文章编号:1007-5321(2022)02-0091-07 **DOI**:10.13190/j.jbupt.2021-204

异构网络中任务卸载与资源分配联合优化算法

张雨晴[1], 李 云[1], 黄鸿锐[1], 庄宏成[2]
(1. 重庆邮电大学 通信与信息工程学院, 重庆 400065; 2. 华为科技有限公司, 深圳 518000)

摘要: 在有限的网络边缘资源约束下,考虑到业务的多样性和网络接入的异构性对任务卸载和计算资源分配的影响,在本地和服务器共同处理任务的背景下,提出了一种异构网络场景下结合李雅普诺夫优化理论和搜索树算法对任务卸载和计算资源分配的联合优化方法,分析了卸载收益与延迟之间的折中关系,优化了任务卸载与计算资源分配。同时,为了对搜索树进行快速分支定界,设计了一种卸载优先级准则。最后,通过仿真实验验证了所提算法的有效性和合理性。

关 键 词: 异构网络; 移动边缘计算; 任务卸载; 资源分配
中图分类号: TN92 **文献标志码**: A

Joint Optimization Algorithm for Task Offloading and Resource Allocation in Heterogeneous Networks

ZHANG Yuqing[1], LI Yun[1], HUANG Hongrui[1], ZHUANG Hongcheng[2]
(1. School of Communication and Information Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China; 2. Huawei Technologies Company Limited, Shenzhen 518000, China)

Abstract: Under the constraint of limited network edge resources, considering the impact of service diversity and network access heterogeneity on task offloading and computational resource allocation, a joint optimization algorithm for task offloading and computing power resources allocation under heterogeneous network is proposed to jointly handle local and server tasks. The proposed algorithm makes a trade-off between system revenue and task offloading delay by invoking Lyapunov theory and search tree algorithm. Furthermore, to quickly branch and bound the search tree, the offloading priority criterion is designed. Finally, the simulation results verify the effectiveness and rationality of the proposed algorithm.

Key words: heterogeneous network; mobile edge computing; task offloading; resources allocation

移动边缘计算(MEC,mobile edge computing)作为从云计算演变的新兴技术,将云资源下沉到网络边缘,有效提升了用户计算体验,且在一定程度上降低了计算密集型应用的负载和终端设备的能耗。

面向异构网络的 MEC 技术取得了一些进展[1-3]。Xia 等[1]将李雅普诺夫优化和搜索树算法结合起来,联合优化动态异构 MEC 网络下的任务卸载和计算资源分配。Hu 等[2]提出了一种基于李雅普诺夫优化和半定规划的在线移动性感知卸载和资源分配算法,通过优化物联网设备的采集能量等,实现了最小化长期总服务成本。Xia 等[3]将能量采集技术应用到 MEC,基于博弈论和李雅普诺夫优化理

收稿日期: 2021-09-27
基金项目: 国家自然科学基金项目(62071077); 重庆邮电大学博士研究生高端人才培养项目(BYJS201806)
作者简介: 张雨晴(1998—), 女, 硕士生。
通信作者: 李 云(1974—), 男, 教授, 博士生导师, 邮箱: liyun@cqupt.edu.cn。

论提出了一种在线分布式优化算法,更灵活地分配有限边缘云计算资源,提高了处理效率。

笔者在文献[1]的基础上,考虑到移动设备本地端,提出一种异构网络场景下的任务卸载和计算资源分配的联合优化方法。首先,考虑任务到达的随机性,根据用户端和服务器端计算资源、任务缓存优先级等因素制定任务卸载资源分配联合优化问题,又根据李雅普诺夫队列优化理论,将任务卸载与计算资源联合分配建模为一种混合整数非线性规划(MINLP, mixed integer non-ninear programming)问题,分析了卸载收益与延迟之间的折中关系;然后提出了一种基于搜索树的启发式算法来优化任务卸载与计算资源分配策略(TOST, heuristic task offloading algorithm based on search tree)。此外,为了对搜索树进行快速地分支定界,设计了一种卸载优先级准则(TOC, task offloading criteria);最后,通过仿真实验验证了TOST-TOC(heuristic task offloading algorithm based on search tree-task offloading criteria)算法的有效性和合理性。

1 系统模型

由若干无线访问接入点(AP, access point)和一个基站(BS, base station)组成的边缘网络如图1所示。所有无线AP都在BS的覆盖范围内,每个无线AP配有1个MEC服务器,表示为 $H=\{0,1,\cdots,N\}$,BS与中心云服务器相连,假设 n_j 表示第 j 个MEC服务器,$n_j=0$ 表示中心云服务器。

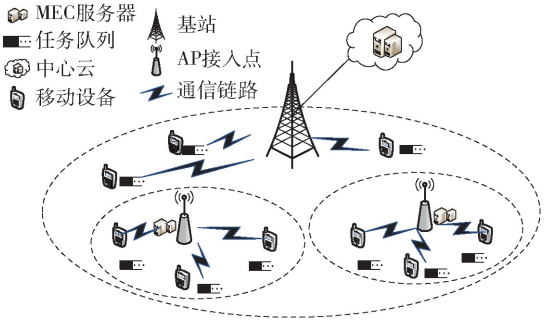

图1 基于移动边缘网络的系统模型

任一移动设备(MD, mobile device)可选择将任务卸载到服务器(MEC服务器或中心云服务器)或留在本地,MEC服务器为覆盖范围内的MD提供计算服务,能力不够时,再将任务转给中心云服务器。设 $Z=\{1,2,\cdots,M_j\}$ 表示 n_j 的MD集合,m_{ij} 表示 n_j 下的第 i 个MD。不同的MD具有不同的流量特征。每个MD维持一个队列来缓冲到达的任务,网络模型运行在离散时隙 $T=\{0,1,\cdots,T-1\}$ 中。

1.1 任务模型

笔者考虑无线接入网中支持数据分割的计算模型[4]。到达 m_{ij} 的任务量为 $A_{ij}(t)$ $(0\leqslant A_{ij}(t)\leqslant A_{ij}^{\max})$,其中 $A_{ij}^{\max}$ 表示到达 m_{ij} 的最大任务量。单位时隙内 n_j 下MD的任务到达率为 $\boldsymbol{\lambda}_j\triangleq\{\lambda_{1j},\lambda_{2j},\cdots,\lambda_{M_jj}\}$,其中 λ_{M_jj} 为 n_j 下第 M_j 个MD的任务到达率。时隙 t 内到达本地队列的任务量为 $a_{ij}(t)$,本地任务量的集合表示为 $\boldsymbol{\Lambda}=\{a_{ij}(t)\}_{i\in Z,t\in T}$,则到达卸载队列的任务量为 $c_{ij}(t)$,满足 $c_{ij}(t)=A_{ij}(t)-a_{ij}(t)$。$Q_{ij}^{\mathrm{l}}(t)$ 和 $Q_{ij}^{\mathrm{o}}(t)$ 分别表示 m_{ij} 在时隙 t 内本地任务队列和卸载任务队列的积压长度,$b_{ij}^{\mathrm{l}}(t)$ 和 $b_{ij}^{\mathrm{o}}(t)$ 分别表示 m_{ij} 在时隙 t 内本地和卸载处理的任务量。由此可得本地任务队列和卸载任务队列的更新规则,分别表示为

$$Q_{ij}^{\mathrm{l}}(t+1)=\max[Q_{ij}^{\mathrm{l}}(t)-b_{ij}^{\mathrm{l}}(t),0]+a_{ij}(t) \quad (1)$$

$$Q_{ij}^{\mathrm{o}}(t+1)=\max[Q_{ij}^{\mathrm{o}}(t)-b_{ij}^{\mathrm{o}}(t),0]+c_{ij}(t) \quad (2)$$

1.1.1 本地任务处理模型

定义 m_{ij} 在时隙 t 中的本地中央处理器(CPU, central processing unit)周期频率为 $f_{ij}^{\mathrm{l}}(t)$,并满足 $f_{\mathrm{l}}^{\min}\leqslant f_{ij}^{\mathrm{l}}(t)\leqslant f_{\mathrm{l}}^{\max}$,假设 $P_{ij}^{\mathrm{l}}(t)$ 为时隙 t 时本地计算的能耗,则 $P_{ij}^{\mathrm{l}}(t)$ 可以表示为

$$P_{ij}^{\mathrm{l}}(t)=\tau K(f_{ij}^{\mathrm{l}}(t))^2,\ i\in Z,\ j\in H,\ t\in T \quad (3)$$

其中,K 为恒定的功率系数,τ 为每个时隙的持续时间。

1.1.2 服务器任务处理模型

将 $I_{ij,\mathrm{o}}(t)\in\{0,1\}$,$\mathrm{o}\in\{\mathrm{m},\mathrm{c}\}$,$j\in H$,$i\in Z$,$t\in T$ 表示 m_{ij} 在时隙 t 的任务卸载策略,其中 $I_{ij,\mathrm{m}}(t)=1$,$I_{ij,\mathrm{c}}(t)=1$ 分别表示在时隙 t 计算任务卸载给MEC服务器或在中心云服务器处理,所有MD的任务卸载策略集合表示为 $I=\{I_{ij,\mathrm{o}}(t)\}_{\mathrm{o}\in\{\mathrm{m},\mathrm{c}\},j\in H,i\in Z}$。在时隙 t 卸载策略必须满足以下约束:

$$I_{ij,\mathrm{m}}(t)+I_{ij,\mathrm{c}}(t)\leqslant 1,j\in H,i\in Z,t\in T \quad (4)$$

一般地,计算结果数据量远小于上传数据量,数据下行传输速率远高于上行传输速率,因此,忽略了结果返回的延迟[5]。

1.2 服务器通信模型

在时隙 t,若 $I_{ij,\mathrm{m}}(t)=1$,MD需要将计算任务通过无线AP传输到MEC服务器上执行。由香农公式可得数据传输速率,表示为

$$R_{ij}^{\mathrm{m}}(t)=B_{\mathrm{m}}\mathrm{lb}\left(1+\frac{p_{ij}^{\mathrm{m}}g_{ij}^{\mathrm{m}}}{\sigma^2+\sum_{x\neq i}\sum_{y\neq j}p_{xy}^{\mathrm{m}}g_{xy}^{\mathrm{m}}}\right)\tag{5}$$

其中：B_{m} 表示 MEC 服务器带宽，p_{ij}^{m} 表示第 i 个 MD 传输到 MEC 服务器的传输功率，g_{ij}^{m} 表示第 i 个 MD 与 MEC 服务器之间的信道增益，σ^2 为噪声功率。同理，$I_{ij,\mathrm{c}}(t)=1$ 时，中心云的数据传输速率为

$$R_{ij}^{\mathrm{c}}(t)=B_{\mathrm{c}}\mathrm{lb}\left(1+\frac{p_{ij}^{\mathrm{c}}g_{ij}^{\mathrm{c}}}{\sigma^2+\sum_{x\neq i}\sum_{y\neq j}p_{xy}^{\mathrm{c}}g_{xy}^{\mathrm{c}}}\right)\tag{6}$$

任务从用户终端到服务器的传输时延和任务传输成本分别表示为

$$T_{ij}^{\mathrm{tr}}(t)=\frac{b_{ij}^{\mathrm{o}}(t)}{R_{ij}^{\mathrm{m}}(t)}(I_{ij,\mathrm{m}}=1)+\frac{b_{ij}^{\mathrm{o}}(t)}{R_{ij}^{\mathrm{c}}(t)}(I_{ij,\mathrm{c}}=1)\tag{7}$$

$$C_{ij}^{\mathrm{tr}}(t)=\eta_{\mathrm{m}}b_{ij}^{\mathrm{o}}(t)(I_{ij,\mathrm{m}}=1)+\eta_{\mathrm{c}}b_{ij}^{\mathrm{o}}(t)(I_{ij,\mathrm{c}}=1)\tag{8}$$

其中 η_{m} 和 η_{c} 分别为 m_{ij} 传输到 MEC 服务器和中心云服务器的单位时间成本。

1.3　服务器计算时延模型

在时隙 t，当 MD 需要卸载任务给服务器时，服务器会分配一定的计算资源来处理卸载任务。假设 $F=\{f_i^{\mathrm{o}}(t)\}$，$\mathrm{o}\in\{\mathrm{m},\mathrm{c}\}$ 表示在时隙 t 给所有 MD 的资源分配集合，$f_i^{\mathrm{c}}(t)$ 表示中心云服务器分配给 m_{ij} 的计算资源，并有 $f_{\mathrm{c}}^{\min}\leqslant f_i^{\mathrm{c}}(t)\leqslant f_{\mathrm{c}}^{\max}$。同理，$f_i^{\mathrm{m}}(t)$ 表示 MEC 服务器分配给 m_{ij} 的计算资源，有 $f_{\mathrm{m}}^{\min}\leqslant f_i^{\mathrm{m}}(t)\leqslant f_{\mathrm{m}}^{\max}$。假设 γ_{ij} 是计算密度（以周期/位为单位），可得卸载任务在服务器的计算时延为

$$T_{ij}^{\mathrm{cp}}(t)=\frac{\gamma_{ij}b_{ij}^{\mathrm{o}}(t)}{f_i^{\mathrm{m}}(t)}(I_{ij,\mathrm{m}}=1)+\frac{\gamma_{ij}b_{ij}^{\mathrm{o}}(t)}{f_i^{\mathrm{c}}(t)}(I_{ij,\mathrm{c}}=1)\tag{9}$$

故 m_{ij} 在时隙 t 内的计算成本为

$$C_{ij}^{\mathrm{cp}}(t)=\beta_{\mathrm{m}}T_{ij}^{\mathrm{cp}}(t)(I_{ij,\mathrm{m}}=1)+\beta_{\mathrm{c}}T_{ij}^{\mathrm{cp}}(t)(I_{ij,\mathrm{c}}=1)\tag{10}$$

其中 β_{m} 和 β_{c} 分别为 MEC 服务器和中心云服务器的单位计算成本。

在服务器处理卸载任务时，产生的能耗包括数据通信能耗 $P_{ij}^{\mathrm{tr}}(t)$ 和服务器计算任务产生的能耗 $P_{ij}^{\mathrm{cp}}(t)$，分别为

$$P_{ij}^{\mathrm{tr}}(t)=p_{ij}^{\mathrm{m}}T_{ij}^{\mathrm{tr}}(t)(I_{ij,\mathrm{m}}=1)+p_{ij}^{\mathrm{c}}T_{ij}^{\mathrm{tr}}(t)(I_{ij,\mathrm{c}}=1)\tag{11}$$

$$P_{ij}^{\mathrm{cp}}(t)=\chi_{\mathrm{m},j}f_i^{\mathrm{m}}(t)^2T_{ij}^{\mathrm{cp}}(t)(I_{ij,\mathrm{m}}=1)+\chi_{\mathrm{c}}\,f_i^{\mathrm{c}}(t)^2T_{ij}^{\mathrm{cp}}(t)(I_{ij,\mathrm{c}}=1)\tag{12}$$

其中 $\chi_{\mathrm{m},j}$ 和 χ_{c} 分别为 MEC 服务器 j 和中心云服务器的 CPU 能耗加权参数。因此，能耗成本表示为

$$C_{ij}^{\mathrm{en}}(t)=\delta_{ij}^{\mathrm{tr}}P_{ij}^{\mathrm{tr}}(t)+\delta_{ij}^{\mathrm{cp}}P_{ij}^{\mathrm{cp}}(t)\tag{13}$$

其中 $\delta_{ij}^{\mathrm{tr}}$ 和 $\delta_{ij}^{\mathrm{cp}}$ 分别为通信与计算的单位能耗成本。

1.4　卸载效用模型

MD 将任务卸载到服务器处理，目的是为了节省自身计算资源，并从中获得相应效益。为了评估在时隙 t 下 m_{ij} 获得的本地和卸载效益，采用了在移动计算领域广泛使用的对数效用函数[6]，即

$$U_{ij}(t)=\zeta_i\mathrm{lb}(1+b_{ij}(t)),i\in Z,\ j\in H,\ t\in T\tag{14}$$

其中：$b_{ij}(t)=b_{ij}^{\mathrm{o}}(t)+b_{ij}^{\mathrm{l}}(t)$，$\zeta_i$ 为 m_{ij} 的加权参数。

根据以上模型，m_{ij} 在时隙 t 的任务卸载收益为

$$r_{ij}(t)=U_{ij}(t)-C_{ij}^{\mathrm{tr}}(t)-C_{ij}^{\mathrm{cp}}(t)-C_{ij}^{\mathrm{en}}(t)\tag{15}$$

因此，优化问题可以表示为

$$\mathrm{P1}:\max_{(I,\Lambda,F)}\overline{R}=\limsup_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}E\left\{\sum_{j=0}^{H}\sum_{i=1}^{Z}r_{ij}(t)\right\}\tag{16}$$

$$\text{s.t.}\quad\left.\begin{aligned}
&\mathrm{C1}:\ T_{ij}(t)\leqslant\tau_{\mathrm{d}},i\in Z,j\in H,t\in T\\
&\mathrm{C2}:\ \sum_{\forall i\in Z}I_{ij,\mathrm{o}}(t)\leqslant 1,I_{ij,\mathrm{o}}(t)\in\{0,1\},\\
&\qquad\mathrm{o}\in\{\mathrm{m},\mathrm{c}\},j\in H\\
&\mathrm{C3}:\ f_{\mathrm{o}}^{\min}\leqslant f_i^{\mathrm{o}}(t)\leqslant f_{\mathrm{o}}^{\max},\ \mathrm{o}\in\{\mathrm{m},\mathrm{c}\},\ i\in Z\\
&\mathrm{C4}:\ \sum_{\forall i\in Z}f_i^{\mathrm{o}}(t)\leqslant f_{\mathrm{o}}^{\max},\mathrm{o}\in\{\mathrm{m},\mathrm{c}\},i\in Z\\
&\mathrm{C5}:\ b_{ij}^{\mathrm{o},\min}\leqslant b_{ij}^{\mathrm{o}}(t)\leqslant b_{ij}^{\mathrm{o},\max},\ i\in Z,j\in H\\
&\mathrm{C6}:\ b_{ij}^{\mathrm{l},\min}\leqslant b_{ij}^{\mathrm{l}}(t)\leqslant b_{ij}^{\mathrm{l},\max},\ i\in Z,j\in H\\
&\mathrm{C7}:\ 0\leqslant a_{ij}(t)\leqslant A_{ij}(t),\ i\in Z,j\in H\\
&\mathrm{C8}:\ \overline{Q}_{ij}^{\mathrm{o}}=\limsup_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{Z}E\{Q_{ij}^{\mathrm{o}}(t)\}<\infty
\end{aligned}\right\}\tag{17}$$

其中：C1 为 MD 卸载延迟不能超过最大计算截止时间；C2 为 MD 只将任务卸载到 MEC 服务器或中心云服务器；C3 为 MD 的计算资源不能超过自身计算资源总量的最大值；C4 为服务器分配总计算资源不能超过 CPU 的最大频率 $f_{\mathrm{o}}^{\max}$；C5 为 MD 卸载到服务器的任务量小于当前时隙任务积压量；C6 为本地处理任务量小于当前队列任务积压量；C7 为进入本地的任务量不能大于到达 MD 的任务量；C8 为确保网络稳定性。

2　任务卸载与资源分配联合优化算法

2.1　基于 Lyapunov 优化的模型分析

在问题 P1 中，MD 的任务到达是随机的，具有时间耦合约束的随机规划问题很难解决，因此引入

李雅普诺夫优化方法,将任务队列的时间耦合解耦,只利用MD的当前状态进行下一步分析,最终得出权衡卸载收益与时延的卸载决策。根据李雅普诺夫优化理论,定义李雅普诺夫函数和李雅普诺夫漂移分别为

$$L(Q(t)) \triangleq \frac{1}{2}\{(Q_{ij}^{\mathrm{l}}(t))^2 + (Q_{ij}^{\mathrm{o}}(t))^2\} \quad (18)$$

$$\Delta(Q(t)) \triangleq E\{L(Q(t+1)) - L(Q(t)) \mid Q_{ij}^{\mathrm{l}}(t), Q_{ij}^{\mathrm{o}}(t)\} \quad (19)$$

最优决策目标是最小化漂移加惩罚函数的上界,即$\Delta_V(Q(t)) = \Delta(Q(t)) - VE\{r(t) \mid Q(t)\}$,其中$r(t)$为总收益,$V$为非负可控参数。

定理1

$$\Delta(Q(t)) - VE\{r(t) \mid Q(t)\} \leqslant \Phi - VE\{r(t) \mid Q(t)\} - E\left\{\sum_{\forall i,j} Q_{ij}^{\mathrm{l}}(t) b_{ij}^{\mathrm{l}}(t) \mid Q(t)\right\} + E\left\{\sum_{\forall i,j} Q_{ij}^{\mathrm{l}}(t) a_{ij}(t) \mid Q(t)\right\} - E\left\{\sum_{\forall i,j} Q_{ij}^{\mathrm{o}}(t) b_{ij}^{\mathrm{o}}(t) \mid Q(t)\right\} + E\left\{\sum_{\forall i,j} Q_{ij}^{\mathrm{o}}(t) c_{ij}(t) \mid Q(t)\right\} \quad (20)$$

其中$\Phi = \frac{1}{2}\sum_{\forall i,j}(2(A_{ij}^{\max})^2 + (b_{ij}^{\mathrm{l},\max})^2 + (b_{ij}^{\mathrm{o},\max})^2)$。

从问题P1可知,最小化漂移加惩罚函数的上界等于最小化不等式(20)的右侧部分。根据定理1,结合机会主义最小化期望准则,将问题P1转化为问题P2。

$$\mathrm{P2}: \max_{(I,A,F)} X(t) = \sum_{j=0}^{H}\sum_{i=1}^{Z}(Q_{ij}^{\mathrm{l}}(t) b_{ij}^{\mathrm{l}}(t) - Q_{ij}^{\mathrm{l}}(t) a_{ij}(t)) + \sum_{j=0}^{H}\sum_{i=1}^{Z}(Q_{ij}^{\mathrm{o}}(t) b_{ij}^{\mathrm{o}}(t) + Q_{ij}^{\mathrm{o}}(t) a_{ij}(t)) + V\sum_{j=0}^{H}\sum_{i=1}^{Z} r_{ij}(t) \quad (21)$$

根据文献[7]可得,平均卸载收益边界和平均队列积压边界分别满足以下不等式:

$$-\overline{R}(t) \triangleq -\limsup_{T\to\infty}\frac{1}{T}\sum_{T=0}^{T-1} E\left\{\sum_{j=0}^{H}\sum_{i=1}^{Z} r_{ij}(t)\right\} \leqslant -R^* + \frac{\Phi}{V} \quad (22)$$

$$\overline{Q} \triangleq \limsup_{T\to\infty}\frac{1}{T}\sum_{T=0}^{T-1}\sum_{i=1}^{Z} E\{Q_{ij}^{\mathrm{l}}(t) + Q_{ij}^{\mathrm{o}}(t)\} \leqslant \frac{\Phi - VR^*}{\varepsilon} \quad (23)$$

其中:$-\overline{R}(t)$为系统平均任务卸载收益,R^*为最佳平均卸载收益,$\overline{Q}$为系统平均队列积压,ε为速率向量λ_j与容量区域之间的距离。式(22)和式(23)说明:在任务卸载收益和队列积压之间是一种$[O(1/V), O(V)]$形式的折中。

2.2 最优策略分析

根据问题P2,应求得每个时间段内问题的最优解,它包括最优任务缓存策略Λ^*、最优的计算资源决策F^*以及最优的服务器选择策略I^*。

2.2.1 任务缓存策略分析

将问题P2对$a_{ij}(t)$求偏导后进行分析,可得

$$a_{ij}(t) = \begin{cases} 0, & Q_{ij}^{\mathrm{l}}(t) > Q_{ij}^{\mathrm{o}}(t) \\ A_{ij}(t), & Q_{ij}^{\mathrm{l}}(t) \leqslant Q_{ij}^{\mathrm{o}}(t) \end{cases} \quad (24)$$

$$c_{ij}(t) = \begin{cases} 0, & Q_{ij}^{\mathrm{l}}(t) \leqslant Q_{ij}^{\mathrm{o}}(t) \\ A_{ij}(t), & Q_{ij}^{\mathrm{l}}(t) > Q_{ij}^{\mathrm{o}}(t) \end{cases} \quad (25)$$

2.2.2 计算资源策略分析

根据上述分析结果,把问题P2转成问题P3,得

$$\mathrm{P3}: \max_{(I,F)} Y(t) = \sum_{j=0}^{H}\sum_{i=1}^{Z}(Q_{ij}^{\mathrm{l}}(t) b_{ij}^{\mathrm{l}}(t) + Q_{ij}^{\mathrm{o}}(t) b_{ij}^{\mathrm{o}}(t)) + V\sum_{j=0}^{H}\sum_{i=1}^{Z} r_{ij}(t) \quad (26)$$

当$I_{ij,\mathrm{m}}(t) = 1$时,根据式(26),将$Y(t)$对$b_{ij}^{\mathrm{l}}(t)$求偏导,可知$Y(t)$是关于$b_{ij}^{\mathrm{l}}(t)$的凸函数。根据二阶偏导和KKT(Karush-Kuhn-Tucker)条件,令$\frac{\partial Y(t)^*}{\partial (b_{ij}^{\mathrm{l}}(t))^*} = 0$,可以得到最优的本地处理策略,即

$$(b_{ij}^{\mathrm{l}}(t))^* = -\left(1 + b_{ij}^{\mathrm{o}}(t) + \frac{V\zeta_i}{Q_{ij}^{\mathrm{l}}(t)\ln 2}\right) \quad (27)$$

同理可得最优的MEC服务器卸载处理策略为

$$(b_{ij}^{\mathrm{o}}(t))^* = \frac{V\zeta_i}{\Theta_1 \ln 2} - 1 - b_{ij}^{\mathrm{l}}(t) \quad (28)$$

其中Θ_1的大小为

$$V\left(\delta_{ij}^{\mathrm{cp}}\chi_{\mathrm{m},j} f_i^{\mathrm{m}}(t)\gamma_{ij} + \eta_{\mathrm{m}} + \frac{\delta_{ij}^{\mathrm{tr}} p_{ij}^{\mathrm{m}}}{R_{ij}^{\mathrm{m}}(t)} + \frac{\beta_{\mathrm{m}}\gamma_{ij}}{f_i^{\mathrm{m}}(t)}\right) - Q_{ij}^{\mathrm{o}}(t)$$

当$I_{ij,\mathrm{c}}(t) = 1$时,最优的本地处理策略如式(27),同理可得,最优的中心云服务器卸载处理策略为

$$(b_{ij}^{\mathrm{o}}(t))^* = \frac{V\zeta_i}{\Theta_2 \ln 2} - 1 - b_{ij}^{\mathrm{l}}(t) \quad (29)$$

其中Θ_2的大小为

$$V\left(\delta_{ij}^{\mathrm{cp}}\chi_{\mathrm{c}} f_i^{\mathrm{c}}(t)\gamma_{ij} + \eta_{\mathrm{c}} + \frac{\delta_{ij}^{\mathrm{tr}} p_{ij}^{\mathrm{c}}}{R_{ij}^{\mathrm{c}}(t)} + \frac{\beta_{\mathrm{c}}\gamma_{ij}}{f_i^{\mathrm{c}}(t)}\right) - Q_{ij}^{\mathrm{o}}(t)$$

根据 $b(t)=\dfrac{f(t)\tau}{L}$ 可得 $f_{ij}^{\mathrm{l}}(t)$，$f_i^{\mathrm{m}}(t)$，$f_i^{\mathrm{c}}(t)$，L 为执行每比特计算任务所需要的 CPU 周期。

2.2.3　服务器选择策略分析

从问题 P3 的分析来看，MD 在每个时隙的卸载策略 $\boldsymbol{I}_{ij,\mathrm{o}}(t)$（$\mathrm{o}\in\{\mathrm{m},\mathrm{c}\}$）可看成整数向量，而计算资源分配向量 $\boldsymbol{F}$ 是连续的。因此，问题 P3 是一个 MINLP 问题[8]。由于问题 P3 中整数向量与连续向量耦合，很难求解。为求得最优解，采用文献[9]中的方法建立典型搜索树，如图 2 所示。从根节点出发，向它的左右 2 个叶节点进行搜索，根据 MD 的卸载任务量 $b_{ij}^{\mathrm{o}}(t)$ 和计算资源在每一条路径下分配向量 $\boldsymbol{F}$，以选择最佳卸载策略。

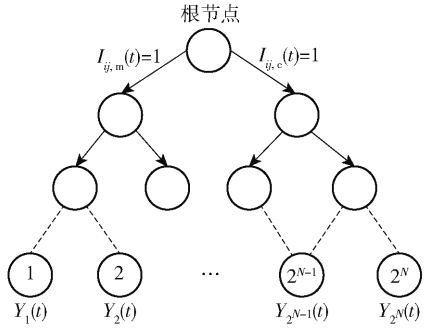

图 2　具有 2^N 条路径的搜索树

从搜索树可得，搜索路径会随 MD 的数量呈指数增加，因此具有很高的复杂度。为了降低系统复杂度和不必要的通信开销，设计了一个基于优先级的任务卸载准则，对搜索树进行快速地分支定界，可大幅度降低系统复杂度。该准则包含以下两方面内容。

1）若 $t_{ij}^{\min}>\tau_{\mathrm{d}}$，表明 m_{ij} 若将计算任务卸载到 MEC 服务器所消耗的传输与计算时间之和大于计算任务的最大截止时间 τ_{d}，则计算任务将被卸载到中心云服务器，即 m_{ij} 属于 G_{c} 组，且 $t_{ij}^{\min}$ 可计算为

$$t_{ij}^{\min}=\frac{b_{ij}^{\mathrm{o},\min}}{R_{ij}^{\mathrm{m}}(t)}+\frac{\gamma_{ij}b_{ij}^{\mathrm{o},\min}}{f_{\mathrm{m}}^{\max}} \tag{30}$$

2）若 $t_{ij}^{\max}\leqslant\tau_{\mathrm{d}}$，表明 m_{ij} 若将计算任务卸载到 MEC 服务器所消耗的传输与计算时间之和小于等于计算任务的最大截止时间 τ_{d}，则计算任务将被卸载到 MEC 服务器，则 m_{ij} 属于 G_{m} 组，$t_{ij}^{\max}$ 可计算为

$$t_{ij}^{\max}=\frac{b_{ij}^{\mathrm{o},\max}}{R_{ij}^{\mathrm{m}}(t)}+\frac{\gamma_{ij}b_{ij}^{\mathrm{o},\max}}{f_{\mathrm{m}}^{\min}} \tag{31}$$

G_{m} 和 G_{c} 组的 MD 分别将任务卸载到 MEC 服务器和中心云服务器，但 G_{o} 组的第 i 个 MD 是未知的，即计算任务可以卸载到 MEC 服务器或卸载到中心云服务器。将 MD 按上述方法进行分类分组后，减少了搜索树的搜索空间，能较大地提升系统的搜索效率。

假设基于 TOST-TOC 算法的搜索树大小为 $|\boldsymbol{\Omega}|$，对于每一条路径，用凸优化方法求解最优的任务卸载量。通过从根节点到叶节点的所有路径，得到优化目标集合 $\{Y_1(t),Y_2(t),\cdots,Y_{\Omega}(t)\}$，每个优化目标值的解为

$$(\phi_1(t),\phi_2(t),\cdots,\phi_{\Omega}(t)\,|\,\phi(t)\in\{\boldsymbol{I},\boldsymbol{F}\})$$

然后，以最大化目标值作为最优解，即

$$\phi^*(t)=\arg\max\{Y_1(t),Y_2(t),\cdots,Y_{\Omega}(t)\,|\,\phi_1(t),\phi_2(t),\cdots,\phi_{\Omega}(t)\} \tag{32}$$

3　仿真与分析

利用 Matlab 仿真软件对笔者所提的 TOST-TOC 决策进行了仿真实验，与 LARAC（lagrangian relaxation based aggregated cost）[10] 算法和 SMSEF（select maximum saved energy fitst）[11] 算法做了对比，以验证笔者所提算法性能。LARAC 算法中 MD 的计算任务全部卸载到 MEC 服务器或者中心云服务器，不考虑本地；SMSEF 算法中队列中的任务完全卸载到中心云。

3.1　仿真参数设置

MEC 服务器和中心云服务器的信道上传速率分别为 5 MHz 和 11 MHz，MEC 服务器和中心云服务器的传输功率分别为 $p_{ij}^{\mathrm{m}}=200$ mW 和 $p_{ij}^{\mathrm{c}}=500$ mW，$\eta_{\mathrm{m}}=1.5$，$\eta_{\mathrm{c}}=3$，$\zeta_i=5$，$\tau=1$ s，$\sigma^2=-75$ dBm。同时，将服务器中 MD 的任务到达设置为独立同分布的泊松过程，$\lambda_i=10$ Mbit/s。任务所需的 CPU 周期为[1 000，1 500] cycle/bit，MEC 服务器和中心云服务器的计算能力分别为 $f_{ij,\mathrm{m}}\in[2,10]$ GHz 和 $f_{ij,\mathrm{c}}\in[4,20]$ GHz。

3.2　性能仿真

不同算法下平均队列积压、平均卸载收益以及平均卸载成本随仿真时间的变化如图 3 ~ 图 5 所示。从图 3 可见，TOST-TOC 算法与 LARAC 算法的平均队列积压都随着时间增加而趋于稳定，而 SMSEF 算法在每个时隙内卸载全部任务到中心云进行处理，因此，队列积压一直为 0，不随时间变化。

不同算法的卸载收益对比如图 4 所示。可见，TOST-TOC 算法的卸载收益要优于 LARAC 算法和 SMSEF 算法。从图 5 可见，TOST-TOC 算法与 LARAC

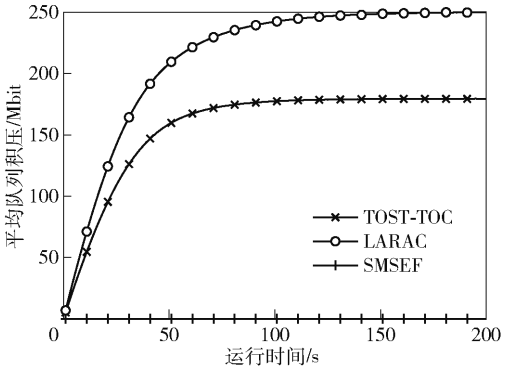

图3 不同算法下随时间变化的平均队列积压对比

算法的卸载成本都随着时间的增加而趋于稳定,但TOST-TOC算法的卸载成本低于LARAC算法。

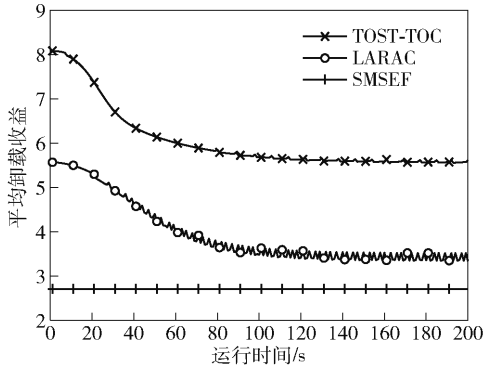

图4 不同算法下随时间变化的平均卸载收益对比

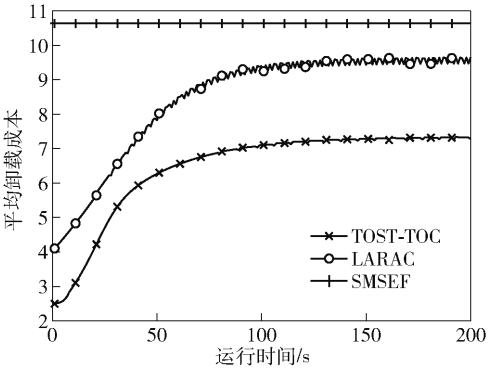

图5 不同算法下随时间变化的平均卸载成本对比

从图3~图5可见,随着仿真时间的增加,3种算法下的队列积压、卸载收益和卸载成本都能趋于稳定,说明卸载系统是稳定的。此外,从队列积压、卸载收益与卸载成本3种性能指标来看,随着运行时间的增加,TOST-TOC算法都能较快趋于稳定,证明TOST-TOC算法能在短时间内收敛,而SMSEF算法将队列任务全部卸载到中心云。因此,3种性能指标一直处于稳定状态,证实了TOST-TOC算法的性能优于其他2种算法。

TOST-TOC算法与其他算法的性能对比如图6所示。其中TOST为普通的搜索树算法,TOST-TOC为降低复杂度卸载准则的搜索树算法。由图6可知,移动设备数量相同时,TOST算法比TOST-TOC算法运行的时间更长,而TOST-TOC算法将移动设备分为3组,大大减少了搜索空间,因此运行时间更短。

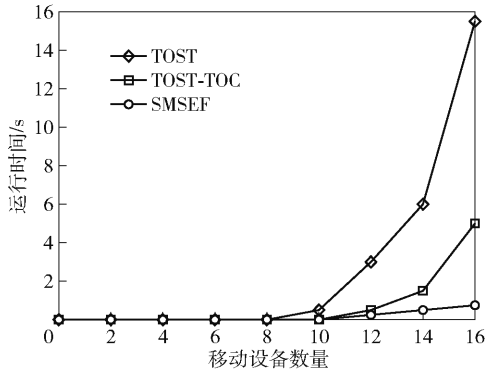

图6 不同卸载算法下的运行时间对比

4 结束语

基于移动边缘网络场景下的任务卸载问题,根据李雅普诺夫优化理论,证明了任务卸载收益与时延之间存在[$O(1/V)$,$O(V)$]的折中关系。为了求得在权衡卸载收益与时延时最佳的任务卸载策略,提出了基于搜索树的启发式任务卸载算法,设计了一种卸载优先级准则,最后通过实验仿真验证了笔者所提算法的有效性和合理性。

参考文献:

[1] XIA S C, WEN X X, YAO Z X, et al. Dynamic task offloading and resource allocation for heterogeneous MEC-enable IoT[C]//2020 IEEE/CIC International Conference on Communications in China (ICCC). Piscataway, NJ: IEEE Press, 2020: 847-852.

[2] HU H, WANG Q, HU R Q, et al. Mobility-aware offloading and resource allocation in an MEC-enabled IoT network with energy harvesting[J]. IEEE Internet of Things Journal, 2021, 8(24): 17541-17556.

[3] 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[J]. IEEE Transactions on Wireless Communications, 2021, 20(10): 6743-6757.

[4] MAO Y Y, YOU C S, ZHANG J, et al. A survey on

mobile edge computing: the communication perspective[J]. IEEE Communications Surveys and Tutorials, 2017, 19(4): 2322-2358.

[5] YOU C S, HUANG K, HYUKJIN C, et al. Energy-efficient resource allocation for mobile-edge computation offloading[J]. IEEE Transactions on Wireless Communications, 2017, 16(3): 1397-1411.

[6] LI Y, XIA S C, ZHENG M Y, et al. Lyapunov optimization based trade-off policy for mobile cloud offloading in heterogeneous wireless networks[J/OL]. IEEE Transactions on Cloud Computing, 2019: 1-12 [2019-8-30]. https://ieeexplore.ieee.org/document/8821330.

[7] NEELY M J. Stochastic network optimization with application to communication and queueing system[J]. Synthesis Lectures on Communication Networks, 2010, 3(1): 1-211.

[8] YVES P, LAURENCE A. Production planning by mixed integer programming[J/OL]. New York: Springer, 2016: 163-175 [2016-01-01]. https://link.springer.com/book/10.1007/0-387-33477-7.

[9] DEFOURNY B, ERNST D, WEHENKEL L. Multistage stochastic programming: a scenario tree based approach to planning under uncertainty[J]. IGI Global, 2012,35(4): 97-143.

[10] ZHANG W W, WEN Y G, WU D O. Collaborative task execution in mobile cloud computing under a stochastic wireless channel[J]. IEEE Transactions on Wireless Communications, 2015, 14(1): 81-93.

[11] WEI F, CHEN S X, ZOU W X. A greedy algorithm for task offloading in mobile edge computing system[J]. China Communications, 2018, 15(11): 149-157.