, the proposed algorithm can find a MEC server that generates the minimum energy consumption in the moving trajectory of users, significantly reduce the energy consumption and delay during task offloading, and improve the application service quality.

Key words: mobile edge computing (MEC); task offloading; Internet of things (IoT); MEC server

伴随 5G 通信技术的发展, 移动游戏、图像视频处理和流媒体等服务变得越来越广泛[1]. 用来提供服务的应用程序需要大量的计算资源, 而移动设备有限的计算能力降低了应用程序的使用性能. MEC 促进应用程序的使用更节能、低时延和高灵敏[2]. 在 MEC 中, 大量 MEC 服务器部署在多个基站上, 形成移动边缘网

① 基金项目: 国家自然科学基金面上项目 (61872239)
收稿时间: 2022-08-29; 修改时间: 2022-09-27; 采用时间: 2022-09-30; csa 在线出版时间: 2022-12-16
CNKI 网络首发时间: 2022-12-19

络 (mobile edge network, MEN)[3]. 移动用户可以通过 MEN 卸载密集型任务, 需要大量计算资源的任务卸载至 MEN 中的一个或多个 MEC 服务器上进行处理. 与传统云计算相比, MEC 服务器比远程云端服务器更接近移动用户, 所以计算和传输的能耗与延迟更少[4].

目前, MEC 的研究工作主要集中在卸载决策问题, 考虑不同的环境因素与用户需求, 如计算速度、能耗优化和资源成本等, 来获取最优的卸载解决方案[5]. 传统的卸载决策适用于用户位置是静态的, 但是移动用户设备一般具有很高的流动性, 所以传统卸载决策不适用于现实应用场景. 当考虑用户移动时, 每个移动用户在通信范围内会有多个 MEC 服务器, 将任务直接分配给一个 MEC 服务器并将所有任务卸载至此服务器不一定是最佳卸载决策, 移动用户还可以将任务上传至一个 MEC 服务器, 然后通过传输到另一个 MEC 服务器来获取结果, 但这一过程还需要考虑 MEC 服务器之间通信和传输成本的影响.

为了有效解决用户移动性对任务执行与卸载决策的影响, 本文通过随机路径点模型表示移动用户轨迹, 将问题建模为满足时延约束的最小能耗 MEC 服务器优化模型, 同时提出一种求解满足时延约束的卸载决策的最优解算法. 仿真结果表明, 所提出的算法可以找到最小能耗 MEC 服务器协同卸载并显著降低 MEN 中任务卸载的能耗与时延.

1 国内外研究现状

计算卸载是 MEC 中的关键技术之一, 可以有效降低任务能耗与时延. Lin 等人[6] 研究在用户移动性下将任务卸载至位于用户轨迹的 MEC 服务器的最佳任务卸载 MEC 服务器, 没有考虑任务属性与约束条件. 协同边缘计算 (collaborative edge computing, CEC) 允许多个服务器协同卸载不同类型的任务, Wang 等人[7] 研究在 CEC 环境下的高效节能任务卸载, 提出基于匈牙利算法的任务卸载方案, 减少了时延与能耗. Kai 等人[8] 研究了 D2D 通信技术在 MEN 中满足时延约束并找到可卸载任务, 然后提出低任务复杂度切换算法进行系统最小化能耗. Wang 等人[9] 使用基于粒子群优化算法 (particle swarm optimization, PSO) 和博弈论的 MEC 分配策略来最小化时延. 在大部分 MEC 计算卸载方案中, 没有考虑移动用户的移动性, 讨论均假设用户是静态的且用户与 MEC 服务器之间的通信是一直存在的. 但在现实应用场景中并不具备实用性, 移动用户普遍具有移动性, 一旦超出规定边缘服务器的传输范围, 被卸载的任务就会传输失败, 从而延长应用程序的响应时间, 导致边缘计算资源的浪费. 因此, 考虑边缘移动用户设备的移动性是非常必要的. Wu 等人[10] 讨论了基于移动感知的任务卸载, 实时寻找每个任务最适合的云或边缘服务器, 通过基于深度学习 (deep learning, DL) 的算法来预测用户轨迹. Liu 等人[11] 研究在 MEC 环境下通过最优卸载来减少任务的传输, 并提出一种次遗传算法 (secondary genetic algorithm, SGA), SGA 将任务卸载到位于用户轨迹上的 MEC 服务器, 从而降低任务时延与能耗. 本文针对用户移动性和延迟约束, 利用沿用户轨迹的基站之间的协作, 提出满足约束最小时延与能耗卸载方案.

2 系统模型

如图 1 所示, MEN 由多个基站组成, 每个基站部署一台 MEC 服务器, MEC 服务器用于接收、执行和传输基站信号范围内用户卸载的计算任务[12], 基站之间通过中央基站进行通信, 移动用户可以随时离开基站的信号范围, 在用户轨迹中的所有边缘服务器都可以执行被卸载的任务. 在图 1 中, 移动用户 1 卸载任务后离开基站 1 的范围并前往基站 2. 在这种情况下, 基站 1 的 MEC 服务器 1 可以将卸载的任务发送给用户当前所在的基站 2 的 MEC 服务器 2 中.

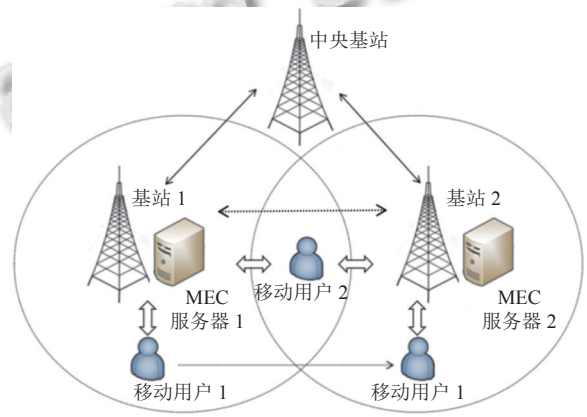


图 1 系统模型

2.1 终端任务模型

假设移动用户表示为$N=\{1,2,\cdots,i,\cdots,N\}$, 每个用户的任务表示为$T_i$: r_i表示任务的大小, 单位为比特 (bit); c_i表示所需的计算资源, 单位为转 (cycle); $t_i^{\max}$表示任务的最大时延, 单位为秒 (s); 在 MEN 系统中, 有

M 个基站, 每个基站的信号范围是r_j, 单位为米 (m), 边缘服务器的计算能力是c_j, 单位为 cycle/s.

2.2 任务上传模型

移动用户首先进行卸载任务, 假设用户i与服务器j之间通信速率为v_{ij}, 则可以表示为:

$$v_{ij} = B\log_2\left(1+\frac{p_i d_{ij}^{-\alpha}}{N_0 B}\right) \tag{1}$$

其中, B表示信道带宽, d_{ij}表示用户 i 与 MEC 服务器 j 之间的距离, N_0表示噪声功率谱密度, α表示信道衰落参数, p_i表示移动用户 i 的传输速率.

任务上传时间取决于任务大小与通信速率, 则可以表示为:

$$t_{ij} = \frac{r_i}{v_{ij}} \tag{2}$$

任务上传能耗取决于上传时间与用户传输速率, 则可以表示为:

$$e_{ij} = t_{ij}\cdot p_i \tag{3}$$

2.3 任务传输模型

MEC 网络中, 任务可以通过基站之间相互通信, 服务器j_1与j_2之间的传输速率可以表示为:

$$v_{j_1 j_2} = B\log_2\left(1+\frac{p_{j_1} d_{j_1 j_2}^{-\alpha}}{N_0 B}\right) \tag{4}$$

其中, $d_{j_1 j_2}$表示服务器j_1和j_2之间的距离, p_{j_1}为 MEC 服务器j_1的传输功率.

传输时间与能耗分别为:

$$t_{j_1 j_2} = \frac{r_i}{v_{j_1 j_2}} \tag{5}$$

$$e_{j_1 j_2} = t_{j_1 j_2}\cdot p_{j_1} \tag{6}$$

2.4 任务执行模型

服务器j中任务i的计算时间为:

$$t_{ij}^c = \frac{c_i}{c_j} \tag{7}$$

计算能耗为:

$$e_{ij}^c = t_{ij}^c \cdot p_j \tag{8}$$

其中, p_j是服务器j的计算功率.

由于执行处理后任务数据非常小, 并且能耗与时延可忽略不计, 所以不考虑下载传输延迟和能耗[13−15].

3 基于移动边缘计算的任务卸载方案

3.1 问题建立

假设每个用户i具有可卸载的任务T_i. 随着用户移动, 可卸载用户i任务的 MEC 服务器随时间变化, M_i^t表示在t时可用 MEC 服务器的集合, mt_i表示用户i移动的总时间. 本节目标是从用户i移动轨迹中的一组可用服务器中找到一个 MEC 服务器, 满足任务i的时延约束并且最小化能耗. 所以任务卸载可以建模为:

$$\begin{cases} \min \sum_{i\in N}\sum_{j\in S_i} x_{ij}E_{ij} \\ C1: t_{ij} \leqslant t_i^{\max} \\ C2: x_{ij}\in\{0,1\} \\ C3: \sum_{j\in S_i} x_{ij} = 1 \end{cases} \tag{9}$$

其中, 约束 1 ($C1$) 表示总时延小于最大时延, 约束 2 ($C2$) 和约束 3 ($C3$) 表示任务T_i只在服务器j上进行一次.

总能耗E_{ij}与总时延T_{ij}包括任务上传、传输和执行所消耗的能量与时间, 可以表示为:

$$E_{ij} = e_{ij_1} + \sum_{t=t_1}^{t_2} e_{j\in S_i^t, j\in S_i^{t+1}} + e_{ij}^c \tag{10}$$

$$T_{ij} = t_{ij_1} + \sum_{t=t_1}^{t_2} t_{j\in S_i^t, j\in S_i^{t+1}} + t_{ij}^c \tag{11}$$

其中, $0 \leqslant t_1 < t_2 < mt_i$.

3.2 算法提出

本节目标为中央基站将每个任务分配给满足最大时延且产生最小能耗的 MEC 服务器, 对于每个用户$i\in N$, 在每个时间段$t\in[0, nt_i]$, 找到可用的 MEC 服务器集合M_i^t. 在此集合中找到两个 MEC 服务器, 一个用于执行任务, 一个用于传输任务, 从中可以将任务卸载到位于下一时间段的服务器中. 由于存在时延约束, 需计算检查每个时间段中任务上传、传输和执行产生的时延是否满足条件.

对每个可用服务器$j\in M_i^t$, 计算 t−1 时传输过程与执行过程所产生的能耗, 将产生能耗最小的服务器分配给任务T_i执行.

将 t−1 之前的传输能耗与 t+1 时任务从服务器卸载至用户所需的能耗相加, 即可得在时间 t 传输所产生的能耗.

最后输出x_i, 即为满足时延约束且生成最小能耗的最优 MEC 服务器.

本文算法简要流程图如图 2 所示.

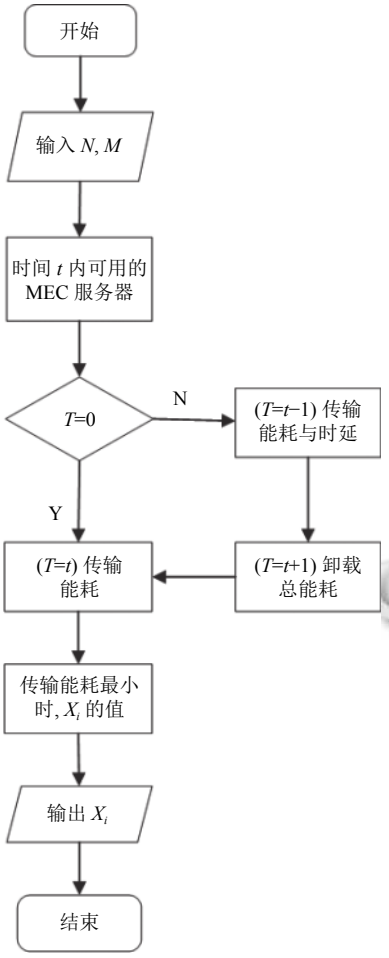


图 2 简要流程图

本文算法流程如算法 1 所示.

算法1. 延迟约束最小能耗卸载算法

输入: 移动用户N, 基站M

输出: $x_i,\forall i\in N$

1) for each $i\in N$ do
2) for each $t\in[0,mt_i]$ do
3) M_i^t
4) for each $j\in M_i^t$ do
5) if t==0 then
6) $t_j^c=t_{ij}+t_{ij}^c$
7) if $t_j^c\leqslant t_i^{\max}$ then
8) $e_j^c=e_{ij}+e_{ij}^c$
9) else
10) $e_j^c\leftarrow\infty$
11) end if
12) else
13) $t_j^c(t-1)$
14) if $t_j^c\leqslant t_i^{\max}$ then
15) $e_j^c(t-1)$
16) else
17) $e_j^c\leftarrow\infty$
18) end if
19) end if
20) end for
22) for each $j\in M_i^t$ do
23) 计算t+1时卸载产生总能耗
24) end for
25) 取$e_{j\omega}^t$最小值时, j_t的值
26) $e_{j_{0,\cdots,t}}=e_{j_{0,\cdots,t-1}}+e_{j_{t-1}j_t}$
27) $t_{j_{0,\cdots,t}}=t_{j_{0,\cdots,t-1}}+t_{j_{t-1}j_t}$
28) end for
29) 取$e_{j_t}^t$最小值时, x_i的值
30) end for
31) return $x_i,\forall i\in N$

4 实验分析

本节中, 通过 Python 编程语言评估所提出算法在考虑用户移动性和延迟约束下通过将任务卸载至最佳 MEC 服务器实现最小能耗的任务卸载性能. 引入文献 [16] 中的参数设置, 设置区域范围内共有 126 个基站, 在基站覆盖区域内的移动用户遵循随机路径点模型[17]. 随机路径点模型是移动用户移动的随机模型, 包含各移动用户的位置、速度和加速度随时间的变化情况. 基站的覆盖半径大小取 [70, 100] m, MEC 服务器的 CPU 容量取 [7, 20] GHz, 计算能力取 [3, 5] cycles/s, 传输功率取 [0.1, 1] W, 通信信道带宽为 1 MHz, 用户取 [1, 3] km/h 的速度移动, 时延约束在 [0.1, 1] s. 移动用户的 CPU 容量取 [1, 10] GHz, 传输功率取 [7, 15] W, 计算功率取 [7, 10] W. 本文提出的算法与本地执行 (local execution, LE) 算法、随机分配 (random assignment, RA) 算法[18] 和贪婪分配 (greedy assignment, GA) 算法[19] 进行比较.

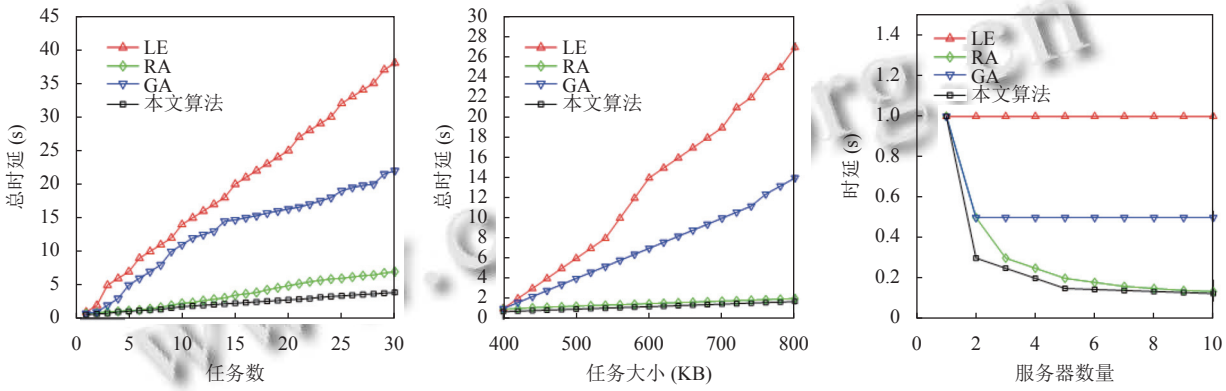
LE: 任务在本地服务器进行存储和处理.

RA: 将一个任务划分为多个子任务, 随机分配给多个相邻的 MEC 服务器进行处理. 这是目前工业上最常见、应用最广泛的任务卸载算法.

GA: 将在本地服务器未卸载的任务分配至相邻 CPU 容量最大的 MEC 服务器进行卸载.

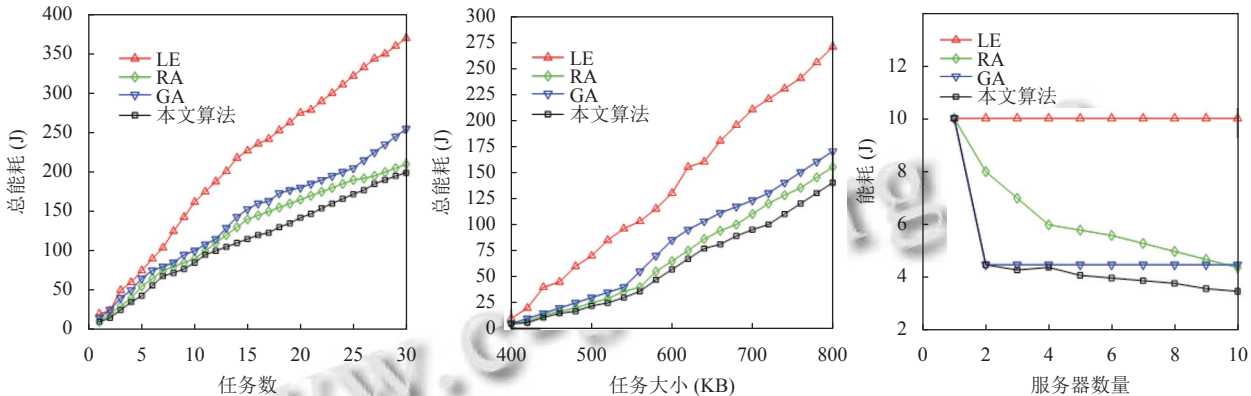
图 3(a) 表示总延迟与移动用户任务数的关系. 总延迟随任务数的增加而增加. 当任务数达到最大时, 本文算法较 LE 算法、GA 算法和 RA 算法时延分别降低约 89.37%、79.81% 和 41.26%. 分析可得, 一方面, 本文算法将任务划分为多个子任务, 提高了任务卸载效率. 另一方面, 将子任务分配给附近产生最低能耗的 MEC 服务器进行分布式处理, 所以产生的时延少于其他算法. 因此, 本文提出的算法更适用于大规模任务分

配卸载场景. 图 3(b) 表示总延迟与输入任务数据大小的关系. 总延迟随任务大小的增加而增加. 当任务大小数据增加至 800 KB 时, 本文算法较 LE 算法、GA 算法和 RA 算法时延分别降低约 93.70%、84.03% 和 13.25%. 因此, 本文提出的算法更适用于大数据任务分配卸载场景. 从图 3(a) 和图 3(b) 可知, LE 算法与 GA 算法产生时延远高于 RA 算法与本文算法, 证明将任务分成各个子任务并分配给其他 MEC 服务器可以有效降低时延, 提高任务处理效率. 图 3(c) 表示时延与 MEN 系统中 MEC 服务器的数量的关系. 当只有一台 MEC 服务器时, 只有本地服务器进行任务处理, 因此 4 种算法时延相同. 在任务输入大小且服务器 CPU 计算能力相同情况下, 随着周围服务器数量的增加, 任务划分后的子任务数量增加, 子任务数据量减少, 相应处理时间减少. 所以, MEN 中 MEC 服务器数量越多, 本文所提算法产生的时延就越少.



(a) 总延迟与任务数的关系 (b) 总延迟与任务大小的关系 (c) 时延与服务器数量的关系

图 3 总延迟与任务数和任务大小的关系, 时延与服务器数量的关系



(a) 总能耗与任务数的关系 (b) 总能耗与任务大小的关系 (c) 能耗与服务器数量的关系

图 4 总能耗与任务数和任务大小的关系, 能耗与服务器数量的关系

图 4(a) 表示总能耗与任务数的关系. 当任务数达到最大时, 本文算法较 LE 算法、GA 算法和 RA 算法能耗分别降低约 51.35%、29.41% 和 14.29%. 表明本文算法可以有效降低能耗, 并且分布式任务分配的方式比集中式任务分配方式所产生的总能耗更小. 图 4(b) 表示总能耗与任务数据大小的关系. 在服务器数量一定的条件下, 随着任务数据大小增加至 800 KB 时, 本文算法较于 LE 算法、GA 算法和 RA 算法所产生的能耗降低约 48.15%、17.54% 和 9.68%, 表明该算法更适合处理大数据任务分配场景. 图 4(c) 表示能耗与服务器数量大小的关系. 在任务输入大小且服务器 CPU 计算能力相同情况下, 本文算法较 LE 算法、GA 算法和 RA 算法产生的能耗最小. 由图可知, 随着服务器数量增加, RA 算法产生的能耗高于 GA 与本文算法, 所以 RA 算法在降低能耗方面具有局限性.

结合以上对比, 本文算法相较于现有算法, 更适用于大规模和大数据量的任务卸载的物联网体系应用环境.

5 总结与展望

面向 MEC 中任务属性、移动用户移动性和时延约束的情况, 本文提出了一种基于边缘计算下的任务卸载优化算法. 该算法将任务与 MEN 中产生最小能耗的 MEC 服务器协同卸载, 根据用户移动性优化任务的时延与能耗. 通过对该算法的大量仿真并与 LE 算法、GA 算法和 RA 算法进行比较, 结果表明, 本文所提算法在优化时延和能耗方面都优于其他算法且更适用于 5G 大规模和大数据的物联网体系. 今后的研究将考虑系统成本模型改进与新变量的插入, 应用于新的智慧物联场景中.

参考文献

1 齐彦丽, 周一青, 刘玲, 等. 融合移动边缘计算的未来 5G 移动通信网络. 计算机研究与发展, 2018, 55(3): 478–486. [doi: 10.7544/issn1000-1239.2018.20170801]

2 施巍松, 张星洲, 王一帆, 等. 边缘计算: 现状与展望. 计算机研究与发展, 2019, 56(1): 69–89. [doi: 10.7544/issn1000-1239.2019.20180760]

3 Ma X, Lin C, Zhang H, *et al*. Energy-aware computation offloading of IoT sensors in cloudlet-based mobile edge computing. Sensors, 2018, 18(6): 1945. [doi: 10.3390/s18061945]

4 Elgendy IA, Zhang WZ, Zeng YM, *et al*. Efficient and secure multi-user multi-task computation offloading for mobile-edge computing in mobile IoT networks. IEEE Transactions on Network and Service Management, 2020, 17(4): 2410–2422. [doi: 10.1109/TNSM.2020.3020249]

5 Wang Z, Zhao ZW, Min GY, *et al*. User mobility aware task assignment for mobile edge computing. Future Generation Computer Systems, 2018, 85: 1–8. [doi: 10.1016/j.future.2018.02.014]

6 Lin H, Zeadally S, Chen ZH, *et al*. A survey on computation offloading modeling for edge computing. Journal of Network and Computer Applications, 2020, 169: 102781. [doi: 10.1016/j.jnca.2020.102781]

7 Wang J, Wu WB, Liao ZF, *et al*. An energy-efficient offloading scheme for low latency in collaborative edge computing. IEEE Access, 2019, 7: 149182–149190. [doi: 10.1109/ACCESS.2019.2946683]

8 Kai Y, Wang JY, Zhu HL. Energy minimization for D2D-assisted mobile edge computing networks. Proceedings of the 2019 IEEE International Conference on Communications. Shanghai: IEEE, 2019. 1–6.

9 Wang YT, Sheng M, Wang XJ, *et al*. Mobile-edge computing: Partial computation offloading using dynamic voltage scaling. IEEE Transactions on Communications, 2016, 64(10): 4268–4282.

10 Wu CR, Peng QL, Xia YN, *et al*. Mobility-aware tasks offloading in mobile edge computing environment. Proceedings of the 2019 7th International Symposium on Computing and Networking. Nagasaki: IEEE, 2019. 204–210.

11 Liu ZL, Wang XX, Wang DY, *et al*. Mobility-aware task offloading and migration schemes in SCNs with mobile edge computing. Proceedings of the 2019 IEEE Wireless Communications and Networking Conference. Marrakesh: IEEE, 2019. 1–6.

12 Tran TX, Pompili D. Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Transactions on Vehicular Technology, 2019, 68(1): 856–868. [doi: 10.1109/TVT.2018.2881191]

13 张海波, 李虎, 陈善学, 等. 超密集网络中基于移动边缘计算的任务卸载和资源优化. 电子与信息学报, 2019, 41(5): 1194–1201. [doi: 10.11999/JEIT180592]

14 Deng XH, Yin J, Guan PY, *et al*. Intelligent delay-aware partial computing task offloading for multi-user industrial Internet of things through edge computing. IEEE Internet of Things Journal, 2021, 8(16): 12559–12568.

15 Choi J, Ahn S. Scalable service placement in the fog computing environment for the IoT-based smart city. Journal of Information Processing Systems, 2019, 15(2): 440–448.

16 Peng QL, Xia YN, Feng Z, *et al*. Mobility-aware and migration-enabled online edge user allocation in mobile edge computing. Proceedings of 2019 IEEE International Conference on Web Services. Milan: IEEE, 2019. 91–98.

17 Lai P, He Q, Abdelrazek M, *et al*. Optimal edge user allocation in edge computing with variable sized vector bin packing. Proceedings of the 16th International Conference on Service-oriented Computing. Hangzhou: Springer, 2018. 230–245.

18 Xing H, Liu L, Xu J, *et al*. Joint task assignment and resource allocation for D2D-enabled mobile-edge computing. IEEE Transactions on Communications, 2019, 67(6): 4193–4207. [doi: 10.1109/TCOMM.2019.2903088]

19 Wei F, Chen SX, Zou WX. A greedy algorithm for task offloading in mobile edge computing system. China Communications, 2018, 15(11): 149–157. [doi: 10.1109/CC.2018.8543056]

(校对责编: 孙君艳)