中国科学: 信息科学 2022 年 第 52 卷 第 7 期: 1349–1361
SCIENTIA SINICA Informationis
论文

不确定网络环境下的任务卸载和资源分配算法

姚枝秀, 夏士超, 李云*

重庆邮电大学通信与信息工程学院, 重庆 400065
* 通信作者. E-mail: liyun@cqupt.edu.cn

收稿日期: 2021–05–30; 修回日期: 2021–07–25; 接受日期: 2021–08–27; 网络出版日期: 2022–07–15

国家自然科学基金 (批准号: 62071077) 和重庆邮电大学博士研究生高端人才培养 (批准号: BYJS201806) 资助项目

摘要 随着计算密集型和时延敏感型应用的大量涌现, 在大数据和低时延计算需求的驱动下, 移动边缘计算 (mobile edge computing, MEC) 在提升用户体验和降低能耗方面将发挥重要作用. 然而, 由于计算资源有限的 MEC 服务器无法快速响应海量突发的计算请求, 计算任务在 MEC 服务器的排队等待时间是不可忽略且难以预测的. 为了保证在排队等待时间不确定的网络环境中满足应用的计算时延需求, 本文在计算时延约束条件下, 以最小化系统能耗为目标, 提出了一种基于随机模拟的任务卸载和资源分配两阶段随机规划算法 SS_2SSP. 仿真结果表明, SS_2SSP 算法能够保证应用的计算时延需求, 同时有效降低了系统能耗.

关键词 移动边缘计算, 不确定网络, 任务卸载, 资源分配, 计算时延, 两阶段随机规划

1 引言

随着 5G 进入大规模商用阶段, 6G 的研究计划已经启动. 在未来的 6G 网络中, 将会涌现更多实时应用场景, 如高保真沉浸式 AR/VR (virtual and augmented reality) 等新应用将具有更高数据速率、更低交互时延的需求 [1]. 此外, 虽然终端设备的计算能力近年来不断提高, 但物理体积和电池容量的限制使其处理计算密集型和时延敏感型应用仍然面临着严峻挑战. 移动边缘计算 (mobile edge computing, MEC) 的提出弥补了终端设备计算能力的不足, 同时提供了高速率和低时延的计算服务环境 [2]. 但随着边缘服务器广泛部署的推进, 可以预见未来 6G 网络将拥有海量无处不在的边缘节点, 这将对能耗带来巨大挑战. 因此, 在保证终端用户低时延要求的同时满足能耗最小化显得尤为重要.

移动边缘计算中任务卸载和资源分配策略是影响用户卸载时延和能耗的关键因素, 为提高用户的体验质量 (quality of experience, QoE), 同时最小化计算卸载能耗, 需要根据任务的计算能耗预算、计算时延, 以及 MEC 服务器的可用资源等性能约束条件制定任务卸载和资源分配策略. 国内外一些研

引用格式: 姚枝秀, 夏士超, 李云. 不确定网络环境下的任务卸载和资源分配算法. 中国科学: 信息科学, 2022, 52: 1349–1361, doi: 10.1360/SSI-2021-0186
Yao Z X, Xia S C, Li Y. Task offloading and resource allocation in an uncertain network (in Chinese). Sci Sin Inform, 2022, 52: 1349–1361, doi: 10.1360/SSI-2021-0186

究人员针对计算时延约束条件下能耗最小化问题进行了深入研究[3~7], 但都忽略了任务在 MEC 服务器端的排队等待时延. 在实际的 MEC 网络环境中, 由于 MEC 服务器的计算、存储等资源非常有限, 通常无法快速响应海量突发的计算请求, 因此, 任务在 MEC 服务器端的排队等待时间是不可忽略的. 针对此问题, Meng 等[8] 考虑了本地队列状态信息、MEC 服务器队列状态信息和信道状态信息, 为了解决本地和服务器队列的强耦合问题, 作者将任务卸载过程建模为马尔科夫 (Markov) 过程, 通过优化传输功率和本地 CPU (central processing unit) 频率, 提出了一种时延最优的任务卸载策略. Han 等[9] 根据用户和 MEC 服务器的队列状态信息确定传输功率和 MEC 服务器计算资源的最优调度策略, 研究了任务卸载时延约束条件下的用户功耗最小化问题. Merluzzi 等[10] 将本地和 MEC 服务器的任务队列的总和度量为计算时延, 提出一种传输功率和 MEC 服务器的 CPU 频率的联合优化策略, 以保证任务队列总和稳定性的同时使得用户能耗最小化.

然而, 上述研究工作都只考虑了任务的平均计算时延, 对于一些时延敏感型应用, 如 AR/VR, 对计算时延的要求不仅包括平均时延, 更要考虑网络中随机时延的影响[11]. 在实际的 MEC 网络环境中, 由于海量计算请求的随机性和突发性, MEC 服务器的排队等待时间是随机且难以预测的. 这种不确定性因素对传统任务卸载和资源分配造成了严峻挑战. 此外, 大部分研究工作都只针对用户端能耗, 而计算资源有限的边缘节点依然面临着大量用户的访问, 随之而来的是计算资源匮乏, 以及高能耗等问题. 因此, 如何在任务计算时延不确定的 MEC 网络环境中制定高效、绿色的任务卸载和资源分配策略具有重要研究意义.

为解决上述问题, 本文考虑排队等待时间不确定的 MEC 网络环境, 提出一种基于随机模拟的任务卸载和资源分配两阶段随机规划算法 SS_2SSP (stochastic simulation based two-stage stochastic programming algorithm for task offloading and resources allocation). 本文主要贡献如下:

(1) 为了描述任务在 MEC 服务器排队等待时间的不确定性, 将排队等待时间建模为一组随机变量, 并在时延约束条件下, 以最小化系统总能耗为目标, 将优化问题建模为基于两阶段随机规划的任务卸载和资源分配问题;

(2) 为了降低两阶段随机规划问题的计算复杂度, 利用随机模拟方法将原问题转化为样本均值近似问题. 由于该问题为混合整数非线性规划 (mixed integer nonlinear programming, MINLP) 问题, 将该问题解耦为本地计算资源分配、传输功率和边缘计算资源分配, 以及卸载决策 3 个子问题;

(3) 为了获得任务卸载和资源分配的最优策略, 首先采用拉格朗日 (Lagrange) 乘子法获得本地计算资源最优分配策略, 同时采用遗传算法获得传输功率和边缘计算资源的最优分配策略, 最后通过分析本地计算和边缘计算的时延估计和能耗预算获得最优任务卸载决策.

2 系统模型

考虑一个 MEC 系统模型, 该模型包含一个配置了服务器的基站, N 个请求任务处理的用户, 定义 UE i 表示第 i 个用户, 用户通过无线网络与 MEC 服务器进行通信. 本文采用任务不可分割的二元卸载模型[12], 即任务必须作为一个整体在用户端或 MEC 服务器上执行. 定义 UE i 需要处理的任务为一个三元组 $A_i = \{D_i, L_i, \tau_i\}$, 其中, D_i 表示需要计算的任务量大小; L_i 表示单位 bit 任务所需要的 CPU 周期数, 单位为 cycles/bit; τ_i 表示该任务的计算时延要求. 定义 $\pi_i \in \{0, 1\}$ 表示 UE i 的卸载决策, $\pi_i = 0$ 表示任务在本地处理, $\pi_i = 1$ 表示将任务卸载到 MEC 服务器处理.

2.1 本地计算模型

假设每个用户的处理器均支持动态电压频率调整 (dynamic voltage frequency scaling, DVFS) 技术, DVFS 技术可以动态调整 CPU 频率, 从而达到节省能耗的目的. 定义 f_i^{l} 表示本地计算时 UE i 的 CPU 频率, 则本地计算的时延为

$$T_i^{\mathrm{l}}\left(f_i^{\mathrm{l}}\right)=\frac{D_iL_i}{f_i^{\mathrm{l}}}, \tag{1}$$

本地计算时 UE i 产生的计算能耗为 [8]

$$E_i^{\mathrm{l}}\left(f_i^{\mathrm{l}}\right)=\kappa_i\left(f_i^{\mathrm{l}}\right)^2T_i^{\mathrm{l}}\left(f_i^{\mathrm{l}}\right), \tag{2}$$

其中 κ_i 表示与 UE i 芯片架构相关的有效能量系数.

2.2 边缘计算模型

任务卸载到 MEC 服务器执行主要经历 4 个过程: 任务上传、MEC 服务器排队等待、MEC 服务器计算, 以及计算结果返回. 由于返回结果时的任务量较小 [13], 为便于分析, 本文忽略计算结果返回的通信时延. UE i 将任务卸载到 MEC 服务器可获得的传输速率为

$$R_i\left(p_i\right)=B_0\log_2\left(1+\frac{h_ip_i}{N_0B_0}\right), \tag{3}$$

其中, B_0 表示通信带宽, h_i 表示 UE i 和 MEC 服务器之间的信道增益, N_0 为噪声功率, p_i 为 UE i 卸载任务时的传输功率. 因此, 当任务卸载到 MEC 服务器处理时, 任务上传的通信时延为

$$T_i^{\mathrm{c}}\left(p_i\right)=\frac{D_i}{R_i\left(p_i\right)}, \tag{4}$$

任务上传的通信能耗为

$$E_i^{\mathrm{c}}\left(p_i\right)=p_iT_i^{\mathrm{c}}\left(p_i\right). \tag{5}$$

当任务上传至 MEC 服务器后, 计算资源有限的 MEC 服务器通常无法快速响应海量随机且突发的计算请求, 因此, MEC 服务器排队等待时间是不可忽略的. 定义 T_i^{w} 为 UE i 在 MEC 服务器的排队等待时间. 任务经过排队等待, MEC 服务器为其提供计算服务, f_i^{r} 表示 MEC 服务器分配给 UE i 的 CPU 频率, 则 UE i 在 MEC 服务器的计算时延为

$$T_i^{\mathrm{e}}\left(f_i^{\mathrm{r}}\right)=\frac{D_iL_i}{f_i^{\mathrm{r}}}, \tag{6}$$

MEC 服务器为 UE i 提供计算服务所产生的计算能耗为

$$E_i^{\mathrm{e}}\left(f_i^{\mathrm{r}}\right)=\gamma(f_i^{\mathrm{r}})^2T_i^{\mathrm{e}}\left(f_i^{\mathrm{r}}\right), \tag{7}$$

其中, γ 表示与 MEC 服务器芯片架构相关的有效能量系数.

3 基于两阶段随机规划的任务卸载及资源分配问题建模

本节首先对排队等待时间的不确定性进行建模; 然后在满足任务计算时延要求的条件下, 以最小化系统能耗为目标, 将优化问题建模为基于两阶段随机规划的任务卸载和资源分配问题; 最后为降低两阶段随机规划问题的计算复杂度, 利用随机模拟方法将原问题转化为基于样本均值近似的问题模型.

3.1 排队等待时间不确定性

由于海量计算请求的随机性和突发性, 任务在 MEC 服务器的排队等待时间是随机变化的, 并且难以获得其准确预测值. 为了描述任务在 MEC 服务器排队等待时间的不确定性, 本文将排队等待时间建模为一组离散型随机变量. 定义 $\Omega_i = \{T_{i,1}^{\mathrm{w}}, T_{i,2}^{\mathrm{w}}, \ldots, T_{i,|\Omega_i|}^{\mathrm{w}}\}$ 为 UE i 卸载任务到 MEC 服务器时所有可能的排队等待时间的集合, 称为场景, 其中 $|\Omega_i|$ 表示场景空间的大小, $T_{i,x}^{\mathrm{w}}, x = 1, 2, \ldots, |\Omega_i|$ 称为场景 Ω_i 中的一个实现. 进一步考虑所有的用户, 定义组合场景 Ω 表示所有用户在 MEC 服务器端排队等待时间的集合, 可表示为笛卡尔 (Cartesian) 乘积 $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_N$. 令 $\omega = \left(T_{1,x}^{\mathrm{w}}, T_{2,x}^{\mathrm{w}}, \ldots, T_{N,x}^{\mathrm{w}}\right) \in \Omega$ 表示组合场景中的一个组合实现, $T_i^{\mathrm{w}}(\omega)$ 表示组合实现为 ω 时 UE i 卸载任务到 MEC 服务器的排队等待时间.

3.2 两阶段卸载模型

MEC 服务器排队等待时间的不确定性因素给传统任务卸载和资源分配等问题带来了严峻挑战. 两阶段随机规划是指第 1 阶段问题的决策可以通过第 2 阶段问题的最优决策进行补偿的随机规划问题, 在资源配置等领域有着广泛的应用 [14]. 为此, 本文采用两阶段随机规划理论解决任务卸载过程中排队等待时间的不确定性问题, 旨在满足用户计算时延要求的条件下, 通过优化本地和 MEC 服务器的 CPU 频率资源、任务传输功率, 以及任务卸载决策最小化系统能耗.

图 1 所示为两阶段卸载模型图, 用户首先估计任务在本地计算时的能耗值; 然后在没有观察到 MEC 服务器不确定性排队等待时间的情况下, 考虑未来所有可能排队等待时间的影响, 基于两阶段随机规划理论估计任务在边缘计算时的时延和能耗期望值; 最后通过衡量本地和边缘计算时的时延估计和能耗预算得到卸载决策. 若卸载决策 $\pi_i = 1$, 即用户选择将任务卸载到 MEC 服务器处理, 则基于两阶段随机规划理论将卸载过程分为如下两个阶段.

第 1 阶段: 用户在没有观察到 MEC 服务器不确定性排队等待时间的情况下, 考虑未来所有可能排队时间的影响, 作出第 1 阶段的传输功率分配策略 p_i;

第 2 阶段: 当任务上传至 MEC 服务器, 排队等待时间实现已知, 在获得排队等待时间的实现 $T_i^{\mathrm{w}}(\omega)$ 和第 1 阶段的传输功率策略 p_i 的条件下, MEC 服务器将根据用户的计算时延约束采取最优追索动作 $f_{i,\omega}^{\mathrm{r}}\left(p_i, T_i^{\mathrm{w}}(\omega)\right)$, 以弥补第 1 阶段策略的不准确预测 [14].

3.3 两阶段随机规划问题建模

结合上文分析, 可得到基于两阶段随机规划的任务卸载和资源分配问题的期望值模型:

$$\mathrm{P1}: \min_{\boldsymbol{\pi}, \boldsymbol{f}^{\mathrm{l}}, \boldsymbol{f}^{\mathrm{r}}, \boldsymbol{p}} \underbrace{\sum_{i=1}^{N}(1-\pi_i)\kappa_i D_i L_i f_i^{\mathrm{l}}}_{\text{本地计算}} + \underbrace{\sum_{i=1}^{N}\pi_i \frac{D_i p_i}{R_i(p_i)}}_{\text{边缘计算第 1 阶段}} + \underbrace{\mathbb{E}_{\Omega}\left[\sum_{i=1}^{N}\pi_i \gamma D_i L_i f_{i,\omega}^{\mathrm{r}}\left(p_i, T_i^{\mathrm{w}}(\omega)\right)\right]}_{\text{边缘计算第 2 阶段}} \tag{8}$$

$$\text{s.t.} \quad p_i^{\min} \leqslant p_i \leqslant p_i^{\max}, \tag{8a}$$

$$f_i^{\mathrm{l,min}} \leqslant f_i^{\mathrm{l}} \leqslant f_i^{\mathrm{l,max}}, \tag{8b}$$

$$f^{\mathrm{r,min}} \leqslant f_i^{\mathrm{r}} \leqslant f^{\mathrm{r,max}}, \tag{8c}$$

$$T_i^{\mathrm{l}}\left(f_i^{\mathrm{l}}\right) \leqslant \tau_i, \tag{8d}$$

$$T_i^{\mathrm{c}}(p_i) + T_i^{\mathrm{w}}(\omega) + T_i^{\mathrm{e}}\left(f_{i,\omega}^{\mathrm{r}}\right) \leqslant \tau_i, \tag{8e}$$

$$\pi_i \in \{0, 1\}, \tag{8f}$$

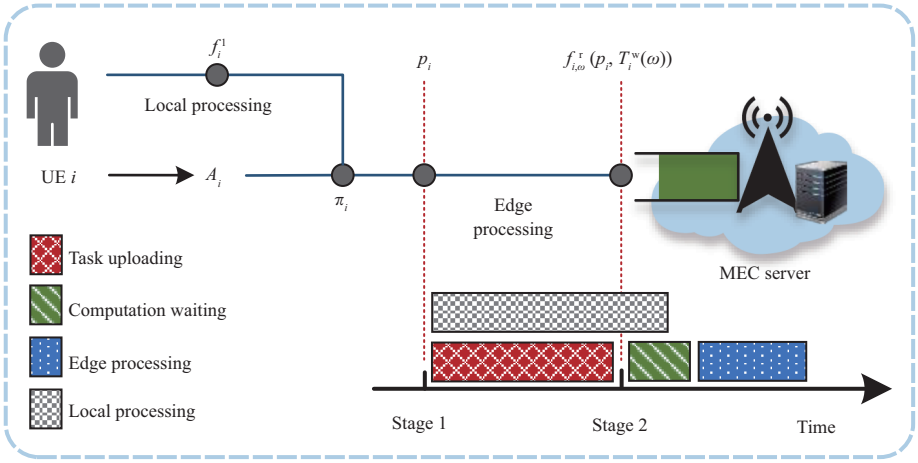

图 1 (网络版彩图) 两阶段卸载模型

Figure 1 (Color online) Two-stage offloading model

其中, $\boldsymbol{\pi} = \{\pi_1, \pi_2, \ldots, \pi_N\}$ 表示任务卸载决策集合; $\boldsymbol{f}^{\mathrm{l}} = \left\{f_1^{\mathrm{l}}, f_2^{\mathrm{l}}, \ldots, f_N^{\mathrm{l}}\right\}$ 表示本地 CPU 频率资源分配策略集合; $\boldsymbol{f}^{\mathrm{r}} = \left\{\boldsymbol{f}_{1,\Omega}^{\mathrm{r}}, \boldsymbol{f}_{2,\Omega}^{\mathrm{r}}, \ldots, \boldsymbol{f}_{N,\Omega}^{\mathrm{r}}\right\}$ 表示所有组合场景下 MEC 服务器 CPU 频率资源分配策略集合, 其中 $\boldsymbol{f}_{i,\Omega}^{\mathrm{r}} = \{f_{i,\omega_1}^{\mathrm{r}}, f_{i,\omega_2}^{\mathrm{r}}, \ldots, f_{i,|\Omega_i|^N}^{\mathrm{r}}\}$, $|\Omega_i|^N$ 表示组合场景空间的大小; $\boldsymbol{p} = \{p_1, p_2, \ldots, p_N\}$ 表示任务传输功率分配策略集合; $\mathbb{E}_{\Omega}[\cdot]$ 表示组合场景的期望; $p_i^{\min}$ 和 $p_i^{\max}$ 分别表示传输功率的最小值和最大值; $f_i^{\mathrm{l,min}}$ 和 $f_i^{\mathrm{l,max}}$ 分别表示本地 CPU 频率的最小值和最大值; $f^{\mathrm{r,min}}$ 和 $f^{\mathrm{r,max}}$ 分别表示 MEC 服务器 CPU 频率的最小值和最大值; $T_i^{\mathrm{w}}(\omega)$ 表示组合实现为 ω 时的排队等待时间. 约束条件式 (8d) 和 (8e) 分别表示在本地和 MEC 服务器执行任务的时间须在其时延要求范围内.

3.4 基于随机模拟的样本均值近似问题建模

本文考虑了 MEC 服务器的不确定性排队等待时间环境下的任务卸载和资源分配优化问题, 并将优化问题建模为一个两阶段随机规划问题. 然而, 求解两阶段随机规划问题通常面临着 “维数灾难” 的挑战, 这将导致较高的计算复杂度. 例如, 当所有用户的场景空间大小为 10000 时, 组合场景 Ω 的数量将达到 10000^N, 求解场景空间大小如此庞大的两阶段随机规划问题是困难且不现实的.

为了解决两阶段随机规划中计算复杂度较高的问题, 本文首先采用随机模拟方法从场景空间 Ω_i 中随机抽取 K 个独立同分布的样本, 并组成样本场景 $\Omega_i' = \left\{T_{i,1}^{\mathrm{w}}, T_{i,2}^{\mathrm{w}}, \ldots, T_{i,K}^{\mathrm{w}}\right\} \in \Omega_i$, 其中 $T_{i,k}^{\mathrm{w}}$, $k = 1, 2, \cdots, K$ 表示该样本中的第 k 个实现. 进一步考虑所有用户, 可得到基于样本空间的样本组合场景 $\Omega' = \{\omega_1, \omega_2, \ldots, \omega_S\}$, 其中 $S = K^N$ 表示该样本组合场景的大小, ω_s, $s = 1, 2, \ldots, S$ 表示该样本组合场景中的第 s 个组合实现. 接着, 基于样本均值近似方法将问题 P1 中的期望值模型近似为样本均值模型:

$$\mathrm{P2}: \quad \min_{\boldsymbol{\pi}, \boldsymbol{f}^{\mathrm{l}}, \boldsymbol{f}^{\mathrm{r}}, \boldsymbol{p}} \quad \underbrace{\sum_{i=1}^{N}(1-\pi_i)\kappa_i D_i L_i f_i^{\mathrm{l}}}_{\text{本地计算}} + \underbrace{\sum_{i=1}^{N}\pi_i \frac{D_i p_i}{R_i(p_i)}}_{\text{边缘计算第 1 阶段}} + \underbrace{\frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{N}\pi_i \gamma D_i L_i f_{i,\omega_s}^{\mathrm{r}}\left(p_i, T_i^{\mathrm{w}}(\omega_s)\right)}_{\text{边缘计算第 2 阶段}} \tag{9}$$

$$\text{s.t.} \quad \text{式 (8a)} \sim \text{(8f)}.$$

4 基于随机模拟的任务卸载和资源分配两阶段随机规划算法

在优化问题 P2 中, π_i 为 $\{0,1\}$ 二元决策变量, 边缘计算第 1 阶段目标函数是关于 p_i 的非线性函数, 因此, 优化问题 P2 为 MINLP 问题, 且为 NP 难问题[15]. 为了求解优化问题 P2 的全局最优解, 本文提出了一种基于随机模拟的任务卸载和资源分配两阶段随机规划算法 SS_2SSP. 首先, 观察到本地计算时的优化变量、边缘计算时的优化变量, 以及卸载决策变量之间完全解耦, 因此, 为解决 MINLP 问题, 本文将优化问题 P2 解耦为本地计算资源分配、传输功率和边缘计算资源分配, 以及卸载决策 3 个子问题; 其次, 采用拉格朗日乘子法获得本地计算资源最优分配策略, 同时采用遗传算法获得传输功率和边缘计算资源的最优分配策略; 最后, 通过分析本地计算和边缘计算的时延估计和能耗预算获得最优任务卸载决策. 为便于分析, 本文接下来只针对一个用户求解其最优策略, 由于每个用户之间相互独立, 最优策略容易扩展到多用户场景.

4.1 本地计算资源分配

假设卸载决策变量 $\pi_i=0$, 即 UE i 选择将任务在本地处理, 则优化问题 P2 可写为本地计算资源分配子问题:

$$\begin{aligned}\text{P2-1}: \quad &\min_{f_i^{\mathrm{l}}} \quad \kappa_i D_i L_i f_i^{\mathrm{l}} \\ &\text{s.t.} \quad \text{式 (8b) 和 (8d).}\end{aligned} \tag{10}$$

在优化问题 P2-1 中, 目标函数是关于 f_i^{l} 的仿射函数, 约束条件式 (8d) 是关于 f_i^{l} 的凸约束, 因此, 优化问题 P2-1 为凸优化问题. 本文采用拉格朗日乘子法求解优化问题 P2-1, 定义拉格朗日函数如下:

$$g_i\left(f_i^{\mathrm{l}},\mu_i,\lambda_i,\nu_i\right)=\kappa_i D_i L_i f_i^{\mathrm{l}}+\mu_i\left(f_i^{\mathrm{l,min}}-f_i^{\mathrm{l}}\right)+\lambda_i\left(f_i^{\mathrm{l}}-f_i^{\mathrm{l,max}}\right)+\nu_i\left(\frac{D_i L_i}{f_i^{\mathrm{l}}}-\tau_i\right), \tag{11}$$

其中, μ_i, λ_i 和 ν_i 均表示拉格朗日乘子. 进一步可得问题式 (11) 的 KKT (karush-kuhn-tucker) 条件为

$$\begin{cases}\dfrac{\partial g_i\left(f_i^{\mathrm{l}},\mu_i,\lambda_i,\nu_i\right)}{\partial f_i^{\mathrm{l}}}\Big|_{f_i^{\mathrm{l}}=f_i^{\mathrm{l}*}}=0, \\ \mu_i\left(f_i^{\mathrm{l,min}}-f_i^{\mathrm{l}*}\right)=0, \\ \lambda_i\left(f_i^{\mathrm{l}*}-f_i^{\mathrm{l,max}}\right)=0, \\ \nu_i\left(T_i^{\mathrm{l}}\left(f_i^{\mathrm{l}*}\right)-\tau_i\right)=0, \\ f_i^{\mathrm{l,min}}\leqslant f_i^{\mathrm{l}*}\leqslant f_i^{\mathrm{l,max}}, \\ T_i^{\mathrm{l}}\left(f_i^{\mathrm{l}*}\right)\leqslant\tau_i, \\ \mu_i,\lambda_i,\nu_i\geqslant 0.\end{cases} \tag{12}$$

通过求解 KKT 条件, 可得到任务在本地计算时最优 CPU 频率分配策略的闭合表达式为

$$f_i^{\mathrm{l}*}=\max\left\{\min\left\{\frac{D_i L_i}{\tau_i},f_i^{\mathrm{l,max}}\right\},f_i^{\mathrm{l,min}}\right\}. \tag{13}$$

4.2 传输功率和边缘计算资源分配

假设卸载决策变量 $\pi_i = 1$, 即 UE i 选择将任务卸载到 MEC 服务器计算, 则优化问题 P2 可写为传输功率和边缘计算资源分配子问题:

$$\text{P2-2}: \quad \min_{\boldsymbol{f}^{\mathrm{r}}_{i,\Omega'}, p_i} \quad \underbrace{\sum_{i=1}^{N} \pi_i \frac{D_i p_i}{R_i\left(p_i\right)}}_{\text{边缘计算第 1 阶段}} + \underbrace{\frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{N} \pi_i \gamma D_i L_i f^{\mathrm{r}}_{i,\omega_s}\left(p_i, T^{\mathrm{w}}_i\left(\omega_s\right)\right)}_{\text{边缘计算第 2 阶段}} \tag{14}$$

$$\text{s.t.} \quad \text{式 (8a), (8c) 和 (8e).}$$

在优化问题 P2-2 中, 由于边缘计算第一阶段目标函数是关于 p_i 的非凸函数, 因此, 优化问题 P2-2 为非凸问题. 遗传算法是一种不依赖目标函数凹凸性的启发式全局优化算法, 并且特别适用于高维决策问题 [16], 为此, 本文接下来采用遗传算法求取问题 P2-2 的全局最优解, 主要包括以下步骤.

(1) 编码. 本文将每一个可行的传输功率进行浮点向量编码, 每一个浮点向量表示一个染色体, 浮点向量维数与解向量维数一致.

(2) 初始种群. 定义 M 表示种群大小, 初始化过程随机产生 M 个染色体. 从用户传输功率的可行域中随机产生一个点, 并检验其是否满足约束条件, 如果满足则作为一个染色体, 否则, 重新产生一个随机点, 直到满足约束条件. 重复以上过程 M 次, 产生 M 个染色体作为初始种群, 记为 $p_{i,1}, p_{i,2}, \ldots, p_{i,M}$.

(3) 评价函数. 评价函数用来对种群中的每个染色体设定一个概率, 以使该染色体被选择的可能性与种群中其他染色体的适应度成比例, 适应度强的染色体被选择产生后代的几率较大. 本文利用优化问题 P2-2 的最优值作为染色体的适应度, 利用适应度的大小决定一个序, 并按照此序确定染色体被选择的概率. 最优值越小, 适应度越强, 即作为父本繁殖下一代的概率越大. 具体操作为: 对于每个染色体 $p_{i,m}$, $m = 1, 2, \ldots, M$, 首先求解如下最小化问题的目标函数最优值作为适应度值:

$$\min_{\boldsymbol{f}^{\mathrm{r}}_{i,\Omega'}} \quad \underbrace{\sum_{i=1}^{N} \pi_i \frac{D_i p_{i,m}}{R_i\left(p_{i,m}\right)}}_{\text{边缘计算第 1 阶段}} + \underbrace{\frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{N} \pi_i \gamma D_i L_i f^{\mathrm{r}}_{i,\omega_s}\left(p_{i,m}, T^{\mathrm{w}}_i\left(\omega_s\right)\right)}_{\text{边缘计算第 2 阶段}} \tag{15}$$

$$\text{s.t.} \quad \text{式 (8c) 和 (8e).}$$

在优化问题式 (15) 中, 边缘计算第 1 阶段的目标值为常数, 边缘计算第 2 阶段目标函数是关于 $f^{\mathrm{r}}_{i,\omega_s}$ 的仿射函数, 约束条件式 (8e) 是关于 $f^{\mathrm{r}}_{i,\omega_s}$ 的凸约束, 因此, 优化问题式 (15) 为凸优化问题. 采用拉格朗日乘子法和 KKT 条件可以求得每个场景下 MEC 服务器最优 CPU 频率分配策略的闭合表达式为

$$f^{\mathrm{r}*}_{i,\omega_s} = \max\left\{\min\left\{\frac{D_i L_i}{\tau_i - T^{\mathrm{c}}_i\left(p_{i,m}\right) - T^{\mathrm{w}}_i\left(\omega_s\right)}, f^{\mathrm{r,max}}\right\}, f^{\mathrm{r,min}}\right\}, \tag{16}$$

将所有组合实现 ω_s 下的最优 CPU 频率值带入式 (15), 可求得每个染色体相应的适应度值, 并按照适应度值从小到大对染色体进行排序, 据此序定义评价函数如下:

$$\mathrm{eval}\left(p_{i,m}\right) = a(1-a)^{m-1}, \quad m = 1, 2, \ldots, M, \tag{17}$$

其中 $a \in (0, 1)$. 对于染色体 $p_{i,m}$, 评价函数值越大, 被选择作为父本繁殖下一代的概率越大.

(4) 选择. 对于每一个染色体 $p_{i,m}$, 计算累积概率 $q_{i,m} = \sum_{j=1}^{m} \mathrm{eval}\left(p_{i,j}\right), j = 1, 2, \ldots, M$, 从区间 $(0, q_{i,M})$ 中生成一个随机数 r, 若满足 $q_{i,m-1} < r < q_{i,m}$, $q_0 = 0$, 则选择染色体 $p_{i,m}$ 作为父本繁殖下一代, 依此规则选择 M 个染色体.

(5) 交叉. 定义 P_{c} 为交叉概率, 则种群中有期望值为 $P_{\mathrm{c}}M$ 个染色体进行交叉操作. 假设 $p_{i,1}$ 和 $p_{i,2}$ 为要交叉的两个染色体, 首先从区间 $(0,1)$ 中生成一个随机数 c, 直到 $p'_{i,1}=cp_{i,1}+(1-c)\,p_{i,2}$ 和 $p'_{i,2}=(1-c)\,p_{i,1}+cp_{i,2}$ 满足约束条件, 则 $p'_{i,1}$ 和 $p'_{i,2}$ 将代替原染色体 $p_{i,1}$ 和 $p_{i,2}$ 成为两个新的染色体.

(6) 变异. 定义 P_{u} 为变异概率, 则种群中有期望值为 $P_{\mathrm{u}}M$ 个染色体进行变异操作. 假设 $p_{i,1}$ 为要变异的染色体, 随机选择一个变异方向 ϕ, 然后从 $(0,p_i^{\max})$ 中生成一个随机数 α, 直到 $p'_{i,1}=p_{i,1}+\alpha\phi$ 满足约束条件, 则 $p'_{i,1}$ 将取代原染色体 $p_{i,1}$ 成为新的染色体.

(7) 终止条件. 经过选择、交叉和变异操作, 可以得到一个新的种群, 并准备进行下一代进化. 如果上述步骤达到了给定的循环次数, 则遗传算法终止. 算法终止后, 从最后一代选择一个适应度最高的染色体, 即得到优化问题的全局最优解.

4.3 卸载决策

用户通过对本地计算和边缘计算的能耗预算和时延估计得到卸载决策, 在获得本地计算资源、传输功率和边缘计算资源最优分配策略的情况下, 优化问题 P2 可写为任务卸载决策子问题:

$$\text{P2-3}:\quad \min_{\pi_i}\ \underbrace{\sum_{i=1}^{N}(1-\pi_i)\,\kappa_i D_i L_i f_i^{\mathrm{l}*}}_{\text{本地计算}}+\underbrace{\sum_{i=1}^{N}\pi_i\frac{D_i p_i^*}{R_i\left(p_i^*\right)}}_{\text{边缘计算第 1 阶段}}+\underbrace{\frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{N}\pi_i\gamma D_i L_i f_{i,\omega_s}^{\mathrm{r}*}\left(p_i^*,T_i^{\mathrm{w}}\left(\omega_s\right)\right)}_{\text{边缘计算第 2 阶段}}\tag{18}$$

$$\text{s.t.}\quad \text{式 (8f).}$$

在优化问题 P2-3 中, 唯一的变量是卸载决策 $\pi_i\in\{0,1\}$, 因此, 优化问题 P2-3 为 0-1 规划问题. 本文采用枚举法分别将 0 和 1 带入问题 P2-3, 再比较目标函数值的大小, 较小的目标函数值对应的 π_i 即为最优卸载决策, 最优卸载决策的闭合表达式如下:

$$\pi_i^*=\begin{cases}0, & \kappa_i D_i L_i f_i^{\mathrm{l}*}\leqslant\dfrac{D_i p_i^*}{R_i\left(p_i^*\right)}+\dfrac{1}{S}\displaystyle\sum_{s=1}^{S}\gamma D_i L_i f_{i,\omega_s}^{\mathrm{r}*}\left(p_i^*,T_i^{\mathrm{w}}\left(\omega_s\right)\right),\\ 1, & \kappa_i D_i L_i f_i^{\mathrm{l}*}>\dfrac{D_i p_i^*}{R_i\left(p_i^*\right)}+\dfrac{1}{S}\displaystyle\sum_{s=1}^{S}\gamma D_i L_i f_{i,\omega_s}^{\mathrm{r}*}\left(p_i^*,T_i^{\mathrm{w}}\left(\omega_s\right)\right).\end{cases}\tag{19}$$

5 仿真与分析

本节利用 MATLAB R2017b 工具对 SS_2SSP 算法性能进行仿真验证, 实验中的对比算法包括:

(1) 基于最大排队等待时间的静态卸载算法 MWT_SO (maximum waiting time based static offloading algorithm) [7]. 用户在基于 MEC 服务器最大排队时间的静态网络中[1)], 对本地 CPU 频率、传输功率、MEC 服务器的 CPU 频率和卸载决策进行优化.

(2) 贪婪式卸载算法 Greedy [6]. 用户在基于 MEC 服务器最大排队时间的静态网络中采用 MEC 服务器最大 CPU 频率的贪婪式卸载策略对本地 CPU 频率、传输功率和卸载决策进行优化.

(3) 本地执行算法 OnlyLocal. 任务只在本地计算.

1) 在 MWT_SO 算法中, 用户端基于固定的 T_i^{w} 作出任务卸载和资源分配策略, 这意味着任务上传至 MEC 服务器后没有追索动作, 由于 MEC 服务器端排队等待时间的不确定性, 如果实际排队等待时间大于 T_i^{w}, 将会导致任务计算失败, 为了保证在计算时延要求范围内成功处理任务, 本文采用基于最大排队等待时间的静态网络.

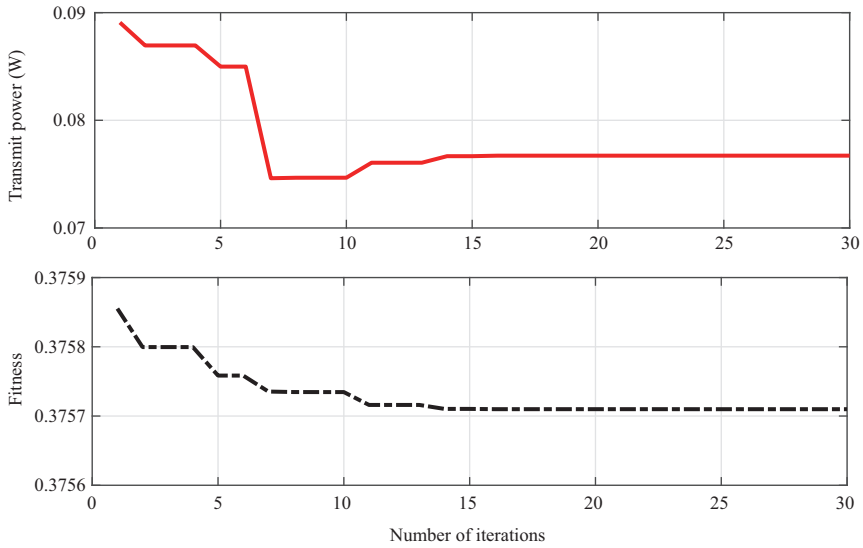

图 2 (网络版彩图) 遗传算法迭代过程

Figure 2 (Color online) Iteration process of the genetic algorithm

5.1 仿真参数设置

本文考虑单个 MEC 服务器多个用户场景, MEC 服务器位于半径为 200 m 的蜂窝网络中心, 用户均匀分布在 MEC 服务器覆盖范围内. 设置信道增益 $H_i = d_i^{-4}$, 其中, d_i 是 UE i 与 MEC 服务器之间的传输距离. 传输带宽 B_0 为 30 MHz; 背景噪声功率 N_0 为 -50 dBm; 用户传输功率在 5~33 dBm 之间; 任务的计算密度 L_i 为 700 cycles/bit; 本地 CPU 频率范围为 100~2500 MHz; MEC 服务器的 CPU 频率范围为 500~5000 MHz; 本地有效能量系数 κ_i 为 10^{-19} W $\cdot$ s^2/cycle^2; MEC 服务器有效能量系数 γ 为 10^{-20} W$\cdot$s^2/cycle^2; 遗传算法中种群大小 M 为 140, 交叉概率 $P_{\rm c}$ 为 0.6, 变异概率 $P_{\rm u}$ 为 0.01, 迭代次数为 50 次; MEC 服务器排队等待时间服从指数分布, 最大排队等待时间设置为 5 s; 若无特别说明, 排队等待时间样本数量 K 均为 100 个, 每组实验独立重复次数为 10 次.

5.2 性能分析

首先对子问题 P2-2 中的遗传算法收敛速度进行评估, 图 2 给出了遗传算法的迭代过程, 实验中用户数为 1 个, 任务量为 20 Mbit, 计算时延需求为 8 s, MEC 服务器排队等待时间服从均值为 2 s 的指数分布. 从图 2 中可以看出, 遗传算法大概迭代 15 次左右就能达到收敛, 而子问题 P2-1 和子问题 P2-3 都可以直接获得最优解的闭合表达式, 这表明 SS_2SSP 算法具有较好的收敛性能.

图 3 给出了 MEC 服务器排队等待时间对 SS_2SSP 算法性能的影响. 实验中包含 3 个不同任务量和不同计算时延约束的用户 (UE 1: $D_1 = 20$ Mbit, $\tau_1 = 8$ s; UE 2: $D_2 = 20$ Mbit, $\tau_2 = 10$ s; UE 3: $D_3 = 35$ Mbit, $\tau_3 = 10$ s), MEC 服务器排队等待时间服从指数分布. 从图 3(a) 和 (b) 中可以看出, 随着 MEC 服务器排队等待时间的递增, 为了满足用户的计算时延需求, 传输功率和 MEC 服务器平均 CPU 频率随之增加. 然而, 根据图 3(c) 所示, 系统能耗随着排队等待时间的增加而增加, 这是由于传输功率和 CPU 频率的增加导致了传输能耗和计算能耗的增加, 这表明满足用户计算时延需求是以系统能耗为代价的. 此外, 通过对比图 3 中 UE 1, UE 2 和 UE 3 可知, 用户计算时延要求越高或任务量越大, 所需传输功率和 CPU 频率越大, 系统能耗也越高.

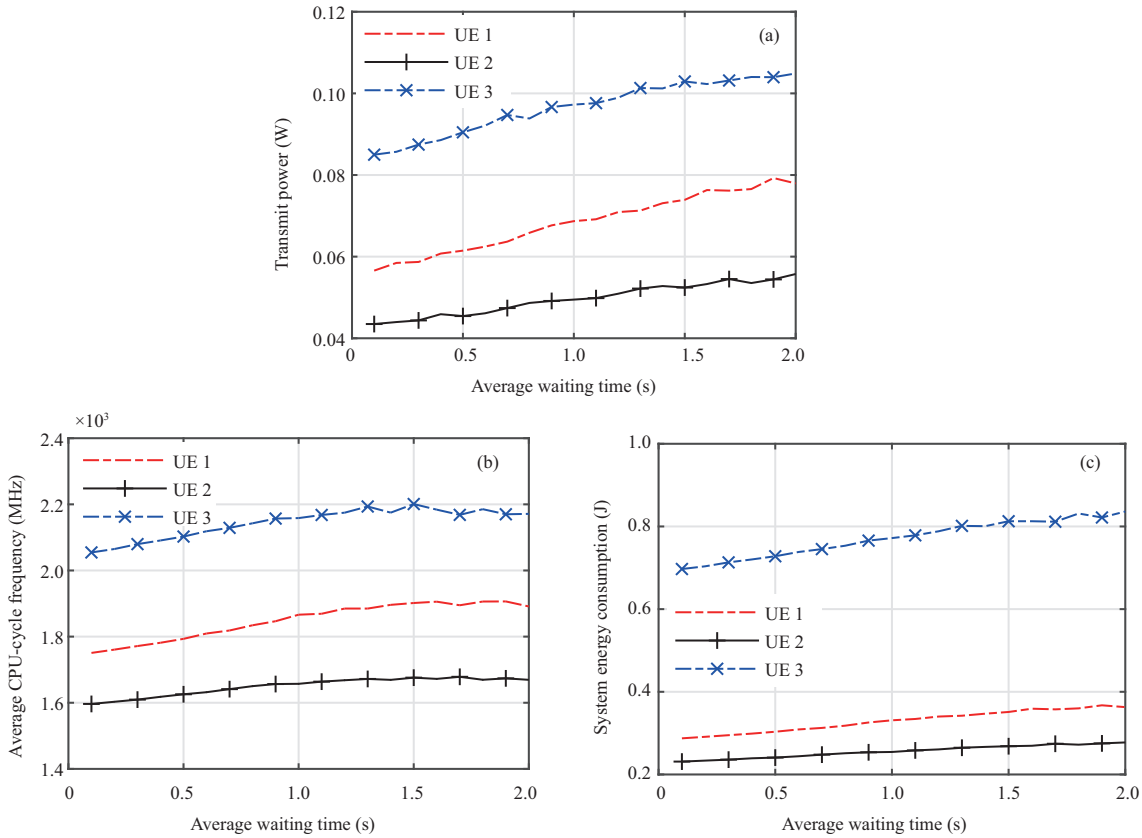

图 3　(网络版彩图) 排队等待时间对 SS_2SSP 性能影响

Figure 3　(Color online) The effect of average waiting time on SS_2SSP performance

为了验证 SS_2SSP 算法的性能, 本实验将其与传统的 MWT_SO 算法、Greedy 算法和 OnlyLocal 算法进行对比分析. 图 4(a) 所示为不同算法下系统能耗随着用户任务量的变化过程图. 实验中包含 1 个用户, 统计该用户在不同任务量下的系统能耗, 每个任务计算时延要求均为 1 s/Mbit, MEC 服务器排队等待时间服从均值为 2 s 的指数分布. 从图 4(a) 中可以看出, 随着任务量的增加, 所有算法的系统能耗随之增加, 而本文所提 SS_2SSP 算法具有最低的系统总能耗, 这是由于 SS_2SSP 算法在第 1 阶段制定传输功率策略时考虑了未来所有可能排队等待时间的影响, 待任务上传至 MEC 服务器后, 排队等待时间已知, 在获得排队等待时间和第 1 阶段传输功率决策的条件下, MEC 服务器根据用户的计算时延约束条件采取第 2 阶段的追索策略, 即 CPU 频率分配策略, 这将弥补第 1 阶段策略的不准确预测. 而 MWT_SO 算法是基于 MEC 服务器最大排队等待时间的静态卸载策略, 这意味着 MWT_SO 算法将会以更高的能耗去满足时延需求. Greedy 算法基于最大的排队等待时间, 且贪婪地使用 MEC 服务器最大 CPU 频率资源, 相比 MWT_SO 和 SS_2SSP 算法具有更高的系统能耗. OnlyLocal 算法中任务全在本地执行, 由于用户设备计算性能和计算资源的限制, 较其他算法具有最高的系统能耗.

图 4(b) 所示为不同算法下系统能耗随着时延约束的变化过程图. 实验中用户数为 1 个, 任务量大小为 20 Mbit, MEC 服务器排队等待时间服从均值为 2 s 的指数分布. 从图 4(b) 可以看出, 随着计算时延的增加, 所有算法的系统能耗随之减小, 本文所提 SS_2SSP 算法具有最低的系统能耗, 而 MWT_SO 算法基于最大排队等待时间具有较高的系统能耗. 与 MWT_SO 算法对比, Greedy 算法没有对 MEC

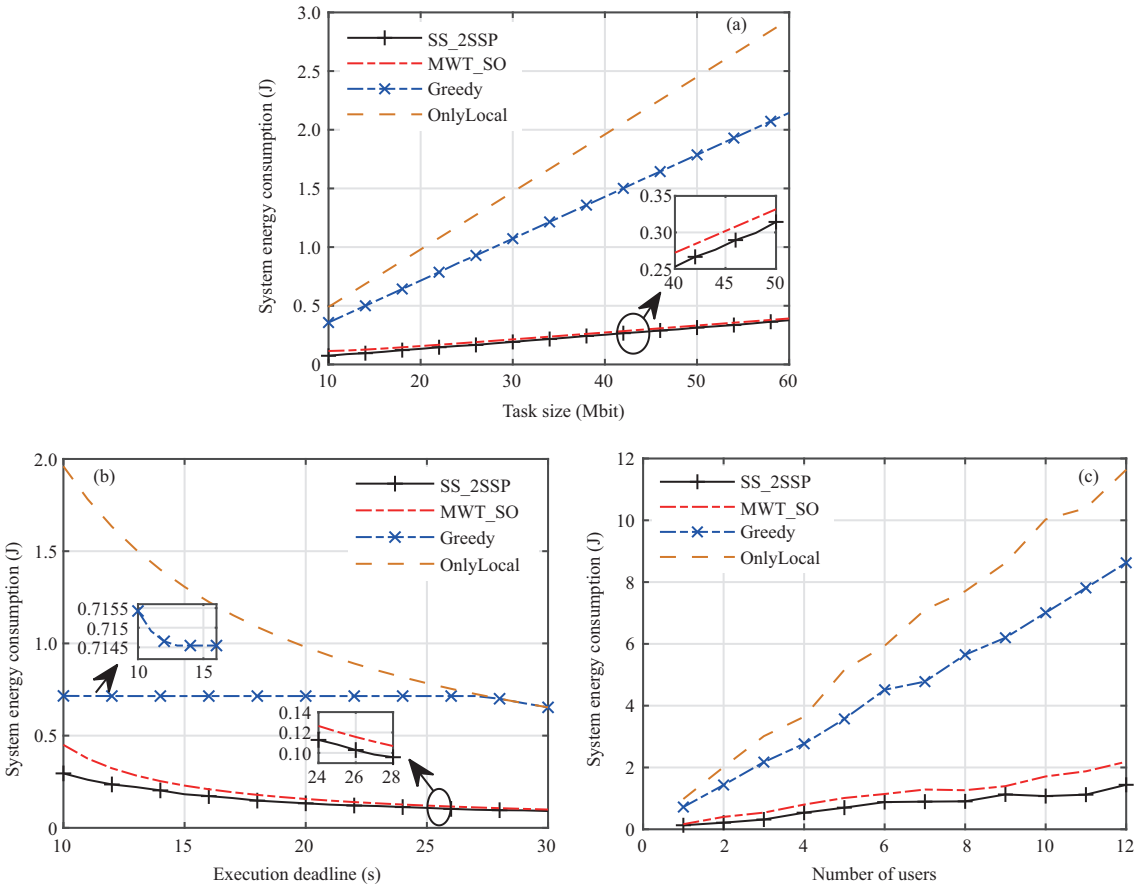

图 4 (网络版彩图) 不同算法下系统能耗随 (a) 任务量, (b) 时延约束, (c) 用户数的变化过程图

Figure 4 (Color online) System energy consumption versus (a) task size, (b) execution delay, (c) number of users under different algorithms

服务器 CPU 频率进行优化, 因此, 具有更高的系统能耗. OnlyLocal 算法中任务只在本地处理, 与其他算法相比, 具有最高的系统能耗.

图 4(c) 所示为不同算法下系统能耗随着用户数量的变化过程图. 实验中设置每个用户的任务量大小服从区间为 (10, 30) 的均匀分布, 单位为 Mbit, 每个用户的任务时延要求均为 1 s/Mbit, MEC 服务器排队等待时间服从均值为 2 s 的指数分布, 排队等待时间样本数量 $K = 5$. 从图 4(c) 中可以看出, 随着用户数量的增加, 所有算法系统能耗随之增加, 可以发现 SS_2SSP 和 MWT_SO 算法的系统能耗远远低于 Greedy 和 OnlyLocal 算法, 这是由于 SS_2SSP 和 MWT_SO 算法对 MEC 服务器的 CPU 频率进行了优化. 这表明合理分配边缘节点的计算资源可以有效降低 MEC 网络系统能耗, 同时满足用户的计算需求. 此外, 本文所提的 SS_2SSP 算法和 MWT_SO 算法相比具有更低的系统能耗, 并且随着用户数量的增加, SS_2SSP 算法的系统能耗愈发具有明显优势.

为了验证随机模拟方法中采样数量对 SS_2SSP 性能的影响, 图 5 给出了不同样本数下系统能耗对比. 实验中用户数为 1 个, 任务量大小为 20 Mbit, MEC 服务器排队等待时间服从均值为 2 s 的指数分布. 从图 5 中可以看出, 抽样数量越多, 系统能耗的波动越小, 这意味着得到的结果越精确.

图 6 描述了 SS_2SSP 算法的计算复杂度. 实验中每个用户的任务量大小均为 20 Mbit, MEC 服务器排队等待时间服从均值为 2 s 的指数分布. 从图 6 中可以看出, 随着样本数量的增加, 系统运行时间

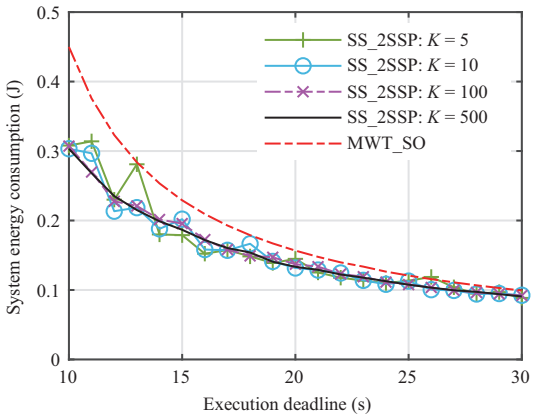

图 5 (网络版彩图) 不同样本数下系统能耗对比

Figure 5 (Color online) System energy consumption comparison under different K

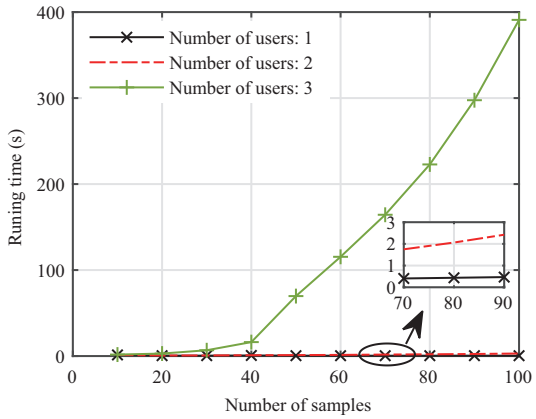

图 6 (网络版彩图) 运行时间 vs. 样本数

Figure 6 (Color online) Running time versus the number of samples

随之增加. 此外, 由于组合场景数量满足 $S = K^N$, 当用户数量增加时, 组合场景数量将呈指数型增长, 这将导致较高的计算复杂度. 这时可以通过随机模拟方法选择合适的样本数量以降低计算复杂度, 同时获得近似最优解.

6 结束语

本文研究了移动边缘计算中任务在边缘服务器的随机排队等待时间导致的计算时延不确定问题, 提出了一种基于随机模拟的任务卸载和资源分配两阶段随机规划算法. 首先, 将任务在 MEC 服务器的排队等待时间建模一组随机变量, 并在时延约束条件下, 以最小化系统能耗为目标, 将优化问题建模为两阶段随机规划问题; 其次, 为了降低两阶段随机规划问题的计算复杂度, 提出了一种基于随机模拟的样本均值近似优化问题, 由于该优化问题为 MINLP 问题, 将优化问题解耦为本地计算资源分配、传输功率和边缘计算资源分配, 以及卸载决策 3 个子问题. 最后, 通过拉格朗日乘子法和遗传算法分别获得本地 CPU 频率最优分配策略, 以及传输功率和边缘 CPU 频率最优分配策略, 并对比分析本地计算和边缘计算的时延估计和能耗预算获得最优任务卸载决策. 仿真结果表明, 与传统算法相比, 本文所提算法能够在时延不确定的网络中满足任务计算时延的要求, 并且有效地降低了系统能耗. 本文没有考虑用户端移动性问题, 在实际的 MEC 网络环境下, 用户的随机移动往往导致接入网侧的随机切换, 从而影响任务卸载和资源分配策略的定制, 在未来的工作中, 将会考虑用户随机移动场景下的卸载策略.

参考文献

1 Zhao Y J, Yu G H, Xu H Q. 6G mobile communication networks: vision, challenges, and key technologies. Sci Sin Inform, 2019, 49: 963–987 [赵亚军, 郁光辉, 徐汉青. 6G 移动通信网络: 愿景, 挑战与关键技术. 中国科学: 信息科学, 2019, 49: 963–987]

2 Abbas N, Zhang Y, Taherkordi A, et al. Mobile edge computing: a survey. IEEE Internet Things J, 2018, 5: 450–465

3 Liu B H, Liu C X D, Peng M G. Resource allocation for energy-efficient MEC in NOMA-enabled massive IoT networks. IEEE J Sel Areas Commun, 2021, 39: 1015–1027

4 Chen H, Zhao D M, Chen Q B, et al. Joint computation offloading and radio resource allocations in small-cell wireless cellular networks. IEEE Trans Green Commun Netw, 2020, 4: 745–758
5 Li H L, Xu H T, Zhou C C, et al. Joint optimization strategy of computation offloading and resource allocation in multi-access edge computing environment. IEEE Trans Veh Technol, 2020, 69: 10214–10226
6 Guo S T, Liu J D, Yang Y Y, et al. Energy-efficient dynamic computation offloading and cooperative task scheduling in mobile cloud computing. IEEE Trans Mobile Comput, 2019, 18: 319–333
7 Malik R, Vu M. Energy-efficient computation offloading in delay-constrained massive MIMO enabled edge network using data partitioning. IEEE Trans Wireless Commun, 2020, 19: 6977–6991
8 Meng X L, Wang W, Wang Y, et al. Closed-form delay-optimal computation offloading in mobile edge computing systems. IEEE Trans Wireless Commun, 2019, 18: 4653–4667
9 Han D, Chen W, Fang Y G. Joint channel and queue aware scheduling for latency sensitive mobile edge computing with power constraints. IEEE Trans Wireless Commun, 2020, 19: 3938–3951
10 Merluzzi M, Lorenzo P D, Barbarossa S. Latency constrained dynamic computation offloading with energy harvesting IoT devices. In: Proceedings of IEEE Conference on Computer Communications Workshops, 2019. 750–755
11 Cao J Y, Feng W, Ge N, et al. Delay characterization of mobile-edge computing for 6G time-sensitive services. IEEE Internet Things J, 2021, 8: 3758–3773
12 Mao Y Y, You C S, Zhang J, et al. A survey on mobile edge computing: the communication perspective. IEEE Commun Surv Tutor, 2017, 19: 2322–2358
13 Xia S C, Yao Z X, Li Y, et al. Online distributed offloading and computing resource management with energy harvesting for heterogeneous MEC-enabled IoT. IEEE Trans Wireless Commun, 2021, 20: 6743–6757
14 Birge J R, Louveaux F. Introduction to Stochastic Programming. New York: Springer, 1997. 49-332
15 Burer S, Letchford A N. Non-convex mixed-integer nonlinear programming: a survey. Surv Oper Res Manage Sci, 2012, 17: 97–106
16 Srinivas M, Patnaik L M. Genetic algorithms: a survey. Computer, 1994, 27: 17–26

Task offloading and resource allocation in an uncertain network

Zhixiu YAO, Shichao XIA & Yun LI*

School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065*, China*
* Corresponding author. E-mail: liyun@cqupt.edu.cn

Abstract With the emergence of massive compute-intensive and delay-sensitive applications, and driven by big data and low delay computation requirements, mobile edge computing (MEC) will play an important role in improving user experience and reducing energy consumption. However, MEC servers have limited computation resources and cannot respond quickly to the large amounts of bursting computation requirements, so computation waiting time at MEC servers is unavoidable and unpredictable. To meet the computation delay requirements of the applications in an uncertain computation waiting time network, this paper formulates a system energy consumption minimization problem with computation delay constraints and proposes a stochastic simulation based two-stage stochastic programming (SS_2SSP) algorithm for task offloading and resource allocation. The simulation results show that the SS_2SSP algorithm can meet the computation delay requirements of the applications while effectively reducing the system energy consumption.

Keywords mobile edge computing, uncertain network, task offloading, resource allocation, computation delay, two-stage stochastic programming