An Overhead-Optimizing Task Scheduling Strategy for Ad-hoc Based Mobile Edge Computing

Li Tianze, Wu Muqing, Zhao Min, Liao Wenxing

一、本文贡献：

基于 ad hoc 的移动边缘计算系统，提出了一种**开销优化的多设备任务调度策略**。该任务调度策略考虑了 opportunity consumption，time delay，energy consumption，and monetary cost，旨在**最小化每个移动设备的开销**。

1. 提出了基于 ad-hoc 的移动边缘计算系统模型，分析了移动设备的开销。
2. 将任务调度问题描述为**一个分布式多设备任务调度博弈**。通过构造一个势函数，证明了**任务调度博弈是一个势博弈**，它具有有限的改进性质，且总是具有纳什均衡。
3. 设计了一种开销优化的多设备任务调度算法。
4. 进行仿真以评估所提出策略的效果。

结果表明，所提出的任务调度策略能有效地降低移动设备的开销，并成功地完成任务。

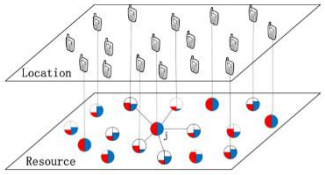
二、系统模型：

移动边缘计算系统分为**基于基础设施的移动边缘云**和**基于 ad hoc 的移动边缘虚拟云**。

基于基础设施的云：由**远程服务器**授权，可以为移动设备提供高效的资源。

基于 ad-hoc 的虚拟云：**由一组移动设备组成**，这些移动设备协同工作以完成任务

本文基于 ad-hoc 的移动边缘计算系统。(ad-hoc 架构如下)



基于 ad-hoc 的移动边缘计算系统实例。红色（蓝色）部分表示可用带宽（计算）资源。

移动设备 j 是一个资源丰富的设备，它的邻居设备可以使用它的资源。

在 ad-hoc 系统中，任务调度问题的**挑战**:

1)设备的移动设备的接触持续时间变化，任务调度策略必须考虑接触持续时间。

2)任务调度涉及**额外的数据传输**，太多的设备同时通过无线接入来加载任务，会对彼此造成严重的**干扰**，从而降低数据传输速率并导致低能量效率。

3)计算资源不能自由使用，当使用另一台设备的资源时，任务发布者应**付费**作为奖励。

1. 任务执行模型

1) Local Execution:

任务执行时间：$T_i^l = (D_i + D_i^w)/F_i^l$

F^l_i:移动设备 i 的计算能力 ；

D_i:任务的工作负载 ； D^w_i:等待在移动设备 i 中执行的任务的工作负载

能耗： $E_i^l = v_i D_i$　　v_i:每个执行单元的能耗

本地计算的开销： $Z_i^l = \alpha_l T_i^l + \beta_l E_i^l$

α l，β l 是权重因子，0≤α l， β l≤1， α l+β l=1。**不同的设备选不同的权重因子**

2) Offloaded Execution:

Ψi 表示移动设备中有任务需要执行为 1，否则为 0。**Φ_ij** 表示移动设备 i 的任务加载到其相邻设备 j 为 1，否则为 0。**Φ_ii** 指示移动设备 i 是否本地执行其任务。

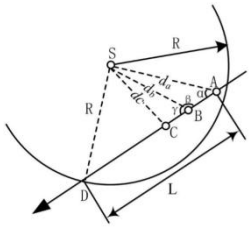
移动设备 i 的邻居表示为 **Ni={j，k，…}**，Ni 中的每个移动设备可直接与 i 通信。

约束： $\phi_{ij} = 0, j \notin N_i$　确保可以将任务加载到与 i 无线连接的移动设备上。

$\phi_{ij} = \{0,1\}$　表示任务可以在本地执行或加载到附近的设备上。

$\sum_{j \in N_i} \phi_{ij} = \psi_i, i \in S$　表示所有移动设备的任务应仅安排一次。

2. 移动设备的接触持续时间



移动设备 i 和 j 之间的相对移动示例图->

假设移动设备 i 在点 A 和移动设备 j 在初始时刻 s 点处

移动设备 i 和 j 之间的接触持续时间为：

$$T_{ij}=\frac{d_{sa}\cos\alpha+\sqrt{R^2-d_{sa}{}^2(1-\cos\alpha^2)}}{v}-2\Delta t \qquad v=\frac{\sqrt{\frac{1}{2}(d_{sa}{}^2+d_{sc}{}^2)-d_{sb}{}^2}}{\Delta t} \qquad \cos\alpha=\frac{d_{sa}{}^2+(v\cdot\Delta t)^2-d_{sb}{}^2}{2d_{sa}\cdot(v\cdot\Delta t)}$$

R 是移动设备 i 和 j 之间的无线链路的最大距离，它们之间的相对运动速度为 v

3. 通信和计算模型

移动设备 j 的邻居设备 Nj 的任务调度决策， Φ_Nj=（Φ_lj, Φ_kj, …, Φ_nj）

移动设备 i 在加载任务时会产生一些消耗：

1)time consumption：Tˆc_n = 任务的输入数据发送时间 tˆs_ij + 在附近设备上执行任务的时间 tˆex_ij +接收输出数据的时间 tˆr_ij

2)energy consumption: Eˆc_n = 数据发送能量 Eˆs_ij + 数据接收能量 Eˆr_ij

3)monetary consumption: 设移动设备 j 一字节的输入数据支付 pˆin_j，per execution unit 支付 pˆ{ex}_{v_ci}，（忽略了输出数据的 monetary cost）

$$P_i^c=\sum_{j\in N_i}p_j^{in}B_i\phi_{ij}+\sum_{j\in N_i}p_j^{ex}D_i\phi_{ij}$$

忽略了移动设备接收输出数据的能量和时间消耗，即 tr ij=0，Er ij=0。

因此，移动设备的总时间消耗 Tˆc_i 和总能耗表示为：

$$T_i^c=B_i\Big/\sum_{j\in N_i}r_{ij}\phi_{ij}+D_i\Big/\sum_{j\in N_i}F_{ij}\phi_{ij} \qquad E_i^c=P_i^sB_i\Big/\sum_{j\in N_i}r_{ij}\phi_{ij}$$

F_ij:由移动设备 j 分配给移动设备 i 的计算资源

Pˆs_i:设备 i 的传输功率

r_ij:无线信道上移动设备 i 和 j 之间的期望吞吐量 $r_{ij}=R_j\frac{w_{ij}}{w_{ij}+\sum_{k\in N_j,k\neq i}w_{kj}}$

Rj 表示移动设备的最大传输速率 j

Wij 是表示无线竞争信道中的移动设备 i 的权重以访问移动设备 j 的整数。

移动设备 i 的总开销： $Z_i^c=\alpha_cT_i^c+\beta_cE_i^c+\gamma_cP_i^c$

α c、β c、γ c 和为 1，并且根据移动设备的需求，每个因子可以设置不同的值

三、Task Offloading Analysis

移动设备 i 在 N_j 中的**所有任务调度决策表示为Φ_{Nj\i}=（Φ_kj, Φ_lj, …, Φ_Nj）**

如果Φ_ii = 1, Zi = Zˆl_i，如果Φ_ii = 0, Zi = Zˆc_i.

上述问题可以表述为一个开销优化的多设备任务调度博弈 $\Gamma=(S,\{\phi_{ij}\}_{i,j\in S},\{Z_i\}_{i\in S})$

Φ*_ij 为 $\min_{\phi_{ij}\in\{0,1\}}Z_i(\phi_{ij},\phi_{N_j\backslash i})$

决定 Φ*_ij 应满足以下四个约束：

1） **opportunity consumption 约束**。只有那些机会消耗小于阈值的移动设备才能提供资源，这是为了在移动设备中没有足够的资源时避免资源消耗。

2） **Overloading 约束。**分配给 j 的所有任务应小于 j 的剩余计算资源 $\sum_{i\in N_j}D_i\phi_{ij}^*\le c_j, j\in S$

3）**延迟容限**。接触持续时间应该大于总时间消耗：$T_{ij}<t_i^c$，执行时间应小于时间延迟限制 Tmax。

4） **利益约束**。$Z_n^c\le Z_n^l$

如果在均衡中没有移动设备可通过改变自己的策略来进一步降低其开销，那么 strategy profile 是纳什均衡。开销优化的多设备任务调度博弈是一个**势博弈**，该博弈存在**纳什均衡**。

A game is named the potential game if there is a potential function $\Phi(\phi)$ which admits that for every mobile device $i\in S$, if: $Z_i(\phi'_{ij},\phi_{N_j\backslash i})<Z_i(\phi_{ij},\phi_{N_j\backslash i})$ We have: $\Phi_i(\phi'_{ij},\phi_{N_j\backslash i})<\Phi_i(\phi_{ij},\phi_{N_j\backslash i})$. There are some attractive properties for a potential game, which are that it always possesses a Nash equilibrium and any asynchronous better update process must be finite and leads to a Nash equilibrium [31]. Next,

$$\Phi(\phi_{ij})=\frac{1}{2}\sum_{i=1}^{N}\sum_{k\neq i}w_{ij}w_{kj}U_{\{\phi_{ij}=\phi_{kj}\}}U_{\{\phi_{ij}>0\}}+\sum_{i=1}^{N}w_{ij}^2H_{ij}U_{\{\phi_{ij}=0\}}$$

四、算法设计

在这种任务调度机制中，每个移动设备根据从其邻居设备接收到的这些参数进行分析。如果一个移动设备是一个为用户卸载有益的 user，那么它将竞争决策更新的机会。如果移动设备得到机会，它可以更新其任务调度决策。否则，移动设备应再次进行分析，并竞争下一次更新机会。具体流程：

1）初始化。初始卸载决定设置为Φij=0。

2）运行任务调度机制。两次广播之间的时间为一个决策时隙。在一个决策时隙 t 中，能够使设备 i 的开销最小化的任务卸载决策Φ*ij（t）是**最优的调度决策**。在该时隙 t 中，由移动设备 i 和 j 在协商后做出的任务调度决策Φij（t）是**最终调度决策**。

3）在 t 时，i 的初始调度决策是Φij（t-1）。如果 i 可更改此决策时隙中的调度策略来减少其开销，则定义更新响应 BUi（t）。应用潜在博弈的有限改进特性，设计规则。

4）对于决策更新消息请求，采用 random back-off mechanism，避免了来自不同移动设备的更新消息的冲突问题。

Algorithm 1 Initialization algorithm

Initialization:
for each mobile device i **do**
  get the state of itself $s_i = (e_i, c_i, e_i^0, c_i^0)$
  compute the opportunity consumption h_i
  if $h_i > h_0$ **then**
    get the parameters about its resource
    get its neighbor devices set N_i
    compute the contact duration with its neighbors $T_{ij}, j \in N_i$
    broadcast these parameters about its resource and these contact durations with its neighbors
  end if
  for a given task a_i
  compute the execution time T_i^l
  make the initial task scheduling decisions $\phi_{ij} = 0 (j \in N_i)$
end for
End initialization

Algorithm 2 Overhead-optimizing Task Scheduling Algorithm

loop for each decision slot t
for each mobile device i in decision slot t **do**
  receive the parameters of its neighbor mobile devices
  get these contact durations $T_{ij}, j \in N_i$ with its neighbors
  if $T_i^l > T_{\max}$ **then**
    select those mobile devices V_i which satisfy these four constraints as described in section IV.
  else
    compute Z_i^l according to formula 6 and Z_i^c on each mobile device nearby.
    select those mobile devices V_i which satisfy $Z_i^c \le Z_i^l$ and these four constraints as described in section IV.
  end if
  find the best task scheduling decision $\phi_{ij}^*(t)$.
  if $(\phi_{ij} = 0, \phi_{ij}^* = 1)$ or $(\phi_{ij} = 1, \phi_{ij}^* = 0)$ **then**
    send REQ to the mobile device j with a random back-off mechanism.
    if receive the REP message for itself **then**
      update the final decision $\phi_{ij}(t) = \phi_{ij}^*(t)$.
    else
      keep the initial decision $\phi_{ij}(t) = \phi_{ij}(t-1)$.
    end if
  else
    keep the initial decision $\phi_{ij}(t) = \phi_{ij}(t-1)$.
  end if
end for
end loop until no REQ in the system for K decision slots

算法分析：计算复杂度为 O（CN），the algorithm can be terminated within at most $\frac{1}{2}W_{\max}^2 N^2 + W_{\max}^2 H_{\max} N / W_{\min}$ iteration times.

五、实验

对比算法：

1）无任务调度的本地计算：每个移动设备选择自己执行任务

2）随机选择的设备进行任务调度

3）基于交叉熵的优化方案，采用集中式交叉熵方法求解任务调度问题

指标：

1） System performance

2） Influence factors analysis

(1) Communication data size

(2) Computation size