文章编号:2095-6134(2020)05-0699-09

非稳态雾赋能网络中的在线任务卸载方法*

朱兆伟[1,2,3†],刘婷[3],钱骅[4],罗喜良[3]

(1 中国科学院上海微系统与信息技术研究所, 上海 200050; 2 中国科学院大学, 北京 100049;
3 上海科技大学, 上海 201210; 4 中国科学院上海高等研究院, 上海 201210)

(2019 年 2 月 15 日收稿; 2019 年 3 月 28 日收修改稿)

Zhu Z W, Liu T, Qian H, et al. Online task offloading in non-stationary fog-enabled networks[J]. Journal of University of Chinese Academy of Sciences, 2020,37(5):699-707.

摘 要 为充分发掘分布在不同位置上的雾节点的计算资源,任务卸载被寄予众望。在雾计算场景下,以尽可能减少任务卸载的长期成本为目标,试图寻找一个高效的在线任务卸载方法。为此,这一问题被建模成一个随机规划问题,该问题中系统参数所对应的随机变量的期望会在未知时刻突变,系统参数相关信息只能在任务完成后的反馈中获得。基于非稳态多臂老虎机模型,提出一个高效的算法来解决这一具有挑战性的随机优化问题,给出理论分析证明该算法的渐进最优性。数值实验证明了该算法的优越性。

关键词 在线学习;雾计算;任务卸载;随机优化;多臂老虎机

中图分类号:TP393 **文献标志码**:A **doi**:10.7523/j.issn.2095-6134.2020.05.015

Online task offloading in non-stationary fog-enabled networks

ZHU Zhaowei[1,2,3], LIU Ting[3], QIAN Hua[4], LUO Xiliang[3]

(1 *Shanghai Institute of Microsystem & Information Technology, Chinese Academy of Sciences, Shanghai* 200050, *China*;
2 *University of Chinese Academy of Sciences, Beijing* 100049, *China*; 3 *ShanghaiTech University, Shanghai* 201210, *China*;
4 *Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai* 201210, *China*)

Abstract To fully exploit the computational resources in different fog nodes, task offloading is emerging. In this work, under the fog computing scenario, an efficient online task offloading strategy is investigated to minimize the long-term cost of task offloading. To achieve this goal, the problem is modeled as a stochastic optimization problem. Moreover, the system parameters are characterized by random variables, and their expectations may change abruptly at unknown time slot. Besides, the information about the system parameters is only available through the feedbacks after the task finishes. Using the non-stationary multi-armed bandit framework, we propose an efficient algorithm to handle this challenging stochastic programming. Furthermore, theoretical analyses are presented to prove the asymptotic optimality of the proposed algorithm. Numerical results reveal the advantages of this algorithm.

Keywords online learning; fog computing; task offloading; stochastic optimization; multi-armed bandit

* 国家自然科学基金(61671436)资助

† 通信作者,E-mail:zhuzhw@shanghaitech.edu.cn

随着物联网的快速发展,各类移动智能设备需要处理的任务量不断提高。比如,应用增强现实技术的在线交互游戏设备需要大量计算和通信资源。因此,传统的个人电脑、智能手机等移动设备和物联网设备在电池和计算能力方面面临巨大的挑战。一个经典的解决方案是将这些任务卸载到能源、存储和计算资源丰富的云端服务器[1],但是远距离云端传输将会带来额外的通信时间。为了满足低时延的服务要求,研究人员提出利用雾计算节点(如具有闲置可用资源的移动、物联网设备)数量庞大、无处不在的天然优势,将计算、存储、控制和通信服务分布在云到雾的连续体中。因此,为了更好地利用周围的雾节点,急需一个高效的算法,来决定在雾计算网络中哪些计算任务需要卸载以及应卸载到哪个节点。

一般来说,把高复杂度的计算任务卸载到其他的节点上能够有效地节约本地节点的计算资源与能量资源。在一些文献中,任务卸载被建模为确定性优化问题,例如,Dinh 等[2]研究的能量和延迟的联合最小化,以及 You 等[3]研究的延迟约束下的能耗最小化。然而,一个实用的任务卸载策略需要依赖于用户和服务器的实时状态,例如,计算队列的长度等信息。从这个方面来说,因计算队列长度的不确定性,任务卸载是典型的随机规划问题,传统的基于确定性参数的优化方法并不适用。为了解决这个问题,在文献[4-7]中,研究人员调用李雅普诺夫(Lyapunov)优化方法,将具有挑战性的随机规划问题转换为顺序决策问题,其中包括每个时隙中的一系列确定性问题。此外,Chen[8]提出一种基于博弈论的分布式解决方案,每个用户可以自主地进行卸载决策。

上述文献中提出的任务卸载方案都假定可以获得关于系统参数的全部信息。但是,在实际系统中,存在参数在用户处未知或部分已知的情况。例如,一些特定的值可能只能作为后验信息,当特定节点被访问时才能获得。Chen 和 Giannakis[9]将每个任务的通信延迟和计算延迟视为后验信息。Tekin 和 Van Der Schaar[10]假设每个用户的移动性是不可预测的。在实际系统中,可用资源通常是有限的,从而导致每次决策可访问的节点数量非常有限。此时,若以最小化长期开销为目标,所做出的决策就必须平衡“探索”与“开发”之间的权重。一方面,为了减小眼前的开销,决策应偏向尽可能“开发”经验最佳节点;另一方面,从长远角度考虑,用户需要“探索”其他节点以找到潜在的性能更优的节点。为了平衡这种关系,一种流行的方法是将“探索”与“开发”困境建模为多臂老虎机(MAB,multi-armed bandit)[11-13]问题。这一理论模型在统计学中受到广泛关注[13]。

在雾计算网络任务卸载的研究中,很少有先前的工作研究这种探索与开发权衡的关系,本文将详细探讨这个问题。首先,假设每个任务的准确的处理时延在任务处理之前是未知的。基于此,尝试基于有限的反馈提出一种高效的任务卸载算法,从而最小化用户长期时延。因此,引入一个非稳态多臂老虎机模型以捕捉随机且未知的时延变动。相比于以往如文献[2-7]中的确定性模型,该随机模型更加符合实际模型。之后,基于置信上界(UCB, upper-confidence bound)策略提出一种高效的任务卸载算法。由于该算法不是直接的 UCB 策略的应用,因此传统的理论分析无法直接适用。针对这一改进版的 UCB 算法,将从理论层面给出算法的性能保证。

1 系统模型

1.1 网络模型

考虑一个雾赋能的网络(参见图 1),其中雾节点按照其功能可分为任务节点和辅助节点两类。每个雾节点都有可能产生计算任务,也可以与附近的节点通信。假定每个节点未完成的任务被缓存在各自节点的先入先出(FIFO, first-input-first-output)队列中。由于单一节点内的计算和存储资源有限,在本地处理的任务通常会经历较高延迟,这会降低服务质量(QoS,quality of service)和体验质量(QoE, quality of experience)。为了实

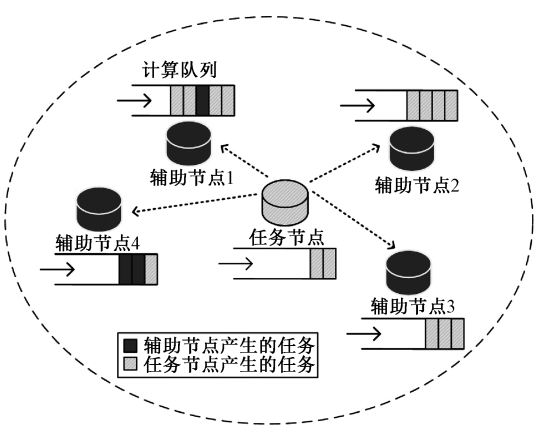


图 1 雾赋能网络的拓扑结构图

Fig. 1 Topography of a fog-enabled network

现低延迟处理,一个任务节点可以将其一些计算任务卸载到附近的辅助节点。这些辅助节点通常拥有更多的计算和存储资源,并且可以按需部署以帮助其他任务节点。在诸如在线游戏等典型应用中,任务通常是周期性地生成的,并且不能任意拆分。因此,假设每个节点在每个时隙 t 的开始生成一个任务 t, 该任务可以作为一个整体由本地节点计算或者分配给一个相邻的辅助节点。

本文的目标是最小化一个特定任务节点的长期时延。特别地,考虑如下所示集合$\mathcal{I}$中的 K 个雾节点:

$$\mathcal{I} := \{\underbrace{1,2,\cdots,K-1}_{\text{Helper nodes}}, \underbrace{K}_{\text{Task node}}\}. \tag{1}$$

在本文中,假设任务节点不能将任务卸载到正在与其他节点通信的辅助节点。还假设每个任务都是独立生成的,且任务节点之间不合作。

用 $T(i)$ 表示传输每一比特信息到节点 i 所需要的时间,它是一个依赖于距离的值,并且可以在传输前进行测量。在这里,假设不同任务节点占用预先分配的正交时间或频谱资源与辅助节点通信,如 TDMA 或 FDMA。任务 t 的数据长度用 L_t 表示,假设任务大小传输延迟 $L_tT(i)$ 不超过一个时隙。在本地处理的任务传输延迟为零,即 $L_tT(K)=0$。

用 $Q_t(i)$ 表示节点 i 在时隙 t 开始时的任务队列长度。同时, $W_t(i)$ 表示节点 i 处理每一比特队列中的任务所需的时间, $P_t(i)$ 表示节点 i 处理任务 t 中每一比特所需的时间。在本文中,变量 $W_t(i)$ 和 $P_t(i)$ 被定义为随机变量,它们的数学期望分别为

$$\mu_t^W(i) := \mathbb{E}[W_t(i)], \mu_t^P(i) := \mathbb{E}[P_t(i)]. \tag{2}$$

假设每个任务的总时延主要由上述时延构成,即传输时延 $L_tT(i)$、队列中的等待时延 $Q_t(i)W_t(i)$ 以及处理时延 $L_tP_t(i)$。传输计算结果等反馈信息所需要的时延在本文中不考虑。根据上述定义,将任务 t 分配给节点 i 处理所需的时延可以定义为

$$U_t(i) := L_tT(i) + Q_t(i)W_t(i) + L_tP_t(i). \tag{3}$$

在进一步分析问题之前,给出下列假设:

· 假设1:时延 $U_t(i)$ 在任务 t 处理完之前是未知的;

· 假设2:队列长度 $Q_t(i)$ 在每个时隙 t 开始时由节点 i 广播至各个雾计算节点;

· 假设3:等待时延和处理时延,即 $L_tT(i)$ 和 $Q_t(i)W_t(i)$, 遵循未知随机分布。相应的数学期望,即 $\mu_t^W(i)$ 和 $\mu_t^P(i)$, 会在未知时间点突变。这些时间点称作断点。

与文献[2-7]中基于已知系统参数的模型的任务卸载问题不同,我们在假设 1 和假设 3 中对 CPU 频率等系统参数与处理延迟之间的关系没有任何限制,文献[9]中也体现出这一思想。这是更加实际的设置,原因如下。首先,任务的复杂度等变量应该被建模为一系列独立的随机变量。这是因为计算本身的不确定性,而且它们的分布可能会由于任务类型的变化而发生突变①。另外,节点的计算能力,例如每个节点所分配用于计算的 CPU 频率、CPU 内核数目、内存大小等都有差异,而且也可能会发生突变。上面提到的所有这些不确定性使得系统难以预测单一节点处理不同任务的时延。而且,单个节点获取系统的全局节点的信息会花费大量通信开销。在传统模型中,如 Mao 等[5]假设时延简单地由任务数据长度和配置的 CPU 频率决定,然而在实际系统中,单个任务节点很难像传统模型那样准确地估计计算时延和等待时延。

本文中,处理时延和等待时延仅在相应的任务处理结束之后反馈给系统。也就是说,等待时延的观测值 τ_t^W 和处理延迟的观测值 τ_t^P 被视为后验信息。这些时延信息可以通过相应节点完成任务后的时间戳反馈获得。因此,随机变量 $W_t(i)$ 和 $P_t(i)$ 在每一时刻的样本值可以由以下公式计算得到:

$$w_t(i) = \frac{\tau_t^W}{Q_t(i)} 1\{I_t = i\}, p_t(i) = \frac{\tau_t^P}{L_t} 1\{I_t = i\},$$

式中: I_t 表示处理任务 t 的节点的编号;$1\{\cdot\}$ 为指示函数,当花括号内条件满足时,该函数取值为 1,否则为 0。

1.2　问题建模

一般的长期平均时延最小化问题可以被建模成如下形式:

$$\begin{aligned} \underset{\{I_t,\forall t\}}{\text{minimize}} \quad & \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{K} U_t(i) 1\{I_t = i\} \\ \text{subject to} \quad & I_t \in \mathcal{I}, t = 1,2,\cdots,T. \end{aligned} \tag{4}$$

① 例如,移动用户在使用不同应用程序时,任务类型、任务复杂度等参数会发生改变。

解决上述问题有两个困难。首先,它是一个随机规划问题。在第 t 个任务完成之前,无法得到时延 $U_t(i)$ 的确切信息。另外,即使 $U_t(i)$ 为已知的先验信息,这个问题仍然是组合优化问题,复杂度为 $\mathcal{O}(K^T)$ 量级。这是因为先前的卸载决策确定了每个雾节点中的队列长度,并进一步影响未来任务的决策。有关示例,请参阅文献[10]。为使任务卸载策略具备在线更新、动态调整的能力,一种流行的方法是在每个时间段将这个具有挑战性的随机和组合优化问题转换为一个低复杂度的顺序决策问题[2-7]。基于在之前的 $(t-1)$ 个时隙中做出的任务卸载决策,最优策略变为将任务 t 卸载到 t 时刻能够获得最低时延的节点。同时,在随机问题的框架下[12],关注随机变量的期望是更自然的做法,即 $\mathbb{E}[U_t(i)]$。因此,表达式(4)中的问题在第 t 个时隙中变成以下问题:

$$\underset{I_t\in\mathcal{I}}{\text{minimize}}\sum_{i\in\mathcal{I}}\mathbb{E}[U_t(i)]1\{I_t=i\}. \tag{5}$$

然而,上述公式仍然是随机规划问题。将经验平均值作为期望 $\mathbb{E}[U_t(i)]$ 的估计值虽然可行,但此信息可能由于观察数量有限而不准确。值得注意的是,有关节点 i 的信息来自节点 i 在完成相应任务时的反馈,因此为了获得某个特定节点更准确的信息,任务节点必须将更多任务卸载到该节点,即使它可能不是经验上最好的卸载节点。因此,在这个问题中存在探索和开发的权衡。下文致力于找到一种有效的方案来解决表达式(5)中的问题。

2 高效的任务卸载策略

2.1 基于 SW-UCB 的任务卸载

当某个任务产生时,需要确定一个雾节点(辅助节点或任务节点)来处理它。与此同时,必须在上述探索和开发之间取得平衡。这种权衡促使我们诉诸老虎机模型。具体来说,任务卸载被建模为非稳态多臂老虎机(MAB)问题[14],其中$\mathcal{I}$中的每个节点被视为老虎机的一个操纵杆(arm)。

按照前面的假设,任务节点在每个时隙开始时产生一个任务。令 $\tau_s\leqslant s+\tau_{\max}$ 为收到第 s 个任务反馈的时间,其中 $\tau_{\max}$ 是最大容许时延。如果任务时延超过最大容许时延,即 $\tau_s>s+\tau_{\max}$,该任务视为失败并被丢弃。根据文献[14],随机变量 $W_t(i)$ 和 $P_t(i)$ 的估计值可以由历史观测值的滑动窗口平均值得到,即

$$\begin{aligned}\overline{W}_t(\tau,i)&:=\frac{1}{N_t(\tau,i)}\sum_{s=t-\tau}^{t-1}w_s(i)1\{I_s=i,\tau_s\leqslant t\},\\ \overline{P}_t(\tau,i)&:=\frac{1}{N_t(\tau,i)}\sum_{s=t-\tau}^{t-1}p_s(i)1\{I_s=i,\tau_s\leqslant t\},\end{aligned} \tag{6}$$

式中 $\tau>0$ 表示滑动窗口的长度,以及

$$N_t(\tau,i):=\sum_{s=t-\tau}^{t-1}1\{I_s=i,\tau_s\leqslant t\}. \tag{7}$$

那么,时延 $U_t(i)$ 就可以被估计为

$$\overline{\mu}_t(\tau,i):=L_tT(i)+Q_t(i)\overline{W}_t(\tau,i)+L_t\overline{P}_t(\tau,i). \tag{8}$$

注意到式(8)中的延迟是根据 $W_s(i)$ 和 $P_s(i)$ 的历史信息估算的,而不是之前的延迟值,即 $U_s(i)$,$s<t$。这是因为时延 $U_s(i)$ 严重依赖于队列长度 $Q_t(i)$ 和任务长度 L_t,这对于不同的任务、不同的节点可能会有很大差异。因此,直接用之前的时延观测值估计 $U_t(i)$ 是不可靠的。另一方面,处理每一个比特的任务所需的时间通常由节点能力和任务类型确定,这些因素相对稳定并因此适合用样本均值估计。

将节点 i 用于处理任务 t 的总时间与最大容忍时延进行比较,定义该时间差为完成任务的奖励,即 $X_t(i):=\tau_{\max}-U_t(i)$。显然,负的奖励表示任务失败。根据式(8)中所估计的时延,奖励的估计值由下式给出

$$\overline{X}_t(\tau,i):=\tau_{\max}-\overline{\mu}_t(\tau,i). \tag{9}$$

我们用 UCB 算法来权衡探索和开发之间的关系。从本质上来说,这种折中关系由探索奖励 $c_t(\tau,i)$ 来衡量,即探索节点 i 会有额外的奖励。在本文中,令

$$c_t(\tau,i):=\tau_{\max}\sqrt{\frac{\xi\ln(t\wedge\tau)}{N_t(\tau,i)}}, \tag{10}$$

式中:ξ 表示探索常数,$t\wedge\tau$ 表示取 t 和 τ 之间的最小值。按照这种方式选取探索奖励 $c_t(\tau,i)$ 的好处将在后文分析。根据该置信上界,即 $\overline{X}_t(\tau,i)+c_t(\tau,i)$,用来处理任务 t 的节点可按照如下方式选择:

$$I_t=\text{argmax}_{i\in\mathcal{I}}\overline{X}_t(\tau,i)+c_t(\tau,i). \tag{11}$$

本文算法的具体流程见算法 1。

算法 1 TOS(task offloading with sliding-window-UCB)任务卸载算法

步骤 1) 设置合适的窗长参数 τ。设置索引

$t=1,\overline{W}_t(\tau,i)=\overline{P}_t(\tau,i)=0,\forall i\in\mathcal{I}$。

步骤 2)　令 $I_t=t$,卸载任务 t 至节点 I_t;

步骤 3)　$t=t+1$;

步骤 4)　若 $t>K$,转至步骤 5);否则跳转至步骤 2);

步骤 5)　根据式(6)、式(7)更新参数 $\overline{W}_t(\tau,i),\overline{P}_t(\tau,i)$ 和 $N_t(\tau,i)$。

步骤 6)　根据式(9)、式(10)更新参数 $\overline{X}_t(\tau,i),c_t(\tau,i)$。

步骤 7)　根据式(11)做出决策 I_t,卸载任务 t 至节点 I_t;

步骤 8)　$t=t+1$;

步骤 9)　若 $t>T$,算法结束;否则跳转至步骤 5)。

由算法 1 可知,本文算法的复杂度主要集中于步骤 5)。若直接按照式(6)、式(7)执行步骤 5),则该步骤复杂度为 $\mathcal{O}(K\tau)$。因此,在执行每一次任务卸载时,决策复杂度为 $\mathcal{O}(K\tau)$。若步骤 5)采取增量更新的方式,只关注最新移入或移出窗口的部分,每次决策的复杂度可进一步降低为 $\mathcal{O}(K)$。根据假设 2,为保障算法运行,每个节点广播其队列长度。因此,对于算法本身来说,功耗主要集中于广播队列长度以及执行决策的功耗。此外,任务卸载本身还包含实际卸载任务的功耗(任务节点承担)以及实际执行任务的功耗(按需部分由辅助节点承担)。

虽然上面提出的任务卸载模型本质上是一个非稳态的 MAB 模型,但与文献[14]中提出的传统模型相比,存在两个主要差异。首先,在传统模型中,决策的反馈是即时获得的。而在我们的模型中,如式(6)所示,在任务完成之前,即 $\tau_s\leqslant t$ 时,反馈是无法得到的。而相应的延迟是无法忽略的,因为它恰好是我们需要的信息。值得注意的是,延迟的反馈会影响算法性能,这一现象被 Joulani 等指出并在文献[15]中分析。其次,常规的非稳态 MAB 模型[14]假设最好的节点只在断点处改变。但是,我们的模型允许最佳节点在不同时刻、处理不同任务时发生变化。因此,目前已有的 SW-UCB 算法的性能保证不能直接应用于我们提出的 TOS。

2.2　理论分析

根据式(2)和式(3),期望时延 $\mathbb{E}[U_t(i)]$ 可以表示为

$$\mu_t(i):=\mathbb{E}[U_t(i)]=L_tT(i)+Q_t(i)\mu_t^W(i)+L_i\mu_t^P(i).$$

给定前 $(t-1)$ 个任务的卸载策略,根据式(5),处理任务 t 的最优节点为 $i_t^*:=\operatorname{argmin}_{i\in\mathcal{I}}\mu_t(i)$。此外,用

$$\tilde{N}_T(i):=\sum_{t=1}^{T}1\{I_t=i\neq i_t^*\}$$

表示前 T 个时隙中节点 i 为非最优节点时所获得的任务数量。根据假设 3,系统参数的期望值在每个断点会发生突变。用 Υ_T 表示时刻 T 之前的断点数目。以下推论给出 $\mathbb{E}(\tilde{N}_T(i))$ 的上界。

推论 1　假设 $\xi>1/2$。对于每个节点 $i\in\mathcal{I}$,期望 $\mathbb{E}(\tilde{N}_T(i))$ 的上界为

$$\mathbb{E}[\tilde{N}_T(i)]\leqslant\frac{T\ln\tau}{\tau}B(\tau)+\Upsilon_T(\tau+\tau_{\max})+\frac{2(\ln\tau)^2+2\ln\tau}{\ln(1+\eta)},\tag{12}$$

其中

$$B(\tau):=\left(\frac{4U_{\max}^2\xi}{(\Delta\mu_T(i))^2}+\tau_{\max}\right)\frac{T/\tau}{T/\tau}+2\frac{\left\lceil\frac{\ln\tau}{\ln(1+\eta)}\right\rceil}{\ln\tau},$$

$$\Delta\mu_T(i):=\min_{t\in\{1,\cdots,T\},i_t^*\neq i}\mu_t(i)-\mu_t(i_t^*).$$

证明见附录。显然,该上界取决于总任务数 T、断点数 Υ_T 以及窗长 τ 的选择。从式(12)中,可以看出第 1 项 $(TB(\tau)\ln\tau)/\tau$ 随着 τ 增加而减少。另一方面,后面两项,即 $\Upsilon_T(\tau+\tau_{\max})$ 和 $(2(\ln\tau)^2+2\ln\tau)/\ln(1+\eta)$,随着 τ 增加而增加。这种折中关系与我们的直觉一致,即更高的窗口长度 τ 有助于在稳定情况下更好地估计,但同时导致对环境突然变化的缓慢反应。因此,在式(12)中的不同项之间存在权衡。为了在稳定和突然变化的环境之间取得平衡,类似于文献[14],将 τ 定义为

$$\tau=2\tau_{\max}\sqrt{(T\ln T)/\Upsilon_T}.\tag{13}$$

相应地,有以下推论。

推论 2　当 $\Upsilon_T=\mathcal{O}(T^\beta),\beta\in[0,1)$,以及 $T\to\infty$ 时,期望 $\mathbb{E}[\tilde{N}_T(i)]$ 的数量级为

$$\mathbb{E}[\tilde{N}_T(i)]=\mathcal{O}(\sqrt{T\Upsilon_T\ln T}).$$

证明　令 $\tau=2\tau_{\max}\sqrt{(T\ln T)/\Upsilon_T}$,取 $T\to\infty$,易得。

为了证明所提出的算法的最优性,定义卸载前 T 个任务的伪后悔度为

$$\zeta_T := \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\left(U_t(I_t) - \mathbb{E}\left[U_t(i_t^*)\right]\right)\right]. \tag{14}$$

关于 ζ_T, 有如下推论。

推论 3 当 $\Upsilon_T = \mathcal{O}(T^\beta)$, $\beta \in [0,1)$ 时,算法 1 中的算法是渐进最优的,即

$$\lim_{T\to\infty}\zeta_T \xrightarrow{\text{a. s.}} 0. \tag{15}$$

证明 注意到 $U_t(I_t) - \mathbb{E}\left[U_t(i_t^*)\right] \leqslant \tau_{\max}1\{I_t \neq i_t^*\}$。伪后悔度可放缩为

$$\zeta_T \leqslant \frac{1}{T}\mathbb{E}\left[\sum_{t=1}^{T}\tau_{\max}1\{I_t \neq i_t^*\}\right]$$

$$\leqslant \frac{\tau_{\max}}{T}\sum_{i\in\mathcal{I}}\mathbb{E}\left[\tilde{N}_T(i)\right].$$

根据推论 1 和推论 2,有

$$\zeta_T = \mathcal{O}\left(\sqrt{(\Upsilon_T \ln T)/T}\right) = \mathcal{O}\left(T^{\frac{\beta-1}{2}}\sqrt{\ln T}\right).$$

那么对于任意 $\varepsilon > 0$, 存在一个有限整数 N_ε, 使得

$$\mathbb{P}(|\zeta_T| \geqslant \varepsilon) = 0, \forall T \geqslant N_\varepsilon.$$

因此,

$$\sum_{T=1}^{\infty}\mathbb{P}(|\zeta_T| \geqslant \varepsilon) \leqslant N_\varepsilon < \infty.$$

上述表达式表明 $\lim_{T\to\infty}\zeta_T \xrightarrow{\text{a. s.}} 0$。证毕。

推论 3 说明,在条件 $\Upsilon_T = \mathcal{O}(T^\beta)$, $\beta \in [0,1)$ 下,当 $T \to \infty$ 时,该算法的后悔度以概率为 1 趋近于零。由 $\lim_{T\to\infty}|\zeta_{T+1}|/|\zeta_T| = (\beta+1)/2$ 知,该算法线性收敛。

3 仿真实验

在这一章节中,通过仿真 10^4 次任务卸载来验证本文提出的算法。每次任务卸载时,任务节点产生一个任务。以下是各个场景中共同的仿真参数设定。

· 该雾赋能的网络包含1个任务节点和 9 个辅助节点;

· 每个时隙长度为20 ms。任务数据长度服从 Unif(1,15) KB;

· 最大允许时延为 $\tau_{\max} = 20$ 时隙,参数 $\xi = 0.6$;

· 每一比特任务的处理时延按照公式 $P_t(i) = \sigma_t^{\text{cplx}}/\sigma_i^{\text{CPU}}$ 仿真,其中 σ_t^{cplx} 表示任务 t 的复杂度, σ_i^{CPU} 反映节点 i 的计算能力。二者服从分布 Unif(1,10);

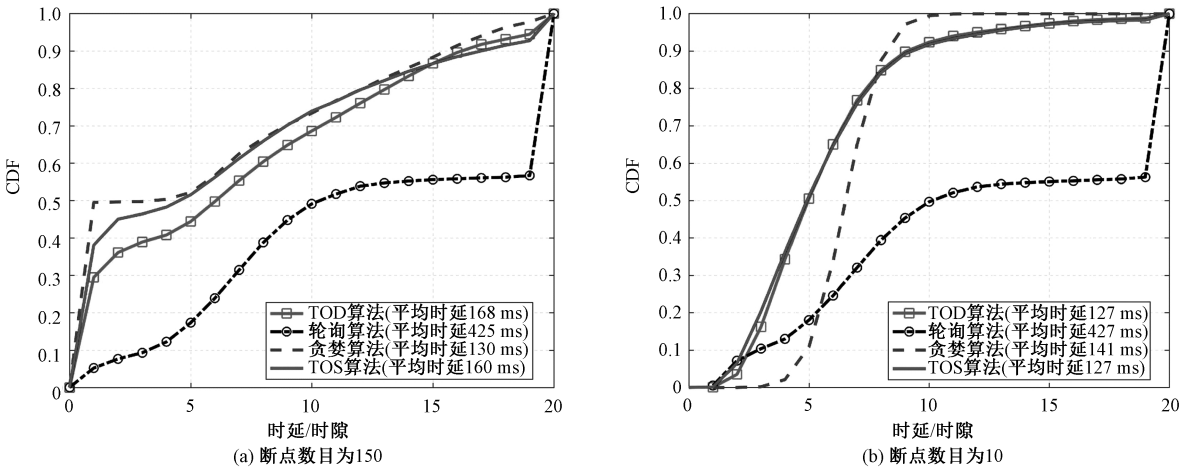
· 节点 i 的计算能力在断点处的变化遵从 $\sigma_i^{\text{CPU}} = \sigma_i^{\text{CPU}}/16$ 或者 $\sigma_i^{\text{CPU}} = \sigma_i^{\text{CPU}} \times 16$。

将算法 TOS 的性能与两个基本的策略(贪婪算法和轮询算法)、以及我们之前提出的基于折扣因子的在线任务卸载算法[16],即 TOD(task offloading with discounted-UCB)算法,进行比较①。在贪婪算法中,假设任务节点已知任务卸载相关随机变量的每一个样本值,并在每个时隙将任务卸载给时延最小的节点。值得注意的是,虽然贪婪算法代表每一时刻所能做出的最优决策,但该贪婪算法不是因果的,现实中不能实现。此外,由于当前时刻决策与下一时刻状态相互耦合,每一时刻的最优决策联合起来并不一定代表问题(4)的最优解。在轮询算法中,每个任务以循环的方式依次分配给各个雾节点。在 TOD 算法中,基于 D-UCB 框架,利用折扣因子 γ 应对非稳态环境。据我们所知,TOD 算法为符合本文场景的最新算法。

在图 2 中,通过仿真不同的断点数目,TOS 算法的有效性和稳定性得到证明。算法性能通过仿真中各个任务的时延的累积分布函数(CDF, cumulative distribution function)来体现。图中, TOD 算法的折扣因子 γ 和 TOS 算法的滑动窗口长度 τ 各自按照其理论公式计算得到。图 2 中的两个场景都证明 TOS 算法的性能远远优于轮询算法,接近于贪婪算法,并且略微优于 TOD 算法。在图 2(a)中,当断点数目 Υ_T 设置成 150 时,平均每 67 个任务就会有一次系统参数的突变。这暗示着 TOS 算法能够快速学习并适应频繁的系统变化,并且适应能力在一定程度上优于 TOD 算法。同样值得注意的是,从图 2(b)可以看出,当系统变化次数非常有限($\Upsilon_T = 10$)时,TOS 甚至表现出比贪婪算法更小的平均时延。这一现象揭示每个时隙的最优决策组合起来并不一定是全局最优这一事实。同时,这也佐证了我们之前的分析,即节点的每一时刻的决策都会影响系统后续的状态,进而影响后续的决策。

图 3 绘制了不同方案的后悔度曲线。仿真

① 注意到本文的核心问题在于,当任务卸载的代价需通过在线学习获得时,如何权衡“探索”和“开发”之间的折中关系。因此,其他预知代价的算法,如文献[5-8]中提到的李雅普诺夫(Lyapunov)优化方法,在本文中难以直接适用。



(a) 断点数目为150　　(b) 断点数目为10

图 2　不同场景下任务处理时延的累积分布函数图

Fig. 2　CDFs of the task processing delay in different scenarios

中,后悔度的计算根据 $\hat{\zeta}_T := \sum_{t=1}^{T}(U_t(I_t) - U_t(i_t))/T$, 其中每一时刻的最优决策基于该时刻的真实样本值,即 $i_t = \text{argmax}_{i\in\mathcal{I}} U_t(i)$。在无限冲激响应(IIR)方案中,探索和开发分为两个阶段。探索阶段采用轮询的方式。在开发阶段,按照文献[16]中加权平均的方法将历史信息按照折扣因子 γ 加权求和。这一方法刚好对应了传统信号处理方法中的 IIR 滤波器。两个阶段最优的分配比例采用二分法搜索的方式以得到最小化后悔度的数值解。从图 3 中,可以看出,本文提出的 TOS 算法和之前提出的 TOD 算法相较于轮询算法和 IIR 算法都能够达到更低的后悔度。这表明我们提出的方案在处理探索和开发的折中关系时表现优秀。

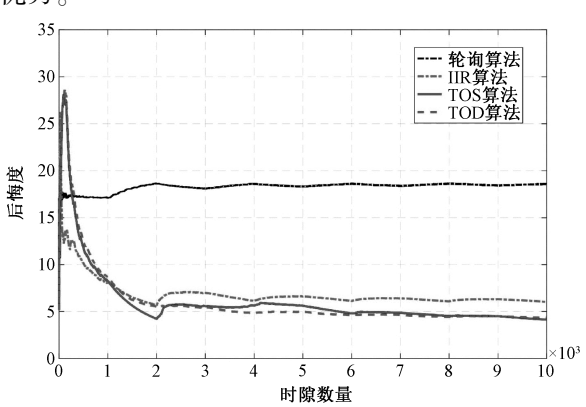


图 3　不同算法的平均后悔度曲线

Fig. 3　Average regret curves for different algorithms

4　结束语

本文研究雾赋能的网络中一种高效的在线任务卸载策略以及相应的性能保证。考虑到节点处理速度的期望可能在未知时刻发生突变,并且相关系统参数信息仅在完成相应任务之后才获得这一场景,将任务卸载问题建模成带有延迟老虎机反馈的随机优化问题。为解决这个问题,基于 UCB 算法提供一个高效的在线任务卸载方案,即 TOS。给定特定数量的断点 Υ_T, 证明卸载到非最佳节点的任务数量的上界是 $\mathcal{O}(\sqrt{T\Upsilon_T \ln T})$。此外,还证明,当任务数量趋近于无穷大时,伪后悔度以概率为 1 趋近于零。数值仿真证明,所提出的 TOS 算法能够在非稳态环境中学习并选择正确的节点以卸载任务,且表现优于 TOD 算法。

参考文献

[1] 王雷,王平建,向继. 云存储环境中的统一认证技术[J]. 中国科学院大学学报,2015,32(5):682-688.

[2] Dinh T Q, Tang J, La Q D, et al. Offloading in mobile edge computing: task allocation and computational frequency scaling[J]. IEEE Transaction on Communication, 2017, 65(8):3571-3584.

[3] You C, Huang K, Chae H, et al. Energy-efficient resource allocation for mobile-edge computation offloading[J]. IEEE Transaction on Wireless Communication, 2017, 16(3):1397-1411.

[4] Kwak J, Kim Y, Lee J, et al. DREAM: Dynamic resource and task allocation for energy minimization in mobile cloud systems [J]. IEEE Journal on Selected Areas of Communication, 2015, 33(12):2510-2523.

[5] Mao Y, Zhang J, Song S H, et al. Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems [J]. IEEE Transaction on Wireless Communication, 2017, 16(9):5994-6009.

[6] Pu L, Chen X, Xu J, et al. D2D fogging: an energy-efficient and incentive-aware task offloading framework via network-

assisted D2D collaboration[J]. IEEE Journal on Selected Areas of Communication, 2016, 34(12):3887-3901.

[7] Yang Y, Zhao S, Zhang W, et al. DEBTS: delay energy balanced task scheduling in homogeneous fog networks[J]. IEEE Internet Things Journal, 2018, 5(3):2094-2106.

[8] Chen X. Decentralized computation offloading game for mobile cloud computing[J]. IEEE Transaction on Parallel and Distributed Systems, 2015, 26(4):974-983.

[9] Chen T, Giannakis G B. Bandit convex optimization for scalable and dynamic IoT management[J]. IEEE Internet Things Journal, 2019, 6(1):1276-1286.

[10] Tekin C, Van Der Schaar M. An experts learning approach to mobile service offloading[C]//The IEEE Annual Allerton Conference on Communication, Control, and Computing. 2014: 643-650.

[11] Auer P, Cesa-Bianchi N, Fischer P. Finite-time analysis of the multiarmed bandit problem[J]. Machine Learning, 2002, 47(2):235-256.

[12] Bubeck S, Cesa-Bianchi N. Regret analysis of stochastic and nonstochastic multi-armed bandit problems[J] Foundations and Trends in Machine Learning, 2012, 5(1):1-122.

[13] Berry D A, Fristedt B. Bandit problems: sequential allocation of experiments[M]. London, U.K.: Chapman & Hall, 1985.

[14] Garivier A, Moulines E. On upper-confidence bound policies for switching bandit problems[C]//The 2011 International Conference on Algorithmic Learning Theory, 2011: 174-188.

[15] Joulani P, Gyorgy A, Szepesvari C. Online learning under delayed feedback[C]//The 30th International Conference on Machine Learning. 2013:1453-1461.

[16] Zhu Z, Liu T, Jin S, et al. Learn and pick right nodes to offload[EB/OL]. (2018-04-24)[2019-02-10]. https://arxiv.org/pdf/1804.08416.pdf.

附录

证明 首先,根据定义做如下事件分解:

$$\widetilde{N}_T(i) \leqslant \sum_{t=K+1}^{T} 1\{I_t = i \neq i_t^*, N_t(\tau,i) < A(\tau,i)\} + \sum_{t=K+1}^{T} 1\{I_t = i \neq i_t^*, N_t(\tau,i) \geqslant A(\tau,i)\} + 1,$$

其中 $A(\tau,i)$ 是参数 τ 的函数,具体形式将在后文给出。接下来,依次给出上述两个求和项的上界。第 1 项上界的推导需要借助文献[14]中的引理 1,即对于任意的 $i \in \mathcal{I}, \tau > 0, m > 0$, 下列不等式成立:

$$\sum_{t=K+1}^{T} 1\{I_t = i, \sum_{s=t-\tau}^{t-1} 1\{I_s = i\} < m\} \leqslant \left\lceil T/\tau \right\rceil m.$$

此外,将从时隙 $(t-\tau)$ 到时隙 $(t-1)$ 期间缺失的反馈数量表示为

$$G_t(\tau,i) := \sum_{s=t-\tau}^{t-1} 1\{I_s = i\} - N_t(\tau,i).$$

显然,缺失的反馈数目小于 $\tau_{\max}$, 即 $G_t(\tau,i) \leqslant \tau_{\max}, \forall t,i$。由于

$$\{t \mid \sum_{s=t-\tau}^{t-1} 1\{I_s = i\} < m\} = \{t \mid G_t(\tau,i) + N_t(\tau,i) < m\} \supseteq \{t \mid \tau_{\max} + N_t(\tau,i) < m\},$$

有

$$\sum_{t=K+1}^{T} 1\{I_t = i, \tau_{\max} + N_t(\tau,i) < m\} \leqslant \left\lceil T/\tau \right\rceil m.$$

令 $m = A(\tau,i) + \tau_{\max}$, 得

$$\sum_{t=K+1}^{T} 1\{I_t = i \neq i_t^*, N_t(\tau,i) < A(\tau,i)\} \leqslant \left\lceil T/\tau \right\rceil (A(\tau,i) + \tau_{\max}).$$

对于第 2 项,定义 $\mathcal{T}(\tau)$ 为"卸载良好"的集合。数学上,其定义为:

$$\mathcal{T}(\tau) := \{t \mid t \in \{K+1, \cdots, T\}; \mu_s^W(j) = \mu_t^W(j),$$
$$\mu_s^P(j) = \mu_t^P(j), \forall s \in (t - C(\tau), t), \forall j \in \mathcal{I}\},$$

式中:$C(\tau)$ 表示卸载时估计较差的任务的数目。特别地,令 $C(\tau) = \tau + \tau_{\max}$。将这些估计较差的任务孤立出来,得到以下不等式

$$\sum_{t=K+1}^{T} 1\{I_t = i \neq i_t^*, N_t(\tau,i) \geqslant A(\tau,i)\} \leqslant \Upsilon_T(\tau + \tau_{\max}) + \sum_{t \in \mathcal{T}(\tau)} 1\{I_t = i \neq i_t^*, N_t(\tau,i) \geqslant A(\tau,i)\}.$$

接下来,需要求得上式中最后一项的上界。注意到 3 个事实:

1)事件 $\{I_t = i \neq i_t^*\}$ 发生当且仅当事件 $\varepsilon_t(\tau,i) = \{\overline{U}_t(\tau,i_t^*) - c_t(\tau,i_t^*) \geqslant \overline{U}_t(\tau,i) - c_t(\tau,i)\}$ 发生;

2) $\varepsilon_t(\tau,i) \subseteq \{\overline{U}_t(\tau,i_t^*) - c_t(\tau,i_t^*) \geqslant \mu_t(i_t^*)\} \cup \{\overline{U}_t(\tau,i) - c_t(\tau,i) < \mu_t(i_t^*)\}$;

3) $\{\overline{U}_t(\tau,i) - c_t(\tau,i) < \mu_t(i_t^*)\} \subseteq \{\overline{U}_t(\tau,i) + c_t(\tau,i) < \mu_t(i)\} \cup \{\mu_t(i) - \mu_t(i_t^*) < 2c_t(\tau,i)\}$.

基于以上事实,可得

$$\{I_t = i \neq i_t^*, N_t(\tau,i) \geqslant A(\tau,i)\} \subseteq \{\overline{U}_t(\tau,i_t^*) - c_t(\tau,i_t^*) \geqslant \mu_t(i_t^*)\}$$
$$\cup \{\overline{U}_t(\tau,i) + c_t(\tau,i) < \mu_t(i)\} \cup \{\mu_t(i) - \mu_t(i_t^*) < 2c_t(\tau,i), N_t(\tau,i) \geqslant A(\tau,i)\}.$$

定义

$$\Delta\mu_T(i) := \min_{t\in\{1,\cdots,T\}, i_t^*\neq i} \mu_t(i) - \mu_t(i_t^*).$$

令 $A(\tau,i) := 4\tau_{\max}^2 \xi \ln\tau (\Delta\mu_T(i))^{-2}$。若上述不等式中的事件 $\{\mu_t(i) - \mu_t(i_t^*) < 2c_t(\tau,i), N_t(\tau,i) \geqslant A(\tau,i)\}$ 成立,则

$$\frac{\Delta\mu_T(i)}{2} = \tau_{\max}\sqrt{\frac{\xi\ln\tau}{A(\tau,i)}} \geqslant c_t(\tau,i). \tag{A1}$$

然而,根据 $\Delta\mu_T(i)$ 的定义有

$$\frac{\Delta\mu_T(i)}{2} \leqslant \frac{\mu_t(i) - \mu_t(i_t^*)}{2} < c_t(\tau,i). \tag{A2}$$

显然,式(A1)与式(A2)矛盾。因此事件 $\{\mu_t(i) - \mu_t(i_t^*) < 2c_t(\tau,i), N_t(\tau,i) \geqslant A(\tau,i)\}$ 发生的概率为零。对于事件 $\{\overline{U}_t(\tau,i_t^*) - c_t(\tau,i_t^*) \geqslant \mu_t(i_t^*)\}$,有

$$\mathbb{P}(\mu_t(i) - \overline{U}_t(\tau,i) > c_t(\tau,i)) = \mathbb{P}\left(\mu_t(i) - \overline{U}_t(\tau,i) > U_{\max}\sqrt{\xi\frac{\ln(t\wedge\tau)}{N_t(\tau,i)}}\right)$$
$$\overset{(a)}{\leqslant} \left\lceil\frac{\ln(t\wedge\tau)}{\ln(1+\eta)}\right\rceil \exp\left(-2\xi\ln(t\wedge\tau)\left(1-\frac{\eta^2}{16}\right)\right),$$

其中不等式(a)由文献[14]中的定理 4 得来。令 $\eta := 4\sqrt{1-1/(2\xi)}$,$\xi > 1/2$,得

$$\mathbb{P}(\mu_t(i) - \overline{U}_t(\tau,i) > c_t(\tau,i)) \leqslant \left\lceil\frac{\ln(t\wedge\tau)}{\ln(1+\eta)}\right\rceil (t\wedge\tau)^{-1}.$$

根据对称性,可知

$$\mathbb{P}(\overline{U}_t(\tau,i_t^*) - c_t(\tau,i_t^*) \geqslant \mu_t(i_t^*)) = \mathbb{P}(\overline{U}_t(\tau,i) + c_t(\tau,i) < \mu_t(i)).$$

因此

$$\mathbb{E}\left[\sum_{t=K+1}^{T} 1\{I_t = i \neq i_t^*, N_t(\tau,i) \geqslant A(\tau,i)\}\right] \leqslant \Upsilon_T(\tau+\tau_{\max}) + 2\sum_{t\in\mathcal{T}(\tau)}\left\lceil\frac{\ln(t\wedge\tau)}{\ln(1+\eta)}\right\rceil (t\wedge\tau)^{-1}.$$

将所有相关项求和并作适当放缩可得

$$\mathbb{E}[\tilde{N}_T(i)] \leqslant \frac{T\ln\tau}{\tau}B(\tau) + \Upsilon_T(\tau+\tau_{\max}) + \frac{2(\ln\tau)^2 + 2\ln\tau}{\ln(1+\eta)},$$

其中

$$B(\tau) := \left(\frac{4U_{\max}^2\xi}{(\Delta\mu_T(i))^2} + \tau_{\max}\right)\frac{\lceil T/\tau\rceil}{T/\tau} + 2\frac{\left\lceil\frac{\ln\tau}{\ln(1+\eta)}\right\rceil}{\ln\tau}.$$

证毕。