软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2023,34(2):849−867 [doi: 10.13328/j.cnki.jos.006417]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向移动边缘计算网络的高能效计算卸载算法*

张祥俊[1,2], 伍卫国[1,2], 张 弛[1], 柴玉香[1], 杨诗园[1], 王 雄[1]

[1](西安交通大学 计算机科学与技术学院, 陕西 西安 710049)

[2](西安交通大学 国家高性能计算中心 (西安), 陕西 西安 710049)

通信作者: 伍卫国, E-mail: wgwu@mail.xjtu.edu.cn

摘 要: 移动边缘计算 (mobile edge computing, MEC) 是一种高效的技术, 通过将计算密集型任务从移动设备卸载到边缘服务器, 使终端用户实现高带宽、低时延的目标. 移动边缘计算环境下的计算卸载在减轻用户负载和增强终端计算能力等方面发挥着重要作用. 考虑了服务缓存, 提出一种云-边-端协同的计算卸载框架, 在该框架中引入D2D (device-to-device, D2D) 通信和机会网络. 基于建立的模型, 将计算卸载决策问题转化为一个混合整数非线性规划问题, 并对无线特性和移动用户之间的非合作博弈交互制定了一个迭代机制来共同确定计算卸载方案. 对提出的计算卸载算法从理论上证明了多用户计算卸载博弈模型为严格势力场博弈 (exact potential game, EPG), 卸载决策可获得全网范围内的最优效益. 考虑到服务器的计算资源、卸载任务数据量和任务延迟需求, 提出对用户和MEC 服务器之间最佳用户关联匹配算法. 最后, 模拟结果表明, 卸载决策算法具有较快的收敛速度, 并在能效方面优于其他基准算法.

关键词: 移动边缘计算; 计算卸载; 用户关联匹配; 服务质量; 严格势力场博弈

中图法分类号: TP393

中文引用格式: 张祥俊, 伍卫国, 张弛, 柴玉香, 杨诗园, 王雄. 面向移动边缘计算网络的高能效计算卸载算法. 软件学报, 2023, 34(2): 849–867. http://www.jos.org.cn/1000-9825/6417.htm

英文引用格式: Zhang XJ, Wu WG, Zhang C, Chai YX, Yang SY, Wang X. Energy-efficient Computing Offloading Algorithm for Mobile Edge Computing Network. Ruan Jian Xue Bao/Journal of Software, 2023, 34(2): 849–867 (in Chinese). http://www.jos.org.cn/1000-9825/6417.htm

Energy-efficient Computing Offloading Algorithm for Mobile Edge Computing Network

ZHANG Xiang-Jun[1,2], WU Wei-Guo[1,2], ZHANG Chi[1], CHAI Yu-Xiang[1], YANG Shi-Yuan[1], WANG Xiong[1]

[1](School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China)

[2](National High Performance Computing Center (Xi'an), Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: Mobile edge computing (MEC) is an efficient technology that enables end users to achieve the goal of high bandwidth and low latency by offloading computationally intensive tasks from mobile devices to edge servers. Computing offloading in the mobile edge computing environment plays an important role in reducing user load and enhancing terminal computing capabilities. This study considers service caching, and proposes a cloud-side-end collaborative computing offloading framework, in which D2D communication and opportunistic networks are introduced. Based on the established model, the offloading decision problem is transformed into a mixed integer nonlinear programming problem, and an iterative mechanism is formulated for the non-cooperative game interaction between wireless characteristics and mobile users to jointly determine the computational offloading plan. The proposed computational offloading algorithm theoretically proves that the multi-user computational offloading game under this framework is an exact potential game (EPG), and the offloading decision is to uninstall under the optimal benefit strategy in the entire network. Taking into account the computing resources of the server, the amount of data for offloading tasks, and the delay requirements of tasks, based on the Gale-Shapley matching theory, the

* 基金项目: 国家重点研发计划 (2016YFB0201800, 2017YFB0203003)
收稿时间: 2021-05-06; 修改时间: 2021-05-23, 2021-06-20; 采用时间: 2021-07-05; jos 在线出版时间: 2022-07-15
CNKI 网络首发时间: 2022-11-15

best user association matching algorithm is improved and proposed. Finally, the simulation results show that the proposed unloading decision algorithm has a faster convergence rate and is superior to other benchmark algorithms in terms of energy efficiency.

Key words: mobile edge computing; computation offloading; user association matching; quality of service; exact potential game

1 引 言

在先进的 5G 蜂窝系统高速发展驱动下, 各种新兴的资源需求型、时延敏感型移动应用不断涌现. 例如数据流、实时视频、3D 游戏等, 这些新兴的应用为人们的生活带来了极大的便利. 然而, 随着业务逐渐复杂和多样化, 移动网络的流量呈指数级增长. 传统的集中式网络架构由于回程链路负载过重、时延较长, 无法满足移动用户的需求[1,2]. 据 IDC 预测, 到 2025 年底将有超过 416 亿终端设备联网, 其中超过 50% 的数据需要在网络边缘分析、处理与存储[3]. 而传统的“云-端二体协同计算”模式已无法满足低时延、高带宽的需求. 移动边缘计算 (mobile edge computing, MEC) 是将网络能力从核心网下放至边缘网络的新型体系结构, 使移动终端能够将计算负载转移到边缘服务器, 为这一问题提供了一种有效的解决方案[4] . 由于接近终端设备, MEC 不仅减轻了云中心的处理压力, 而且节省了端到云的高带宽成本, 降低了端到边缘节点的网络响应时延[5,6].

计算卸载作为移动边缘计算中一种增强移动设备计算能力的技术已经崛起. 它能够打破移动设备资源限制, 拓展移动设备计算能力、电池容量和存储能力等[7]. 通过将移动设备的计算任务卸载到边缘服务器中执行, 达到增强移动设备的计算能力、缩短服务时延、节省移动设备能耗的目的. 然而, 在动态变化的网络环境中, 移动用户很难实时的做出细粒度的计算卸载决策, 从而无法最大化降低系统卸载开销[8,9]. 一方面, 由于移动用户与边缘服务器之间的无线链路具有实施不确定性; 另一方面, 相比于云服务器, 边缘服务器在计算、存储等方面的资源仍相对有限[10]. 近年来, 基于 MEC 的计算卸载问题得到了越来越多的关注, 学界提出了各种计算卸载策略. 现有的工作大多集中在小型同步 MEC 系统的卸载设计[11−15], 卸载策略问题通常被表述为一个静态的凸或非凸优化问题, 并将能源消耗和时延做为主要的性能指标. 例如, Tan 等人[16]研究了边缘计算系统中的可持续计算卸载. 他们提出在线奖励-最优拍卖的策略来优化卸载任务的长期奖励总和, 同时适应高度动态的能量收集过程和计算任务到达. 针对大规模异步 MEC 系统场景, Liu 等人[17]设计了一种基于索引的实时任务卸载策略, 该方法可捕获随机临界性的任务. 文献 [18] 提出基于混合非正交倍数接入 (non-orthogonal multiple access, NOMA)-正交多址 (orthogonal multiple access, OMA) 传输策略, 采用逐次凸逼近技术设计了一种求解复杂非凸问题的有效算法.

然而, 上述工作大多在卸载决策前认定信道状态信息是已知和静态的, 这意味着无线信道的相干时间远远大于应用程序的执行时间[19,20]. 另外, 为了提高频谱效率和增强边缘用户的覆盖, 网络部署趋向于密集化, 这大大增加了不同接入单元 (计算接入点或微基站) 的密度和规模, 导致传统卸载方法计算复杂度增大, 在信道相干时间内难以求解, 并且大多数方法仅考虑用户和运营商单方面的效益. 与此同时, 为实现用户邻近性并提高服务质量, 访问单元集成移动边缘服务器 (MEC servers, MESs) 也趋向于密集, 所以合理的用户关联也十分重要. 为此, 需要设计快速、多用户的计算卸载机制, 同时考虑符合实际计算资源和任务延迟场景下的用户关联决策.

本文考虑了多用户的 MEC 场景 (如图 1 所示), 我们设计了高能效的计算卸载算法进行卸载决策, 同时寻求最佳的用户关联匹配. 特别是在卸载决策问题中, 引入机会网络做服务缓存, 并使用博弈方法分析建立了一个全网最优效益的计算卸载算法. 最后, 模拟结果表明, 提出的计算卸载算法与基准算法相比始终保持了最小化能耗和延迟, 且具有良好的性能和时间复杂度特性. 综上所述, 本文的贡献如下.

(1) 本文提出了一个具有 MEC 密集部署的计算卸载系统模型, 其中小基站 (small base stations, SBSs) 和边缘计算节点都配备了 MESs. 在边缘网络计算卸载过程结合了服务缓存, 显著地降低了延迟和能耗, 提升了系统的整体效率.

(2) 提出了云-边-端协同的计算卸载框架, 引入 D2D 通信和机会网络的计算卸载场景, 并结合服务缓存, 更好适配了 5G 蜂窝系统下的 MEC 应用场景. 从而使云计算中心强大计算能力和边缘云距离终端设备的邻近性优势互补. 另外, 对不同卸载模式建立了能耗和时延模型, 将效益最优化问题转化成混合整数非线性规划问题, 该问题属于 NP-hard.

(3) 对多用户计算卸载问题, 每个用户是理性的 (为收益最大化而调整卸载策略), 采用博弈方法可保证全网每

个用户的收益最大化, 并通过数学推导证明了多用户计算卸载问题为严格势力场博弈. 最后设计了基于博弈理论的计算卸载算法, 并分析算法收敛性和时间复杂度. 此外, 在卸载决策基础上, 改进基于 Gale-Shapley 的匹配算法, 设计了最佳用户关联匹配算法.

(4) 在模拟实验中, 我们考虑了一个实际的场景, 其中部署多个用户和 SBSs, 并与其他基准算法进行比较, 验证了该算法的高能效.

本文第 2 节总结相关工作. 在第 3 节中, 我们详细描述系统模型和问题形式化. 第 4 节给出多用户联合计算卸载和用户关联匹配算法. 第 5 节讨论计算卸载和关联匹配算法的实验结果和性能评价. 最后, 第 6 节对本文进行总结.

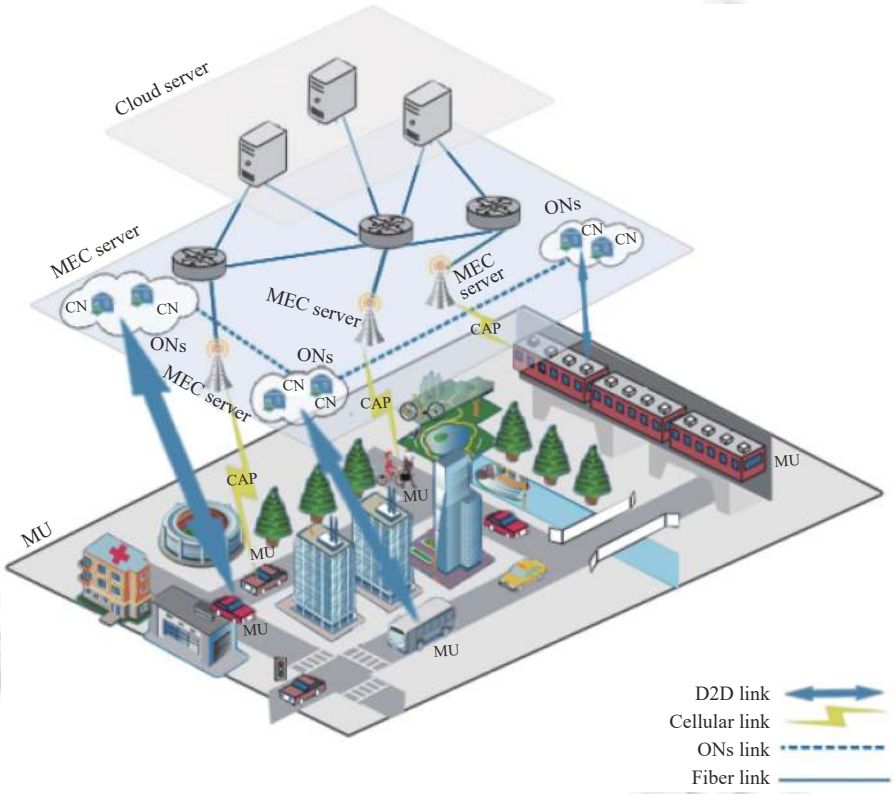


图 1 联合计算卸载模型

2 相关工作

移动边缘计算 (MEC) 作为物联网和 5G 的一项关键使能技术, 可显著减少核心网络拥塞导致的延迟. 计算卸载是近年来 MEC 中的一个关键问题和研究热点[21,22], 例如, 文献 [23] 提出了一个分时和计算时限约束下的总能耗最小化优化方法, 制定了对具有异构输入数据到达时间和计算截止时间的资源管理策略. 在文献 [24] 中, 设计了一种混合光纤-无线网络, 为集中云和多接入边缘计算的共存提供支持, 并提出了采用光纤-无线接入网络的架构和卸载方案. 为了提高效率, 文献 [25−30] 集中于二进制全卸载策略的研究, 以最小化延迟或能耗. Guo 等人[31] 综合考虑了卸载决策、信道分配以及计算资源分配方案的优化. 通过结合遗传算法和粒子群算法的优点改进卸载算法性能和收敛速度, 提出了一种基于遗传算法和粒子群算法的次优算法. 然而, 这些工作只关注无线接入网的计算卸载过程, 忽略了无线接入网信道信息的动态变化和干扰. 针对多用户计算卸载问题, 文献 [32] 提出了一个综合效益最优的 D2D 资源共享方案, 该方案可显著改善卸载网络的性能. 文献 [33] 利用 D2D 卸载计算、并结合用户行为和特定网络运行条件, 开发了大规模 D2D 支持的蜂窝网络 ESE 评估框架. 为分析信噪比分布和平均速率,

文献 [34] 提出了一种 D2D 通信框架. 此外, 文献 [35] 中, 作者采用了频谱共享, 并提供了两种不同共享模式的覆盖性能比较分析.

最近的一些工作致力于采用分散式的计算卸载方案. 例如, 文献 [36] 提出一个以共同确定计算卸载方案、传输调度规则和定价规则的博弈算法. 该算法可使全网范围内的效益最大化, 同时实现移动用户之间的博弈均衡. Quang 等人[37]针对边缘节点对通信和计算资源争用问题提出了一种非合作的具体潜在博弈 (noncooperative exact potential game), 旨在最大化计算卸载的长期效益. Zhang 等人[38]采用进化博弈论 (evolutionary game) 优化子载波分配方案, 为信道状态较好的子载波分配更多的子载波, 使能量效率最优. 在文献 [39] 中研究了一种 MEC 网络中新的计算卸载问题, 移动设备可以通过 D2D 通信或 MEC 服务器将任务卸载到分布式计算节点上, 并将移动设备的卸载决策问题推导为一个序列博弈. 然而, 文献 [39] 没有到考虑服务缓存, 即在分布式计算节点和边缘服务器[40]上没有缓存与计算任务运行所依赖的二进制文件. 由于边缘服务器的计算和存储资源有限, 会导致不可避免的影响服务质量.

鉴于此, 我们对多接入 MEC 的计算卸载进行了新的研究. 与已有工作关注点不同的是, 本文研究了异构网络环境下多用户计算卸载和传输调度的能耗和时延最小化问题, 引入 D2D 通信和机会网络到 MEC 网络, 并结合服务缓存. 为克服边缘节点资源不足的缺陷, 我们设计了云-边-端协同的计算卸载框架, 既保证了边缘服务器与终端设备的邻近性, 又利用了云端的强大计算和存储资源. 本文集中于研究在云-边-端协同的计算卸载问题上使用博弈方法, 相比其他传统方法, 所设计的基于多用户卸载问题的博弈卸载算法和用户关联匹配算法, 可在信道相干时间内快速求解组合优化问题, 并综合考虑了全网效益最优的卸载策略. 数值计算结果表明, 与现有的优化方法相比, 该算法在计算时间上显著减少, 且性能接近最优.

3 系统模型及问题形式化

3.1 网络模型

如图 1 所示, 我们提出了一个云-边-端协同的计算卸载框架, 与已有卸载框架不同, 所提出的框架是结合了 D2D 通信和机会网络, 并考虑计算任务服务缓存的联合计算卸载模型. 该框架 3 层如下.

(1) 云服务器层: 该层是由各类资源充足的云节点组成, 可提供集中式的云服务和缓存终端用户任务依赖的二进制文件功能.

(2) 边缘服务层: 在本层中, 接受计算的服务器包括多接入 MEC 服务器和具备一定计算能力的计算节点 (computing node, CN). MEC 服务器被附加到蜂窝通信基站中, 移动设备连接到 MEC 服务器和计算节点分别通过蜂窝链路和 D2D 链路, 这两条链路之间由于频率不同, 相互隔离, 互不干扰.

(3) 移动设备层: 该层由各类移动终端组成, 任意移动设备的计算密集型任务都可以在本地运行或卸载到边缘服务层进行计算. 边缘服务层缓存了计算任务所依赖的二进制文件以确保卸载任务正确执行. 注意, 当 MEC 服务器和计算节点存储空间不够, 未能缓存所需的二进制文件时, 可通过核心网向云服务器层或机会网络缓存所需的文件.

考虑移动边缘计算环境是由 N 个移动用户 MUs (mobile user, MU) 和 K 个计算接入点 CAPs (computational access points, CAPs) 构成的多用户移动边缘计算系统, 我们分别用 $N=\{1, 2, \ldots, i, \ldots, N\}$ 和 $K=\{1, 2, \ldots, k, \ldots, K\}$ 表示, 这里的接入点可以是具有计算能力的基站和移动边缘计算服务器等. 定义 $\Gamma=\{l_1, l_2, \ldots, l_j, \ldots, l_J\}$ 表示所有的任务集, j 表示计算任务的类型, 每个移动设备运行着不同类型的计算任务, 如计算密集型和延迟敏感的应用程序. 每个移动设备 i 上的 j 类型的计算任务 $L_i^j=\{B_i^{j,\text{in}}, B_i^{j,\text{out}}, C_i^j, D_i^j, T_{i,\text{MAX}}^j\}, i\in N, j\in J$. 其中, $B_i^{j,\text{in}}$ 是计算任务的输入数据, $B_i^{j,\text{out}}$ 是返回的计算结果, C_i^j 是计算任务所需的 CPU 周期数, D_i^j 是计算任务所需依赖的二进制文件大小, $T_{i,\text{MAX}}^j$ 是任务 L_i^j 的最大可接受时延. 每个 MU i 要么在本地执行计算任务 L_i^j, 要么完全卸载任务到 MEC 服务器执行.

3.2 通信模型

在 MEC 系统中, 移动设备可以通过计算无线接入点将计算任务卸载到边缘服务器[41]. 我们定义每个移动用户 MU n 的设备传输速率为 V_n, 表示为:

$$V_n = W_k \log_2\left(1+\frac{p_n h_{n,k}}{w_0+\sum\limits_{m\in N\setminus\{n\}} p_m h_{m,k}}\right) \tag{1}$$

本文考虑了通信的链路之间的相互干扰. 在上式中, p_n 是 MU n 的传输功率, W_k 是信道带宽, w_0 是信道的背景噪音功率, $h_{n,k}$ 是 MU n 和计算接入点 k 间的信道增益, 是一个独立同分布的随机变量[42], 可由公式 (2) 得出.

$$h_{n,k} = g_0\left(\frac{d_0}{d}\right)^{\xi} \tag{2}$$

其中, ξ 是路径损耗指数, d_0 是参考距离, d 是移动设备与 MEC 服务器的距离, g_0 是路径损耗常数.

3.3 计算模型

考虑到服务缓存, 我们在卸载模型中引入了 D2D 通信和机会网络 (opportunistic networks, ONs). 机会网络是一种自组织网络, 它引入节点移动带来的相遇机会从而实现通信, 使通信不再依赖于源节点与目标节点之间的完整链路. 其次, 当通信双方距离较近时, 移动终端可以直接利用 D2D 技术进行通信, 节省了大量信令带宽资源, 并减少了传输延迟. 由于路径损耗远低于基站到 MU 之间的通信损耗, D2D 技术可以提高网络信道的频谱效率[42−44]. 每个 MU 决定是否将其计算任务卸载到 K 个计算接入点中任意一个, 我们定义 MU i 的 j 类计算任务的卸载类型和链路传输数据时延分别为 $a_i^j=\{0,1,2,3,4\}$、T^{off}, 使用指示函数 II 表示不同卸载模式, 具体如下.

(1) 本地计算: 当 $a_i^j=0$, , 计算任务 L_i^j 通过 MU i 的本地计算资源执行任务. 定义 MU 的设备 CPU 周期频率为 f_{MU}. 令 $t_{L_i^j}^{\text{local}}$ 表示本地执行任务的时间. 可由公式 (3) 给出:

$$t_{L_i^j}^{\text{local}} = \frac{C_{L_i}}{f_{\text{MU}}} \tag{3}$$

$$T_{\text{local}}^{\text{off}} = 0 \tag{4}$$

(2) MEC 服务器计算: 考虑到服务缓存, 卸载任务到 MEC 服务器分两种情况. 一方面, 当 $a_i^j=1$, 计算任务 L_i^j 将卸载到与 SBSs 相连的 MEC 服务器上, 并且 MEC 服务器已缓存了计算任务所依赖的二进制文件. 我们用 $t_{L_i^j}^{\text{MEC}}, t_{L_i^j}^{\text{Tran}}, t_{L_i^j}^{\text{Comp}}, t_{L_i^j}^{\text{Res}}, T_{\text{MEC}}^{\text{off}}$ 分别表示卸载任务执行总耗时、传输时延、计算耗时、接收返回结果时延和数据总的链路传输时间. 令 MEC 的 CPU 周期频率、光纤传输速率、MU 与 SBSs 之前链路的传输速率、传输功率、传输带宽、信道增益和噪音功率分别为 $f_{\text{MEC}}, c, V_{L_n^j}^{\text{MEC}}, p_{\text{MEC,tran}}^i, W_{\text{MEC}}, h_{m,k}, w_0$, 则传输速率和总时延为:

$$V_{L_i^j}^{\text{MEC}} = W_{MEC}\log_2\left(1+\frac{p_{\text{MEC,tran}}^i h_{i,k}}{w_0+\sum\limits_{m\in N\setminus\{n\}} II(a_i^j=1)p_{\text{MEC,tran}}^m h_{m,k}}\right) \tag{5}$$

$$t_{L_i^j}^{\text{MEC}} = t_{L_i^j}^{\text{Tran}} + t_{L_i^j}^{\text{Comp}} + t_{L_i^j}^{\text{Res}} = \frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{MEC}}} + \frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{c} + \frac{C_i^j}{f_{\text{MEC}}} \tag{6}$$

其中, $\dfrac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{c}$ 为卸载任务从基站通过光纤接入到 MEC 服务器的传输链路时延. 则卸载的时延为:

$$T_{\text{MEC}}^{\text{off}} = t_{L_i^j}^{\text{Tran}} + t_{L_i^j}^{\text{Res}} = \frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{MEC}}} + \frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{c} \tag{7}$$

另一方面, 当 $a_i^j=2$, 即接受计算任务的 MEC 服务器没有缓存卸载任务 L_i^j 计算所依赖的二进制文件, 此时需要通过核心网访问远程云中心层下载所需二进制文件, 同时更新缓存内容, $t_{L_i^j}^{\text{Cloud}}$ 和 $t_{D_i^j}^{\text{Cloud}}$ 分别代表总的时延和从远程云服务中心下载所需文件的时延, 则总时延为:

$$t_{L_i^j}^{\text{Cloud}} = t_{L_i^j}^{\text{Tran}} + t_{L_i^j}^{\text{Comp}} + t_{L_i^j}^{\text{Res}} + t_{D_i^j}^{\text{Cloud}} = \frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{MEC}}} + \frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{c} + \frac{C_i^j}{f_{\text{MEC}}} + \frac{D_i^j}{c} \tag{8}$$

$$T_{\text{Cloud}}^{\text{off}}=t_{L_i^j}^{\text{Tran}}+t_{L_i^j}^{\text{Res}}+t_{D_i^j}^{\text{Cloud}}=\frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{MEC}}}+\frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}+D_i^j}{c} \tag{9}$$

(3) D2D 卸载到计算节点: 同理, D2D 卸载分为两种情况. 当 $a_i^j=3$, 计算任务 L_i^j 通过 D2D 链路卸载到邻近的计算节点 (如 Cloudlet, 微数据中心等), 同时该计算节点已缓存了卸载任务所依赖的二进制文件, D2D 链路与 MU 之间的传输速率、传输带宽, 传输功率, 信道增益, 噪音功率, 数据总的链路传输时间分别为 $V_{L_n^j}^{\text{D2D}}, W_{\text{D2D}}, p_{\text{D2D,tran}}^i$, $h_{i,u_n}, w_0, T_{\text{D2D}}^{\text{off}}$, 由以下公式得出.

$$V_{L_i^j}^{\text{D2D}}=W_{\text{D2D}}\log_2\left(1+\frac{p_{\text{D2D,tran}}^i h_{i,u_n}}{w_0+\sum\limits_{m\in N\setminus\{n\}} II(a_i^j=3)p_{\text{D2D,tran}}^m h_{m,u_n}}\right) \tag{10}$$

$$t_{L_i^j}^{\text{D2D}}=t_{L_i^j}^{\text{Tran}}+t_{L_i^j}^{\text{Comp}}+t_{L_i^j}^{\text{Res}}=\frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{D2D}}}+\frac{C_i^j}{f_{\text{D2D}}} \tag{11}$$

$$T_{\text{D2D}}^{\text{off}}=t_{L_i^j}^{\text{Tran}}+t_{L_i^j}^{\text{Res}}=\frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{D2D}}} \tag{12}$$

当 $a_i^j=4$, 即卸载的计算节点没有缓存卸载任务计算依赖的二进制文件, 类似文献 [45], 此时需要通过机会网络链接任务运行所依赖的文件并下载他们, 机会网络不需要源节点和目的节点之间存在完整路径, 利用节点移动带来的相遇机会实现网络通信, 之后更新计算节点的缓存. 令传输速率、传输带宽、 传输功率、信道增益、 噪音功率、数据总的链路传输时间分别用 $V_{\text{load}}^{\text{ONs}}, W_{\text{ONs}}, p_{\text{load}}^{\text{ONs}}, h_{n,u_n}, w_0, T_{\text{ONs}}^{\text{off}}$ 表示, 则传输速率和总时延为:

$$V_{\text{load}}^{\text{ONs}}=W_{\text{ONs}}\log_2\left(1+\frac{p_{\text{load}}^n h_{n,u_n}}{w_0}\right) \tag{13}$$

$$t_{L_i^j}^{\text{ONs}}=t_{L_i^j}^{\text{Tran}}+t_{L_i^j}^{\text{Comp}}+t_{L_i^j}^{\text{Res}}=\frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{D2D}}}+\frac{C_i^j}{f_{\text{D2D}}}+\frac{D_i^j}{V_{\text{load}}^{\text{ONs}}} \tag{14}$$

$$T_{\text{ONs}}^{\text{off}}=t_{L_i^j}^{\text{Tran}}+t_{L_i^j}^{\text{Res}}=\frac{B_i^{j,\text{in}}+B_i^{j,\text{out}}}{V_{L_i^j}^{\text{D2D}}}+\frac{D_i^j}{V_{\text{load}}^{\text{ONs}}} \tag{15}$$

则根据不同卸载方式下, 总的时延表示为:

$$T^{\text{off}}=\{T_{\text{local}}^{\text{off}}, T_{\text{MEC}}^{\text{off}}, T_{\text{Cloud}}^{\text{off}}, T_{\text{D2D}}^{\text{off}}, T_{\text{ONs}}^{\text{off}}\} \tag{16}$$

综合以上不同的卸载类型, MU 可通过无线传输向 CAP 发送数据时, 可以调整传输功率. 因此, MU 的卸载策略 s_n 可表示为:

$$s_n=\{P_n, a_n\} \tag{17}$$

其中, $P_n\in[0,P_{\max}]$, 分别表示设备的发射功率和策略集, 则策略集 $S=(s_1,s_2,\ldots,s_n)$.

3.4 能耗模型

针对计算模型提出的 5 种卸载类型, 本节将分别对不同卸载类型建立能耗模型, 定义 $e^i, e_{\text{MEC}}^i, e_{\text{D2D}}^i$ 分别表示计算任务在本地、MEC、D2D 环境下每个 CPU 周期消耗的能量.

(1) 本地计算能耗: 当 $a_i^j=0$, 则 MU i 的 j 类型任务在本地执行的能耗 $e_{L_i^j}^{\text{local}}$ 为:

$$e_{L_i^j}^{\text{local}}=C_i^j e^i \tag{18}$$

结合公式 (3) 的本地计算总时延, 令 α 和 β 分别表示计算任务 L_i^j 的时延和能耗权重系数, 且 $\alpha+\beta=1$. 可以得出本地执行任务的总开销 $E_{L_i^j}^{\text{local}}$ 为:

$$E_{L_i^j}^{\text{local}}=\alpha t_{L_i^j}^{\text{local}}+\beta e_{L_i^j}^{\text{local}} \tag{19}$$

(2) MEC 服务器卸载能耗: 当 $a_i^j=1$, 则卸载到 MEC 服务器执行任务的能耗 $e_{L_i^j}^{\text{MEC}}$ 为:

$$e_{L_i^j}^{\mathrm{MEC}}=p_{\mathrm{MEC,tran}}^{i}\frac{B_i^{j,\mathrm{in}}+B_i^{j,\mathrm{out}}}{V_{L_i^j}^{\mathrm{MEC}}}+\frac{B_i^{j,\mathrm{in}}+B_i^{j,\mathrm{out}}}{c}+C_i^j e_{\mathrm{MEC}}^{i} \tag{20}$$

结合公式 (6) 的 MEC 卸载总时延 $t_{L_i^j}^{\mathrm{MEC}}$, 可以得出总开销 $E_{L_i^j}^{\mathrm{MEC}}$ 为:

$$E_{L_i^j}^{\mathrm{MEC}}=\alpha t_{L_i^j}^{\mathrm{MEC}}+\beta e_{L_i^j}^{\mathrm{MEC}} \tag{21}$$

同理, 当 $a_i^j=2$, 此时 MEC 服务器通过访问远程云中心下载任务所依赖的二进制文件后执行任务, 则总能耗 $e_{L_i^j}^{\mathrm{Cloud}}$ 为:

$$e_{L_i^j}^{\mathrm{Cloud}}=p_{\mathrm{MEC,tran}}^{i}\frac{B_i^{j,\mathrm{in}}+B_i^{j,\mathrm{out}}}{V_{L_i^j}^{\mathrm{MEC}}}+\frac{B_i^{j,in}+B_i^{j,\mathrm{out}}}{c}+C_i^j e_{\mathrm{MEC}}^{i}+\frac{D_i^j}{c} \tag{22}$$

结合公式 (8) 的访问云主机的卸载总时延 $t_{L_i^j}^{\mathrm{Cloud}}$, 可以得出总开销 $E_{L_i^j}^{\mathrm{Cloud}}$ 为:

$$E_{L_i^j}^{\mathrm{Cloud}}=\alpha t_{L_i^j}^{\mathrm{Cloud}}+\beta e_{L_i^j}^{\mathrm{Cloud}} \tag{23}$$

(3) D2D 卸载能耗: 类似 D2D 计算模型, 当 $a_i^j=3$, 则通过 D2D 链路卸载的能耗 $e_{L_i^j}^{\mathrm{D2D}}$ 为:

$$e_{L_i^j}^{\mathrm{D2D}}=p_{\mathrm{D2D,tran}}^{i}\frac{B_i^{j,\mathrm{in}}+B_i^{j,\mathrm{out}}}{V_{L_i^j}^{\mathrm{D2D}}}+e_{\mathrm{D2D}}^{i}C_i^j \tag{24}$$

结合公式 (11) 的 D2D 卸载总时延 $t_{L_i^j}^{\mathrm{D2D}}$, 可以得出总开销 $E_{L_i^j}^{\mathrm{D2D}}$ 为:

$$E_{L_i^j}^{\mathrm{D2D}}=\alpha t_{L_i^j}^{\mathrm{D2D}}+\beta e_{L_i^j}^{\mathrm{D2D}} \tag{25}$$

同理, 当 $a_i^j=4$, 此时, 通过机会网络下载文件后执行计算卸载的能耗为 $e_{L_i^j}^{\mathrm{ONs}}$ 为:

$$e_{L_i^j}^{\mathrm{ONs}}=\frac{p_{\mathrm{D2D,tran}}^{i}B_i^{j,\mathrm{in}}+p_{i,\mathrm{rec}}^{i}B_i^{j,\mathrm{out}}}{V_{L_i^j}^{\mathrm{D2D}}}+e_{\mathrm{D2D}}^{i}C_i^j+\frac{p_{\mathrm{load}}^{n}D_i^j}{V_{\mathrm{load}}^{\mathrm{ONs}}} \tag{26}$$

结合公式 (14) 的访问机会网络 ONs 卸载的总时延 $t_{L_i^j}^{\mathrm{ONs}}$, 可以得出总开销 $E_{L_i^j}^{\mathrm{ONs}}$ 为:

$$E_{L_i^j}^{\mathrm{ONs}}=\alpha t_{L_i^j}^{\mathrm{ONs}}+\beta e_{L_i^j}^{\mathrm{ONs}} \tag{27}$$

综合以上 5 种卸载类型, 总的开销 $E(S)$ 为:

$$E(S)=\{E_{L_i^j}^{\mathrm{local}},E_{L_i^j}^{\mathrm{MEC}},E_{L_i^j}^{\mathrm{Cloud}},E_{L_i^j}^{\mathrm{D2D}},E_{L_i^j}^{\mathrm{ONs}}\} \tag{28}$$

3.5 效益函数和问题形式化

通过以上两节建立的计算和能耗模型, 我们可以给出每个 MU 的效益函数 $\chi_n(S)$ 模型. 首先令 $D_n^l, D_n^{\mathrm{OFF}}$ 分别表示计算任务在本地执行和卸载到边缘服务器执行所需要的 CPU 周期数. 类似文献 [46], 假设卸载任务执行所需 CPU 周期数与卸载数据比特数成正比, 则:

$$D_n^l=II(a_n^j=0)C_i^j \tag{29}$$

$$D_n^{\mathrm{OFF}}=II(a_i^j\neq 0)\theta T^{\mathrm{off}}V_i=\theta T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_i^j\in\{1,2,3,4\}}}W_k\log_2\left(1+\frac{a_n^j p_n h_{n,k}}{w_0+\sum_{m\in N\setminus\{n\}}a_m^j p_m h_{m,k}}\right) \tag{30}$$

其中, II 为指示函数, 表示不同的卸载模式. θ 为计算数据比, 即处理每 bit 数据所需消耗的 CPU 周期数, 则执行任务总的 CPU 周期数 D_n 表示为:

$$D_n(S)=D_n^l+D_n^{\mathrm{OFF}}=II(a_n^j=0)C_i^j+\theta T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}}W_k\log_2\left(1+\frac{a_n^j p_n h_{n,k}}{w_0+\sum_{m\in N\setminus\{n\}}a_m^j p_m h_{m,k}}\right) \tag{31}$$

由于在多接入 MEC 系统中, 参与计算卸载的每个用户是理性的, 也就是说, 每个用户都愿意通过增大各自的传输功率来卸载计算密集型任务到边缘云处理, 即最大化已处理 CPU 周期数, 最终导致整个传输信道性能下降.

鉴于此, 每个 MU 的效益函数必须仔细考虑能耗和性能[47]. 在实际情况下, 能源消耗是有成本的, 本文考虑了一种能源感知的效益函数模型$\chi_n(S)$为:

$$\chi_n(S)=\mu_1 D_n(S)-\mu_2 E_n(S) \tag{32}$$

该效益函数模型可反映每个 MU 的卸载收益, 综合考虑了能耗和时延. 因此, 每个 MU 的任务卸载决策可形式化为联合优化问题 P1:

$$\begin{cases}(\mathrm{P1})\ \forall n\in N, \max \chi_n(S)\\ \text{s.t.}\begin{cases}\leqslant t_{L_i^j}\leqslant T_{i,\mathrm{MAX}}^j\ (33a)\\ a_i^j\in\{0,1,2,3,4\}\ (33b)\\ 0<P_i\leqslant P_{\max}\ (33c)\\ 0<P_i^{\mathrm{MEC}}\leqslant P_{\max}^{\mathrm{MEC}}\ (33d)\\ \sum_{i\in N} D_i\leqslant D_{\mathrm{MEC}}\ (33e)\\ i\in N, j\in J\ (33f)\end{cases}\end{cases} \tag{33}$$

公式 (33) 中的约束 33a 表示任务最大能容忍时延. 33b 表示卸载类型, 33c 表示每个 MU 的发射功率不得超过设备最大功率, 33d 限定了每个边缘服务器功率不得超过最大功率. 33e 限定了缓存任务运行必备的二进制文件大小不应超过 MEC 服务器最大存储容量D_{MEC}. 相应的 33f 约束了设备和计算任务类型的范围. 该效益函数全面反映每个 MU 的整体收益, 既体现卸载任务带来的收益, 又考虑了能耗的影响. 注意, 效益函数的推导过程如公式 (34) 所示.

$$\begin{aligned}&\chi_n(s)=\mu_1 D_n(S)-\mu_2 E_n(S)\\&=\mu_1\left(II(a_n^j=0)C_n^j+\theta T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(1+\frac{a_n^j p_n h_{n,k}}{w_0+\sum_{m\in N\setminus\{n\}} a_m^j p_m h_{m,k}}\right)\right)-\mu_2 E_{L_n^j}(S)\\&=\theta\mu_1 T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(w_0+\sum_{m\in N} a_m^j p_m h_{m,k}\right)+\mu_1 II(a_n^j=0)C_n^j-\mu_2 E_{L_n^j}(S)\\&\quad-\theta\mu_1 T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(w_0+\sum_{m\in N\setminus\{n\}} a_m^j p_m h_{m,k}\right)\end{aligned} \tag{34}$$

$$\begin{aligned}\chi_n(S_n',S_{-n})-\chi_n(S_n'',S_{-n})=&\left(\theta\mu_1 T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(w_0+\sum_{m\in N\setminus\{n\}} a_m p_m h_{m,k}+a_n' p_n' h_{n,k}\right)\right.\\&\left.-\theta\mu_1 T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(w_0+\sum_{m\in N\setminus\{n\}} a_m p_m h_m+a_n'' p_n'' h_{n,k}\right)\right)\\&-[\mu_2 E_{L_n^j}(s_n)'-\mu_2 E''_{L_n^j}(s_n)]+[\mu_1 II(a_n')C_n^j-\mu_1 II(a_n'')C_n^j]\end{aligned} \tag{35}$$

$$\begin{aligned}\varphi(S_n',S_{-n})-\varphi(S_n,S''_{-n})=&\left(\theta\mu_1 T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(w_0+\sum_{m\in N\setminus\{n\}} a_m p_m h_{m,k}+a_n' p_n' h_{n,k}\right)\right.\\&\left.-\theta\mu_1 T^{\mathrm{off}}\sum_{\substack{k\in K\\ a_n^j\in\{1,2,3,4\}}} W_k\log_2\left(w_0+\sum_{m\in N\setminus\{n\}} a_m p_m h_{m,k}+a_n'' p_n'' h_{n,k}\right)\right)\\&-\left[\mu_2\sum_{m\in N\setminus\{n\}} E_{L_n^j}(S_m)+\mu_2 E_{L_n^j}(S_n)'-\left(\mu_2\sum_{m\in N\setminus\{n\}} E_{L_n^j}(s_m)+\mu_2 E''_{L_n^j}(s_n)\right)\right]\\&+[\mu_1 II(a_n')C_n^j-\mu_1 II(a_n'')C_n^j]\end{aligned} \tag{36}$$

4 多用户联合计算卸载和关联匹配算法

基于每个 MU 的能效模型, 我们设计一种高能效计算卸载算法, 即 EDCO (energy efficient distributed computing offloading) 算法. 采用基于博弈理论为基础的多用户联合计算卸载算法, 并证明了该博弈是一个严格势力场博弈

(exact potential game, EPG). EDCO 算法可在有限次迭代后达到全网最优效益的纳什均衡状态. 同时我们分析了 EDCO 算法的收敛性和时间复杂度. 最后, 针对用户关联匹配问题, 我们受 Gale-Shapley 匹配理论启发, 对其做出修改并改进, 并结合 EDCO 算法, 设计了最佳用户关联匹配算法 (best user association matching algorithm, BUAM).

4.1 博弈模型

与传统单用户计算卸载不同, 在对接入 MEC 系统中, 随机分布的每个参与方是自私的, 为了实现个人效益最大化, 每个用户倾向于调整自己的发射功率来卸载计算密集型任务, 即最大化任务已处理 CPU 周期数来相互竞争. 这势必会导致一个分布式决策问题, 更具体地说, 这是一个多用户计算卸载系统上的非合作博弈. 形式上, 这个博弈可以表示为:

$$G = \{N, A = A_1 \ldots A_n, \chi : AR^N\} \tag{37}$$

其中, N 表示博弈的参与者集合 $i \in N$, A 表示每个参与者的策略空间 S_i, $\forall i \in N$, χ 表示每个参与者的效益函数, 即公式 (33) 定义的形式. 令 $s_{-i} = \{s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\}$ 表示除了用户 i 以外的所有其他参与者的策略组合向量. 则博弈 G 的纳什均衡 (Nash equilibrium, NE) 可定义为:

定义 1. 在博弈 G 中, 如果由每个博弈方的一个策略组成的某个策略组合 $s^* = \{s_1^*, s_2^*, \ldots, s_i^*, \ldots, s_n^*\}$ 中, 任意一个博弈方 MU i 的策略 s_i^* 都是对其余博弈方策略组合 $s_{-i}^* = \{s_1^*, s_2^*, \ldots, s_{i-1}^*, s_{i+1}^*, \ldots, s_n^*\}$ 的最佳对策, 即:

$$\chi_n(s_i^*, s_{-i}^*) \geqslant \chi_n(s_i, s_{-i}^*), \ \forall s_i^* \neq s_i \tag{38}$$

则称 $\{s_1^*, s_2^*, \ldots, s_i^*, \ldots, s_n^*\}$ 为 G 的一个纳什均衡. 显然, 博弈 G 的纳什均衡很大程度上依赖于卸载策略 s_i, 因此, 为了保证所有移动用户都愿意按效益最大化的方式自行管理各自计算卸载策略, 以保证用户自身的效益.

在卸载决策时, 应以全网范围内的最优解为博弈 G 中的 NE, 即:

$$s_i^* = \hat{s}_i, \ \forall i \in N \tag{39}$$

$s_i^*, \hat{s}_i$ 分别是博弈 G 的纳什均衡解和计算卸载效益优化问题 P1 的输出. 为方便分析本文提出的 EDCO 算法收敛性, 在分析其收敛性前, 我们引入收敛性较好的严格势力场博弈. 其中, 严格势力场函数 (exact potential function, EPF) 可反映参与博弈的移动用户策略单独发生改变时效用函数的变化[38].

定义 2. 博弈 G 是一个以 φ 为势函数的严格势力场博弈. φ 的形式如下:

$$\varphi(S) = \theta\mu_1 T^{\text{off}} \sum_{\substack{k \in K \\ a_n^j \in \{1,2,3,4\}}} W_k \log_2\left(W_0 + \sum_{m \in N} a_m^j p_m h_{m,k}\right) + \mu_1 II(a_n^j = 0) C_n^j - \mu_2 \sum_{n \in N} E_{L_n^j}(S_n) \tag{40}$$

证明: 博弈 G 是严格势力场博弈 (EPG), 存在一个势函数 $\varphi : S \to R, \forall S_i', S_i'' \in S_i$, 使下列等式恒成立.

$$\varphi(S_i', S_{-i}) - \varphi(S_i'', S_{-i}) = \chi(S_i', S_{-i}) - \chi(S_i'', S_{-i}) \tag{41}$$

EPG 的证明推导过程如公式 (35) 所示. 其中, S_i', S_i'' 表示 MU n 上一个不同于 s_n 的卸载策略, 也就是说, 当 MU i 从策略 S' 切换到策略 S'' 时, 潜在的变化等于该用户效用值的变化. 根据公式 (34) 效用函数 $\chi_n(S)$ 的形式, 我们可推导出公式 (35). 注意, 公式 (34) 的最后一项只和其他 MU 的策略 S_{-n} 有关, 所以 $\chi(S_i', S_{-i}) - \chi(S_i'', S_{-i})$ 可以消除这一项. 势函数 φ 的详细的推导细节在公式 (36), 类似公式 (35) 推导过程, $\varphi(S_i', S_{-i}) - \varphi(S_i'', S_{-i})$ 中可以消除只与其他 MU 的策略相关的项 S_{-n}. 综合公式 (35) 和公式 (36), 可知公式 (41) 恒成立. 故博弈 G 是一个以函数 φ 为势函数的严格势力场博弈. 综上所述, 如果一种博弈模型满足 EPG 时, 当用户以最佳对策动态连续调整时, 博弈就会收敛到纯纳什平衡点, 并使势函数 φ 取得最大值. 任何异步最佳对策更新过程都必须是有限的, 通过有限异步最佳对策更新过程来达到纳什均衡. 即有以下推论[48]:

推论 1. 对于 EPG 博弈 G, 存在一个纯策略纳什均衡, 并且可以通过有限次的最佳响应过程得到.

4.2 高能效的的分布式计算卸载算法

基于上述博弈理论, 我们设计了一个通过有限次最佳对策得到高效能计算卸载决策的算法 (EDCO algorithm), 算法具体步骤如下:

(1) 初始化卸载策略及关联匹配关系表, 并随机选择一个计算任务 L_i^j, 选中的概率为 $\frac{1}{N}$, 卸载策略 $S_i = \varnothing$;

(2) 在每次迭代中, 计算接入点依次对每个 MU i 进行最佳对策更新 S_i. 具体来讲, 在给定更新后的用户计算

卸载决策$\{s_1^{t+1}, s_2^{t+1}, \ldots, s_{n-1}^{t+1}\}$和在第 t 次迭代中剩余未更新用户的卸载决策$\{s_{n+1}^{t}, s_{n+2}^{t}, \ldots, s_N^{t}\}$情况下, 用户 n 将选择卸载决策$\left\{s_n^{t+1}\right\}$来获取最大化收益. 即:

$$s_n^{t+1} = \underset{a_i \in \{0,1,2,3,4\}}{\operatorname{argmax}} \chi(s, s_1^{t+1}, s_2^{t+1}, \ldots, s_{n-1}^{t+1}, s_{n+1}^{t}, \ldots, s_N^{t}) \tag{42}$$

(3) 根据公式 (17) 所示的 $s_i = \{P_i, a_i\}$, 每个用户抢占更新的概率是均等的, 并且每个用户在更新决策后, 会广播决策 s_n^{t+1}给其他用户, 让其他用户知道当前的卸载决策.

(4) 每个移动设备通过有限次迭代, 异步执行各自的最佳对策更新, 直至达到纳什均衡状态, 得出最佳卸载决策 S. 之后调用最佳用户关联匹配算法 BUAM, 计算最佳关联匹配算法. 注意, BUAM 算法的具体细节见第 4.3 节.

EDCO 算法和 BUAM 算法具体细节如算法 1 和算法 2 所示.

4.2.1 收敛性分析

由于纳什均衡解不一定是最优解, 我们使用无政府代价 (price of anarchy, *PoA*) 来量化纳什均衡最坏的情况[49]. 令 σ 为纳什均衡集, $\max\limits_{S_i^* \in \sigma} \sum\limits_{n \in K} \chi_n(s_i^*)$表示最大化收益. 则 *PoA* 为:

$$PoA = \frac{\max\limits_{S_i^* \in \sigma} \sum\limits_{n \in K} \chi_n(s_i^*)}{\min\limits_{S_i \in S} \sum\limits_{n \in K} \chi_n(s_i)} \tag{43}$$

定理 1. 博弈 G 的 *PoA* 最大值为 $\dfrac{\sum\limits_{n \in K}(\mu_1 D_n^l - \mu_2 E_n^{\text{local}})}{\sum\limits_{n \in K} \min\{\chi_n^{\text{local}}, \chi_n^{\text{off}}\}}$.

证明: 在纳什均衡下, 每个用户 n 的能量消耗最多为 E_{local}, 即选择始终在本地执行计算任务. 因此

$$\sum_{n \in K} \chi_n(s_i^*) \leqslant \sum_{n \in K} \chi_n^{\text{local}} = \sum_{n \in K}(\mu_1 D_n^l - \mu_2 E_n^{\text{local}}) \tag{44}$$

$$\sum_{n \in K} \chi_n(s_i) \geqslant \min\{\chi_n^{\text{local}}, \chi_n^{\text{off}}\} \tag{45}$$

其中, $\chi_n^{\text{off}} = \mu_1 D_n^{\text{OFF}} - \mu_2 E_n(S) = \mu_1 \theta T^{\text{off}} \sum\limits_{\substack{k \in K \\ a_n^j \in \{1,2,3,4\}}} W_k \log_2\left(1 + \dfrac{a_n^j p_n h_{n,k}}{w_0 + \sum_{m \in N \setminus \{n\}} a_m^j p_m h_{m,k}}\right) - \mu_2 E_{L_n^j}(s_i)$, 基于公式 (44), 公式 (45) 可得:

$$PoA \leqslant \frac{\sum\limits_{n \in K}(\mu_1 D_n^l - \mu_2 E_n^{\text{local}})}{\sum\limits_{n \in K} \min\{\chi_n^{\text{local}}, \chi_n^{\text{off}}\}} \tag{46}$$

该结果表明 EDCO 算法具有很好的收敛性, 在有限次迭代后可快速收敛. 实际上, 在后续实验结果也表明了该算法的收敛性明显好于传统基准算法.

算法 1. EDCO.

输入: K, N, μ_1, μ_2, W_{MEC}, W_{D2D}, W_{ONs}, w_0, N_0, $h_{n,k}$, P_{MU}, P_{MEC}, P_{D2D}, P_{ONs}, P_{res}, P_{MAX}, d_0, θ, g_0, f_{MEC}, f_{D2D}, f_{MU}, $e_{L_i^j}^{\text{local}}$, $e_{L_i^j}^{\text{MEC}}$, $e_{L_i^j}^{\text{D2D}}$, $e_{L_i^j}^{\text{ONs}}$ $(0 < i < n)$;
输出: 计算卸载策略 S, 最小能耗 $E_{\min}$, 关联匹配 Θ.

1. 初始化阶段: 被占用的信道数为 k=0, 时间段内所有的移动设备 MU 的卸载决策 $S_i = 0$, $\forall i \in N$, 默认是本地执行, 随机选择一个计算任务 L_i^j, (选中的概率为 $\dfrac{1}{N}$), 卸载策略 S_i=Ø.

2. **while** t<T **do**

3. **for** all $i \in N$ **do**

4. 根据公式 (5), 公式 (7), 公式 (9) 计算 slot t+1 的不同类型设备的传输速率: $V_{L_i^j}^{\mathrm{MEC}}, V_{L_i^j}^{\mathrm{D2D}}, V_{\mathrm{load}}^{\mathrm{ONs}}$, 并根据公式 (33) 以及相应的约束条件计算最大收益的卸载策略 s_i^j.
5. **if** s_i^j 满足约束规则 33a, 33b, 33c, 33d, 33e **then**
6. 更新 s_i^j
7. **end if**
8. **end for**
9. **if** $s^t \neq s^{t+1}$ **and** $S \neq \varnothing$ **then**
10. **if** 如果设备 i 争取到更新机会 **then**
11. 根据公式 (33) 用户选择卸载决策 s_n^{t+1} 最大化自己的收益.
12. $s_i = s_i^{t+1}$
13. **else**
14. $s_i = s_i^t$
15. **end if**
16. **end if**
17. Set $t = t + 1$
18. **end while**
19. 根据公式 (28) 计算出最优决策 S 下的最小功耗 $E_{\min}$.
20. $\Theta \leftarrow BUAM\ (S)$
21. return S, $E_{\min}$, Θ.

算法 2. BUAM.

1. 初始化 A_{user}, $\Theta_K\ (k)$, $J_{\max}$
2. 从接受的所有卸载请求协议中获取卸载决策 S, 并计算 MU 和 MESs 的偏好.
3. **for** each n=1 to N **do**
4. 对卸载请求 S 中的每个 MU 根据公式 (16) 对 MU 偏好 $\pi_N\ (n, k)$ 进行升序排序, 得到 $L_N\ (n)$
5. **end for**
6. **for** each k=1 to K **do**
7. 对每个 MESs 根公式 (33) 对 MESs 偏好 $\pi_K\ (k, n)$ 进行降序排序, 得到得到 $L_K\ (k)$
8. **end for**
9. **while** $|\Theta_K\ (k)| \leqslant J_{\max}$ **do**
10. $n = \arg\max_{\Theta_K^C(k)} \pi_K(k, n)$
11. **if** MU 和 MESs 都满足对 QoS 的要求 **then**
12. 考虑过的 MESs 分配给 MU
13. 更新 $\Theta_K\ (k) \leftarrow \Theta_K\ (k) \cup n$
14. $\Theta_K^C(k) \leftarrow \{\Theta_K^C(k) \backslash n\}$
15. $A_{\mathrm{user}} \leftarrow \{A_{\mathrm{user}} \backslash n\}$
16. $S \leftarrow \{S \backslash n\}$
17. **else**
18. 考虑过的 MESs 不能分配给 MU
19. 清空 $\Theta_K\ (k) = \varnothing$.

20. **end if**
21. **end while**

4.2.2 时间复杂度分析

命题 1. 算法 1 中, EDCO 的计算复杂度为 O(T·N), 其中 T 为 EDCO 收敛迭代次数, N 为计算任务数.

证明: 算法 1 中的 EDCO 共有两层循环. 根据推论 1, 外部 for 循环表示博弈参与者, 当博弈达到 NE 态时, 其收敛时间是有限的. 我们用 T 表示 EDCO 中外部 for 循环的迭代次数. 此外, 步骤 1–6 的内部 for 循环最多运行 N 次, 从步骤 7–15 的过程时间复杂度为 O(1). 最后, 总运行时间 EDCO 的计算方法是将它们相乘, 即 O(T·N), 其中 T 为算法的收敛迭代次数, N 为计算任务数.

4.3 用户关联算法

在算法 1 解算出卸载决策后, 可对用户和 MEC 服务器的关联匹配做出最优分配, 该问题可形式化为:

$$\begin{cases} \text{(P2)} \quad \forall n \in N, \max \chi(a) \\ \text{s.t.} \begin{cases} 0 < a_n^k \leqslant q_{\max} \text{ (47a)} \\ \sum\limits_{m=1}^{M} a_n^m = 1 \text{ (47b)} \\ n \in N,\ k \in K \text{ (47c)} \end{cases} \end{cases} \tag{47}$$

其中, P2 是一个关于卸载决策 a 的匹配问题, 47a 限制了每个 MESs 关联的用户数量不能超过 $q_{\max}$. 47b 表示每个用户只能关联一个 MESs, 47c 表示用户范围约束. 我们设计的用户关联匹配算法受 Gale-Shapley 理论[50]的启示, 对 P2 中的用户和 MEC 服务器两组参与者可以根据多对一双边匹配理论做出用户关联决策. 为此, 我们改进了该算法, 并提出了最佳用户关联匹配 BUAM 算法. 首先给出用户和 MESs 的偏好函数和偏好列表.

(1) 用户偏好: 为了满足服务质量的保障, 用户会更倾向于与较低延迟的 MESs 关联. 因此, 我们使用公式 (16) 中的卸载延迟作为用户的偏好函数 π_N, 并根据函数值的升序得到偏好列表 L_N, 则 MU n 对 MESs 的偏好函数为:

$$\pi_N(n,k) = T^{\text{off}} \text{ (a)} \tag{48}$$

定义 MU n 的偏好列表为 $L_N(n) = [1,\ldots,k,\ldots,K]$, 即 π_N 按升序排列 $\pi_N(n,1) \leqslant \cdots \leqslant \pi_N(n,k) \leqslant \cdots \leqslant \pi_N(n,K)$.

(2) MESs 偏好: MESs 运营商为了获得更好的效益, 更愿意接受能够带来高效益的用户请求. 基于此, 我们使用公式 (33) 效益函数作为 MESs 的偏好函数 π_K, 并根据函数值的递减顺序确定偏好列表 L_K. MESs 对 MU n 的偏好为:

$$\pi_k(k,n) = \chi_n(S) \tag{49}$$

令 MES m 的偏好列表为 $L_K(k) = [1,\ldots,n,\ldots,N]$, 即 π_k 按降序排列 $\pi_k(k,1) \geqslant \cdots \geqslant \pi_k(k,n) \geqslant \cdots \geqslant \pi_N(K,N)$. 令 Θ 为 MU 和 MESs 的一组关联匹配关系对, $\Theta = \{(n,k)$, 表示 MU n 与 MES k 关联, $n \in N, K \in K\}$, 给定匹配函数 Θ, $\Theta_N(n)$ 为 MU n 匹配的一组 MESs. 相反, $\Theta_K(k)$ 表示匹配 MESs k 的一组用户. A_{user} 记录了当前未匹配任何 MESs 的用户集合, $\Theta_K^C(k)$ 表示请求至 MESs k 的所有用户集合, $J_{\max}$ 表示 MESs 最多能关联的 MU 数量. 基于 EDCO 计算卸载算法求解的卸载决策后, 本文提出了一种基于 Gale-Shapley 匹配理论的最佳用户关联匹配算法 (BUAM) 来获得最优匹配. 算法的细节如算法 2 所示.

5 数值模拟

在本节中, 我们通过模拟实验验证提出计算卸载算法和关联匹配算法的能效. 下面首先给出模拟设置的详细信息, 然后讨论下列评价问题.

- EDCO 算法相比其他基准算法整体的能效和时延表现如何?
- 关联匹配算法 BUAM 算法相比其他关联匹配算法的能效表现如何?
- 权重系数对卸载时延和能耗有多敏感?
- 设备和基站的距离对能耗的影响? 即算法对用户移动性的支持.
- EDCO 算法的收敛速度如何?

5.1 模拟设置

本实验通过 Matlab 进行数值模拟实验进行验证, 表 1 是试验中用到的主要参数. 根据 4G 蜂窝网络特性[45], 设定每个移动用户的传输功率为 [100, 200] mW. 基于移动边缘计算辅助视频和游戏等应用的配置[51], 设置输入数据包大小和任务所需的平均 CPU 周期数分别是 [300, 800] KB、[100, 1000] Megacycles. 对于不同的移动用户, 其计算任务可能是异构的, 因此本实验设定卸载任务的最大容忍时延范围为 t_{max} =[1, 2] s 之间. MU 与边缘服务器间的距离 d, 均匀分布在 (200, 500) 米范围内.

为了进一步验证提出的 EDCOS (energy efficient distributed computing offloading scheme) 的性能, 我们将其与以下几种基准算法进行比较, 分别为始终不卸载 NOS (no offloading scheme)、随机组件委派模式 RCAS (random component assignment scheme)、基于遗传算法 GACS (genetic algorithm based computation scheme). 在 NOS 中, 所有用户都选择计算任务在本地执行, 不再卸载任务到边缘服务器. 特别地, 本节中的能耗是指完成任务所消耗的全部能耗. 在 RCAS 中, 用户随机做出卸载决策和传输功率, SBSs 随机的分配信道和计算资源给用户. GACAS 是基于遗传算法进行计算卸载. 最后, 为对比关联匹配算法的效果, 在关联匹配算法的对比基准算法上, 我们与基于最近关联原则的联合卸载决策算法 (JODRA-NAP) 和资源均等分配的联合用户关联算法 (JUAOD-ERA)[52,53]对比.

表 1 模拟参数列表

模拟参数	值	模拟参数	值
K	5	T_{MAX}	[0, 2] s
N	10	f_{MU}	1.32 GHz
μ_1	2×10^{-6}	f_{MEC}	20 GHz
μ_2	500	f_{D2D}	1.5 GHz
P_{MEC}	800 mW	g_0	−40 dBm/Hz
P_{D2D}	190 mW	d_0	200 m
P_{ONs}	150 mW	θ_0	330
P_{res}	200 mW	N_0	−75 dBm
W_0	–60 dBm	$e^{local}_{L_i^j}$	10^{-3} J
W_{MEC}	40×10^6 MHz	$e^{MEC}_{L_i^j}$	2.8788×10^{-8} J
W_{MU}	20×10^6 MHz	$e^{D2D}_{L_i^j}$	8×10^{-7} J
W_{ONs}	17×10^6 MHz	$e^{MU}_{L_i^j}$	1.6542×10^{-9} J
P_{MAX}	1 000 mW	$e^{Cloud}_{L_i^j}$	1.4788×10^{-9} J
T^{off}	1 ms		

5.2 性能对比

在图 2、图 3 中, 除了 NOS 方式, 对于其他 3 种卸载模式, 每个用户的平均时延成本和能耗都随着 MU 数量的增加而增大. 其原因是: (1) 移动用户数量越大, 卸载任务数增多, 排队系统越拥挤, 等待延迟和能耗越高. (2) 信道干扰随着设备的增加而增加. 由于始终在本地执行计算, NOS 保持不变. 从图中结果表明所提出的 EDCOS 保持了最小化能耗和总时延代价. EDCOS 的调度规则很好地平衡了所有移动用户在计算卸载中的等待成本和能耗成本, 取得了明显优于其他基准算法的效果.

图 4(a) 给出了不同卸载策略下, 如本地计算 ($a_i = 0$)、MEC 卸载 ($a_i = 1, 2$) 和 D2D 卸载 ($a_i = 3, 4$), 每种卸载策略的效益值与 MUs 数量之间的定量关系. 可以看出, 随着 MU 数量的增加, 选择 D2D 卸载和 MEC 卸载的效益值逐渐低于 EDCOS 的效用值. 由于随着 MU 数的增加, 传输能量消耗和传输延迟增加, 导致开销增大, 选择本地

计算与其他卸载方式相结合的策略成为更好的选择. 在图 4(b) 中展示了不同卸载模式下的能耗和基站数量之间的定量关系. 结果表明, EDCOS 和 GACS 的能耗随着 SBSs 数量的增加而增大. 虽然系统在这种情况下保留了更多的资源, 但干扰也随之增加. 因此, EDCOS 和 GACS 的能耗都出现上升趋势. 对于 NOS, 始终选择本地执行任务, 不会因为基站数量改变而改变. 对于 RCAS, 由于资源是随机分配的, 所以无法确保性能. 图 4(c) 展示了随着 SBSs 数量的增加, 运营商的效益值变化. 显然, 我们设计的 BUAM 算法获得了用户关联算法中最大的效用, 并且随着 SBSs 数目的增加, 其相对于基准算法的增益也越来越明显. 因为 EDCOS 共同优化了用户关联, 而基准算法只考虑其中的一种或两种.

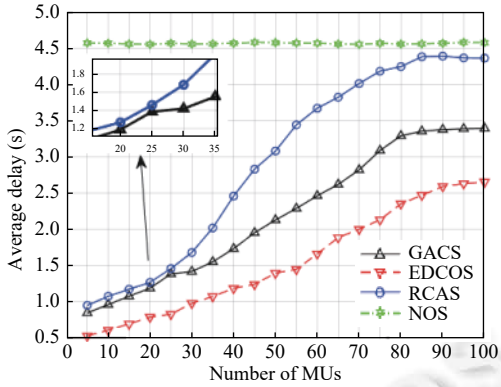


图 2 不同 MU 数量下各种卸载模式的时延对比

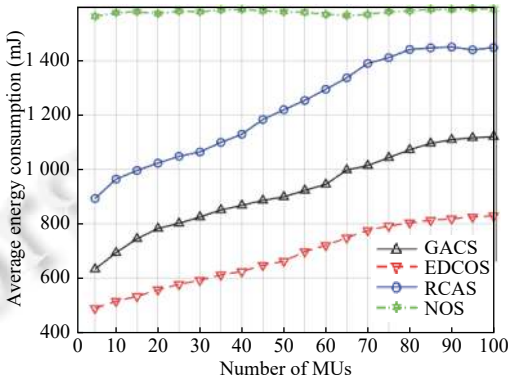
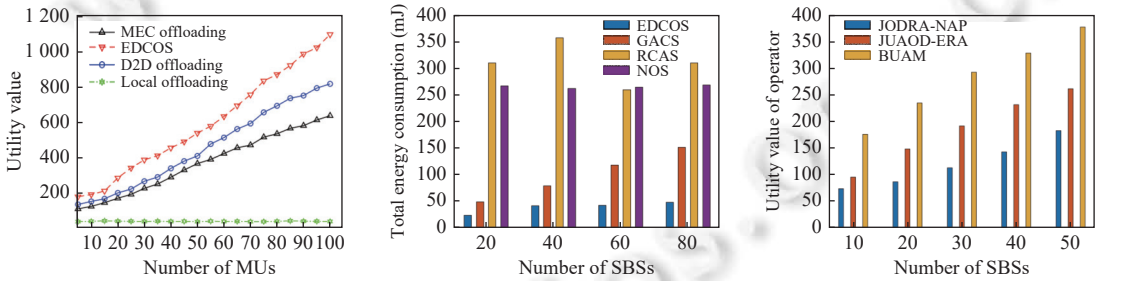


图 3 不同 MU 数量下各种卸载模式的能耗对比

从图 4(a) 和图 4(b) 可以看出, 我们提出的 EDCOS 算法性能优于其他 3 种基准算法. 在 RCAS 中, 卸载决策是随机进行的, 这可能导致不同 SBSs 之间的信道干扰增大, 从而导致能耗变得很大. 且由公式 (1) 可知传输速率也会随着干扰而下降, 这导致了卸载时延的增加, 所以在两图中 RCAS 保持了最高能耗和时延. EDCOS 通过优化信道分配和功率控制, 从而控制卸载 MU 的数量与信道干扰的数据量, 因此节省了整体的卸载能耗和时延. GACS 性能优于 RCAS 和 NOS, 遗传操作 (交叉和变异) 可以保持解的多样性, 扩展搜索区域, 从而跳出局部最优. EDCOS 利用博弈思想, 保持了每个 MU 的结果最优, 性能优于其他 3 种基准算法.



(a) 不同 MU 数量下各种卸载方案的能效值

(b) 不同 SBSs 数量下各种卸载模式的能耗

(c) 不同 SBSs 数目下各种关联匹配算法 (JODRA-NAP, JUAOD-ERA, BUAM) 的运营商效益值

图 4 不同卸载模式的能耗及不同关联匹配算法的效益值比较

5.3 权重系数对性能的影响

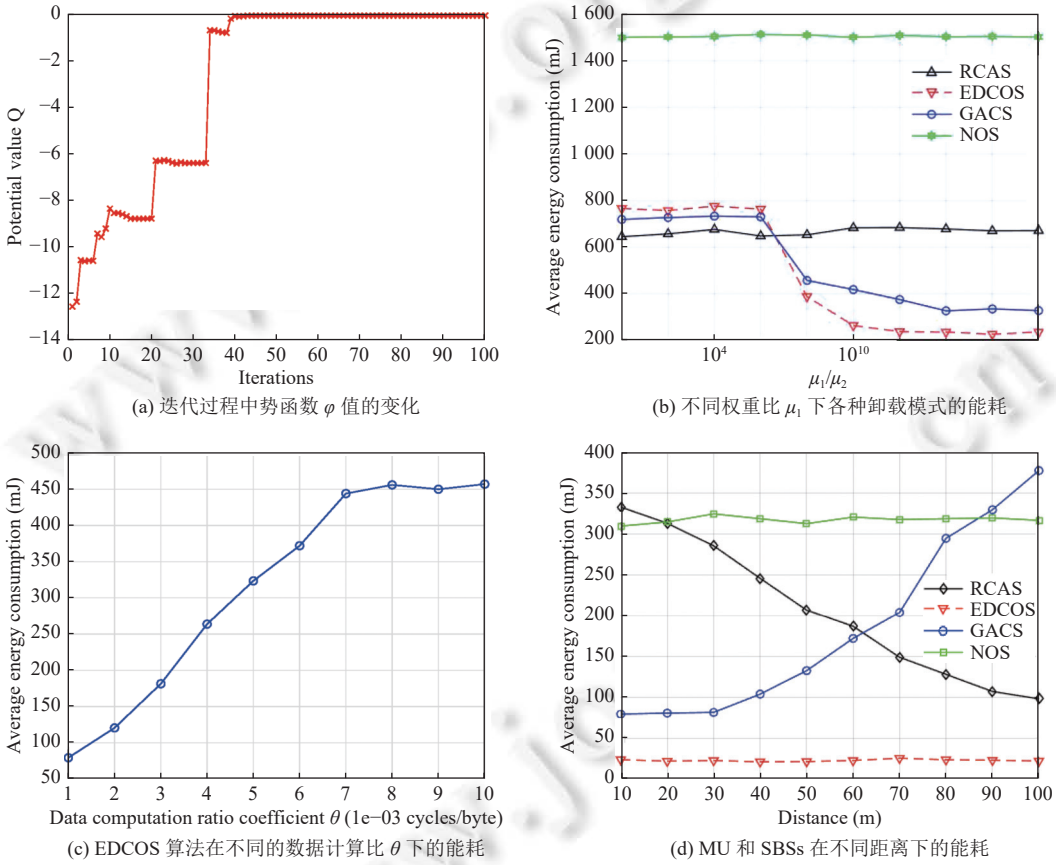
图 5(a) 展示了随着 MU 数量增加, EDCOS 算法在迭代过程中势函数 φ 值的变化. 数值模拟显示算法在 Iteration=45 后, 曲线保持水平静止, 表明博弈达到纳什均衡状态, 进一步验证了收敛点是 NE. 根据势博弈的性质, 势函数 φ 值在其最大值为 NE 时单调递增并收敛. 因此, 博弈经过少量迭代后实现了纳什均衡.

在图 5(b) 描述了不同的 μ_1/μ_2 的比值下, 不同卸载模式的能耗变化. 随着 μ_1 的增大, 由于 NOS 始终在本地计算, 其能耗比其他卸载算法要高得多. 当 $\mu_1/\mu_2 = 10^6$ 之后, RCAS, GACS, EDCOS 的能耗都降到 800 mJ 以下.

GACS 算法在当 μ_2 足够高时, MU 的传输功率反而会降低, 导致能耗增大. 关于随机分配和本地处理, 由于 MU 的卸载策略不依赖于参数 μ_1,μ_2, 所以它们的能耗是常量. 通过对比可以发现, 基于博弈理论的 EDCOS 算法保持了更小的能耗成本.

图 5(c) 中展示了不同数据计算比 θ 下 EDCOS 的能耗. 实际上, 当 θ 在低值时 (如运行一些低 CPU 周期数的任务时), MU 的最优决策是使用本地进程的 CPU 周期. 当 θ 增大时, MU 的卸载任务数量增多, 需要处理的 CPU 周期数也会增加. 这意味着更多的任务会通过无线信道与计算接入点连接, 导致严重的信道干扰, 近而导致能耗增加. 图 5(d) 描述了 MU 和 SBSs 之间距离对能耗的影响. 对比发现: EDCOS 的性能总是优于粗粒度的卸载模式 (RCAS 和 GACS). 其中本地计算 NOS 模式由于始终在本地进行计算任务, 保持了较高恒定的能耗. 此外, 我们观察到距离 d 对 EDCOS 的影响很小, 这意味着 EDCOS 支持 MU 的移动性, 在各种网络条件下能够保持稳定.

本文所考虑的 3 层卸载框架可对应于一个静态的物联网网络, 该网络的发射机和接收机固定在某个位置. 文献>[54,55] 的测量实验表明, 信道不变的信道相干时间在 1~10 s 之间, 通常不小于 2 s. 以 N=30 的 MEC 网络为例, 我们设计的 EDCO 算法的执行完毕时间平均为 0.061 s, 这对于现场部署来说是可以接受的开销. 因此, EDCO 算法使得信道衰落环境下无线 MEC 网络的计算卸载成为切实可行的.



(a) 迭代过程中势函数 φ 值的变化　(b) 不同权重比 μ_1 下各种卸载模式的能耗

(c) EDCOS 算法在不同的数据计算比 θ 下的能耗　(d) MU 和 SBSs 在不同距离下的能耗

图 5　不同参数下 EDCOS 算法的性能对比

6 结　论

本文研究了基于多用户多 MEC 场景中的计算卸载决策和用户关联算法. 首先我们提出一个云-边-端的联合计算卸载框架, 将云计算中心和边缘节点优势互补, 并通过优化卸载决策, 同时考虑了干扰管理和功率控制. 接着

对多用户计算卸载模型进行形式化描述为一种能效感知的效益模型, 将能耗和时延最优化问题转化为混合整数非线性规划问题, 这是 NP-hard 问题. 其次, 多用户计算卸载中每个用户是理性的, 即每个用户会追求自身效益最大化而调整卸载策略. 为了确保以网络范围内的最优方案进行卸载决策, 建立了博弈论模型来共同确定计算卸载方案, 并证明了该场景下的多用户计算卸载为严格势力场博弈. 所设计的计算卸载算法可以使每个用户的效用最大化. 此外, 为了进一步优化服务质量, 我们设计了用户关联匹配算法, 寻找边缘服务器和请求卸载的用户之间的最佳匹配关系. 最后, 相比起基准算法, 我们提出卸载算法在收敛速度和能效上明显优于传统基准算法.

我们的方法也存在不足, 在未来的工作中, 我们将对 D2D 和机会网络节点之间链路丧失导致数据丢失或计算失败情况下卸载策略的代价考虑进来. 同时, 我们也对不同性能需求情况下制定合理的定价规则和资源分配感兴趣. 从运营商的角度出发做效益优化, 保证在多用户 QoS 约束下提高运营商的潜在效益.

References:

[1] Zhou Z, Chen X, Li E, Zeng LK, Luo K, Zhang JS. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proc. of the IEEE, 2019, 107(8): 1738–1762. [doi: 10.1109/JPROC.2019.2918951]

[2] Gerards MET, Hurink JL, Kuper J. On the interplay between global DVFS and scheduling tasks with precedence constraints. IEEE Trans. on Computers, 2015, 64(6): 1742–1754. [doi: 10.1109/TC.2014.2345410]

[3] Giang NK, Lea R, Blackstock M, Leung VCM. Fog at the edge: Experiences building an edge computing platform. In: Proc. of the 2018 IEEE Int'l Conf. on Edge Computing (EDGE). San Francisco: IEEE, 2018. 9–16. [doi: 10.1109/EDGE.2018.00009]

[4] Li E, Zeng LK, Zhou Z, Chen X. Edge AI: On-demand accelerating deep neural network inference via edge computing. IEEE Trans. on Wireless Communications, 2020, 19(1): 447–457. [doi: 10.1109/TWC.2019.2946140]

[5] Li E, Zhou Z, Chen X. Edge intelligence: On-demand deep learning model co-inference with device-edge synergy. In: Proc. of the 2018 Workshop on Mobile Edge Communications. Budapest: ACM, 2018. 31–36. [doi: 10.1145/3229556.3229562]

[6] Shi WS, Cao J, Zhang Q, Li YHZ, Xu LY. Edge computing: Vision and challenges. IEEE Internet of Things Journal, 2016, 3(5): 637–646. [doi: 10.1109/JIOT.2016.2579198]

[7] Chouhan S. Energy optimal partial computation offloading framework for mobile devices in multi-access edge computing. In: Proc. of the 2019 Int'l Conf. on Software, Telecommunications and Computer Networks (SoftCOM). Split: IEEE, 2019. 1–6. [doi: 10.23919/SOFTCOM.2019.8903763]

[8] Xu FM, Yang F, Zhao CL, Fang C. Edge computing and caching based blockchain IoT network. In: Proc. of the 1st IEEE Int'l Conf. on Hot Information-Centric Networking (HotICN). Shenzhen: IEEE, 2018. 238–239. [doi: 10.1109/HOTICN.2018.8606001]

[9] Mao YY, You CS, Zhang J, Huang KB, Letaief KB. A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials, 2017, 19(4): 2322–2358. [doi: 10.1109/COMST.2017.2745201]

[10] Xu YZ, Cheng P, Chen Z, Ding M, Li YH, Vucetic B. Real-time task offloading for large-scale mobile edge computing. In: Proc. of the 2020 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP). Barcelona: IEEE, 2020. 4975–4979. [doi: 10.1109/ICASSP40776.2020.9054413]

[11] Lin L, Liao XF, Jin H, Li P. Computation offloading toward edge computing. Proc. of the IEEE, 2019, 107(8): 1584–1607. [doi: 10.1109/JPROC.2019.2922285]

[12] Ren JK, Yu GD, Cai YL, He YH. Latency optimization for resource allocation in mobile-edge computation offloading. IEEE Trans. on Wireless Communications, 2018, 17(8): 5506–5519. [doi: 10.1109/TWC.2018.2845360]

[13] Tran TX, Hajisami A, Pandey P, Pompili D. Collaborative mobile edge computing in 5G networks: New paradigms, scenarios, and challenges. IEEE Communications Magazine, 2017, 55(4): 54–61. [doi: 10.1109/MCOM.2017.1600863]

[14] Li HX, Shou GC, Hu YH, Guo ZG. Mobile edge computing: Progress and challenges. In: Proc. of the 4th IEEE Int'l Conf. on Mobile Cloud Computing, Services, and Engineering (MobileCloud). Oxford: IEEE, 2016. 83–84. [doi: 10.1109/MobileCloud.2016.16]

[15] Abbas N, Zhang Y, Taherkordi A, Skeie T. Mobile edge computing: A survey. IEEE Internet of Things Journal, 2018, 5(1): 450–465. [doi: 10.1109/JIOT.2017.2750180]

[16] Zhang DY, Tan L, Ren J, Awad MK, Zhang S, Zhang YX, Wan PJ. Near-optimal and truthful online auction for computation offloading in green edge-computing systems. IEEE Trans. on Mobile Computing, 2020, 19(4): 880–893. [doi: 10.1109/TMC.2019.2901474]

[17] Liu LN, Sun B, Wu Y, Tsang HKD. Latency minimization for computation offloading with hybrid multiple access methods. In: Proc. of the 2020 Int'l Conf. on Computing, Networking and Communications (ICNC). Big Island: IEEE, 2020. 820–825. [doi: 10.1109/ICNC47757.2020.9049805]

[18] Chang Z, Liu LQ, Guo XJ, Chen T, Ristaniemi T. Dynamic resource allocation and computation offloading for edge computing system. In: Proc. of the IFIP Int'l Conf. on Artificial Intelligence Applications and Innovations. Neos Marmaras: Springer, 2020. 61–73. [doi: 10.1007/978-3-030-49190-1_6]

[19] Ma LL, Yi SH, Carter N, Li Q. Efficient live migration of edge services leveraging container layered storage. IEEE Trans. on Mobile Computing, 2019, 18(9): 2020–2033. [doi: 10.1109/TMC.2018.2871842]

[20] Hmimz Y, El Ghmary M, Chanyour T, Malki MQC. Computation offloading to a mobile edge computing server with delay and energy constraints. In: Proc. of the 2019 Int'l Conf. on Wireless Technologies, Embedded and Intelligent Systems (WITS). Fez: IEEE, 2019. 1–6. [doi: 10.1109/WITS.2019.8723733]

[21] Al-Zihad M, Akash SA, Adhikary T, Razzaque A. Bandwidth allocation and computation offloading for service specific IoT edge devices. In: Proc. of the 2017 IEEE Region 10 Humanitarian Technology Conf. (R10-HTC). Dhaka: IEEE, 2017. 516–519. [doi: 10.1109/R10-HTC.2017.8289012]

[22] Zhang DG, Gong CL, Jiang KW, Zhang XD, Zhang T. A kind of new method of intelligent trust engineering metrics (ITEM) for application of mobile ad hoc network. Engineering Computations, 2020, 37(5): 1617–1643. [doi: 10.1108/EC-12-2018-0579]

[23] Zhang DG, Wu H, Zhao PZ, Liu XH, Cui YY, Chen L, Zhang T. New approach of multi-path reliable transmission for marginal wireless sensor network. Wireless Networks, 2020, 26(2): 1503–1517. [doi: 10.1007/s11276-019-02216-y]

[24] You CS, Zeng Y, Zhang R, Huang KB. Asynchronous mobile-edge computation offloading: Energy-efficient resource management. IEEE Trans. on Wireless Communications, 2018, 17(11): 7590–7605. [doi: 10.1109/TWC.2018.2868710]

[25] Guo HZ, Liu JJ. Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks. IEEE Trans. on Vehicular Technology, 2018, 67(5): 4514–4526. [doi: 10.1109/TVT.2018.2790421]

[26] Wang DF, Zhao J. Computation offloading and resource allocation in mobile edge computing via reinforcement learning. In: Proc. of the 11th Int'l Conf. on Wireless Communications and Signal Processing (WCSP). Xi'an: IEEE, 2019. 1–6. [doi: 10.1109/WCSP.2019.8927985]

[27] Zhang N, Guo ST, Dong YF, Jiang QC, Jiao J. Joint task offloading and data caching in mobile edge computing. In: Proc. of the 15th Int'l Conf. on Mobile Ad-Hoc and Sensor Networks (MSN). Shenzhen: IEEE, 2019. 234–239. [doi: 10.1109/MSN48538.2019.00053]

[28] Li YQ, Wang X, Gan XY, Jin HM, Fu LY, Wang XB. Learning-aided computation offloading for trusted collaborative mobile edge computing. IEEE Trans. on Mobile Computing, 2020, 19(12): 2833–2849. [doi: 10.1109/TMC.2019.2934103]

[29] Jeong HJ, Jeong IC, Lee HJ, Moon SM. Computation offloading for machine learning web apps in the edge server environment. In: Proc. of the IEEE 38th Int'l Conf. on Distributed Computing Systems (ICDCS). Vienna: IEEE, 2018. 1492–1499. [doi: 10.1109/ICDCS.2018.00154]

[30] Chen JQ, Mao GQ, Li CL, Liang WF, Zhang DG. Capacity of cooperative vehicular networks with infrastructure support: Multiuser case. IEEE Trans. on Vehicular Technology, 2018, 67(2): 1546–1560. [doi: 10.1109/TVT.2017.2753772]

[31] Chen JQ, Mao GQ, Li CL, Zhang DG. A topological approach to secure message dissemination in vehicular networks. IEEE Trans. on Intelligent Transportation Systems, 2020, 21(1): 135–148. [doi: 10.1109/TITS.2018.2889746]

[32] Zhao YL, Li Y, Cao Y, Jiang T, Ge N. Social-aware resource allocation for device-to-device communications underlaying cellular networks. IEEE Trans. on Wireless Communications, 2015, 14(12): 6621–6634. [doi: 10.1109/TWC.2015.2457427]

[33] Zhao GG, Chen S, Qi L, Zhao LQ, Hanzo L. Mobile-traffic-aware offloading for energy- and spectral-efficient large-scale D2D-enabled cellular networks. IEEE Trans. on Wireless Communications, 2019, 18(6): 3251–3264. [doi: 10.1109/TWC.2019.2912596]

[34] Ye QY, Al-Shalash M, Caramanis C, Andrews JG. Resource optimization in device-to-device cellular systems using time-frequency hopping. IEEE Trans. on Wireless Communications, 2014, 13(10): 5467–5480. [doi: 10.1109/TWC.2014.2340879]

[35] Lin XQ, Andrews JG, Ghosh A. Spectrum sharing for device-to-device communication in cellular networks. IEEE Trans. on Wireless Communications, 2014, 13(12): 6727–6740. [doi: 10.1109/TWC.2014.2360202]

[36] Yi CY, Cai J, Su Z. A multi-user mobile computation offloading and transmission scheduling mechanism for delay-sensitive applications. IEEE Trans. on Mobile Computing, 2020, 19(1): 29–43. [doi: 10.1109/TMC.2019.2891736]

[37] Dinh TQ, La QD, Quek TQS, Shin H. Learning for computation offloading in mobile edge computing. IEEE Trans. on Communications, 2018, 66(12): 6353–6367. [doi: 10.1109/TCOMM.2018.2866572]

[38] Zhang DG, Chen C, Cui YY, Zhang T. New method of energy efficient subcarrier allocation based on evolutionary game theory. Mobile Networks and Applications, 2021, 26(2): 523–536. [doi: 10.1007/s11036-018-1123-y]

[39] Hu GS, Jia YJ, Chen ZC. Multi-user computation offloading with D2D for mobile edge computing. In: Proc. of the 2018 IEEE Global Communications Conf. (GLOBECOM). Abu Dhabi: IEEE, 2018. 1–6. [doi: 10.1109/GLOCOM.2018.8647906]

[40] Xu J, Chen LX, Zhou P. Joint service caching and task offloading for mobile edge computing in dense networks. In: Proc. of the 2018

IEEE Conf. on Computer Communications. Honolulu: IEEE, 2018. 207–215. [doi: 10.1109/INFOCOM.2018.8485977]

[41] Wei ZL, Zhao BK, Su JS, Lu XC. Dynamic edge computation offloading for Internet of Things with energy harvesting: A learning method. IEEE Internet of Things Journal, 2018, 6(3): 4436–4447. [doi: 10.1109/JIOT.2018.2882783]

[42] Zhang DG, Liu XH, Cui YY, Chen L, Zhang T. A kind of novel RSAR protocol for mobile vehicular ad hoc network. CCF Trans. on Networking, 2019, 2(2): 111–125. [doi: 10.1007/s42045-019-00019-5]

[43] Dai YY, Xu D, Zhang K, Lu YL, Maharjan S, Zhang Y. Deep reinforcement learning for edge computing and resource allocation in 5G beyond. In: Proc. of the 19th IEEE Int'l Conf. on Communication Technology (ICCT). Xi'an: IEEE, 2019. 866–870. [doi: 10.1109/ICCT46805.2019.8947146]

[44] Hu M, Zhuang L, Wu D, Zhou YP, Chen X, Xiao L. Learning driven computation offloading for asymmetrically informed edge computing. IEEE Trans. on Parallel and Distributed Systems, 2019, 30(8): 1802–1815. [doi: 10.1109/TPDS.2019.2893925]

[45] Li JW, Zhang HL, Ji H, Li X. Joint computation offloading and service caching for MEC in multi-access networks. In: Proc. of the 30th IEEE Annual Int'l Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC). Istanbul: IEEE, 2019. 1–6. [doi: 10.1109/PIMRC.2019.8904354]

[46] Zhao HL, Du W, Liu W, Lei T, Lei QW. Qoe aware and cell capacity enhanced computation offloading for multi-server mobile edge computing systems with energy harvesting devices. In: Proc. of the 2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). Guangzhou: IEEE, 2018. 671–678. [doi: 10.1109/SmartWorld.2018.00133]

[47] Liu S, Zhang DG, Liu XH, Zhang T, Wu H. Adaptive repair algorithm for TORA routing protocol based on flood control strategy. Computer Communications, 2020, 151: 437–448. [doi: 10.1016/j.comcom.2020.01.024]

[48] Cui YY, Zhang DG, Zhang T, Chen L, Piao M, Zhu HL. Novel method of mobile edge computation offloading based on evolutionary game strategy for IoT devices. AEU-Int'l Journal of Electronics and Communications, 2020, 118: 153134. [doi: 10.1016/j.aeue.2020.153134]

[49] Saraydar CU, Mandayam NB, Goodman DJ. Efficient power control via pricing in wireless data networks. IEEE Trans. on Communications, 2002, 50(2): 291–303. [doi: 10.1109/26.983324]

[50] Zhang HW, Jing WP, Lu ZM, Wen XM, Zhang JY. Joint user association and value-aware computation offloading for MEC-enabled networks. In: Proc. of the 2020 IEEE Wireless Communications and Networking Conf. Workshops (WCNCW). Seoul: IEEE, 2020. 1–6. [doi: 10.1109/WCNCW48565.2020.9124793]

[51] Zhang DG, Chen L, Zhang J, Chen J, Zhang T, Tang YM, Qiu JN. A multi-path routing protocol based on link lifetime and energy consumption prediction for mobile edge computing. IEEE Access, 2020, 8: 69058–69071. [doi: 10.1109/ACCESS.2020.2986078]

[52] Zhang DG, Piao M, Zhang T, Chen C, Zhu HL. New algorithm of multi-strategy channel allocation for edge computing. AEU-Int'l Journal of Electronics and Communications, 2020, 126: 153372. [doi: 10.1016/j.aeue.2020.153372]

[53] Nguyen PD, Le LB. Joint computation offloading, SFC placement, and resource allocation for multi-site MEC systems. In: Proc. of the 2020 IEEE Wireless Communications and Networking Conf. (WCNC). Seoul: IEEE, 2020. 1–6. [doi: 10.1109/WCNC45663.2020.9120597]

[54] Howard SJ, Pahlavan K. Doppler spread measurements of indoor radio channel. Electronics Letters, 1990, 26(2): 107–109. [doi: 10.1049/el:19900074]

[55] Herbert S, Wassell I, Loh TH, Rigelsford J. Characterizing the spectral properties and time variation of the in-vehicle wireless communication channel. IEEE Trans. on Communications, 2014, 62(7): 2390–2399. [doi: 10.1109/TCOMM.2014.2328635]

张祥俊(1993－), 男, 博士生, CCF 学生会员, 主要研究领域为云计算, 大数据, 边缘计算, 网络, 信息安全.

伍卫国(1963－), 男, 博士, 教授, CCF 高级会员, 主要研究领域为云计算, 大数据, 高性能, 存储系统, 嵌入式.

张弛(1990－), 男, 博士生, 主要研究领域为云计算, 大数据.

柴玉香(1996－), 女, 硕士生, 主要研究领域为云计算, 大数据.

杨诗园(1996－), 男, 硕士生, 主要研究领域为云计算, 大数据.

王雄(1996－), 男, 硕士生, 主要研究领域为云计算, 大数据.