Price-Based Distributed Offloading for Mobile-Edge Computing with Computation Capacity Constraints

Mengyu Liu and Yuan Liu, Member, IEEE

**目标：**边缘云设置价格使其**收益最大化**，每个用户单独设计卸载决策，以将其**成本**（延迟+支付）**最小化**。

**系统模型：**

K个用户和一个基站

用户的计算数据可以按**位**任意划分，分为部分本地计算和部分卸载。

总带宽B (平均分配给K个用户)

C\_k：user k计算 1 bit的输入数据的CPU cycles（cycles/bit）

R\_k：user k的输入数据大小（bit）

l\_k：卸载到边缘云的数据大小（bit）

R\_k-l\_k：本地计算数据大小（bit）

F\_k：用户k的本地CPU频率 （cycles/s）

C:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\8B86E10.tmp**本地计算时间：**

C:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\8CE7CE9E.tmp**卸载时间**：

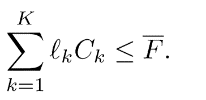
t\_{oﬀ,k} = the uplink transmission time （t\_{u,k}） + the execution time at the cloud t\_{c,k} + the downlink feedback time t\_{d,k}

C:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\3942AFC0.tmpC:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\4ECAD968.tmpC:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\F80566CA.tmp其中：

α\_k表示向cloud中加载的输出与输入比特的比例

f\_{c,k} 表示分配给用户k的边缘云的计算速度

f\_C : cloud 的总的计算速度C:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\B7F1945C.tmp

**总的时间：**

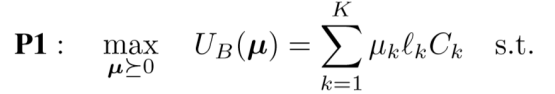
边缘云有计算能力限制：其计算每个加载周期中接收数据之和的CPU周期上限为F

**Stackelberg Game Formulation：**

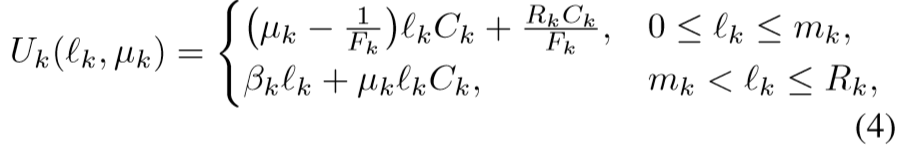
应用Stackelberg博弈来模拟边缘云和用户之间的交互，其中边缘云是leader，用户是follower。Edge cloud（leader）首先对用户的CPU周期定价。然后，用户（follower）根据edge cloud公布的价格，将输入数据分别划分为本地计算和卸载。

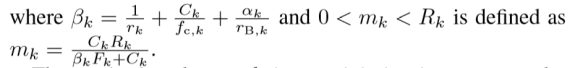
定义用户的CPU周期价格：**µ = {µ1,… ,µK}.**

**边缘云方面：**目标是最大化它的收益。优化问题表示为：

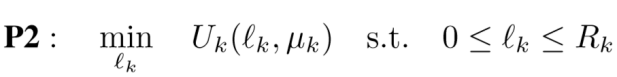


**C:\Users\28174\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\5144BEFA.tmp用户方面：**每个用户的成本=延迟时间+边缘云收取的费用：

****



β\_k ：1 bit的数据卸载时间。m\_k是一个分界线：l\_k≤m\_k,本地计算时间>卸载时间，l\_k > m\_k,卸载时间>本地计算时间。

每个用户k的目标是根据由边缘云给定的价格μ\_k选择最优卸载数据大小l\_k来最小化其自身成本。问题表示为：

**OPTIMAL ALGORITHM :**

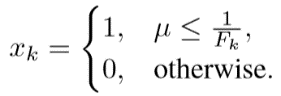
**反向归纳：**根据给定的价格µ通过解决P2得到 user k 的 卸载策略l^∗\_k；边缘云知道每个用户的卸载决策 l^∗\_k(µk)，然后通过解决P1来确定其最优价格µ^∗。

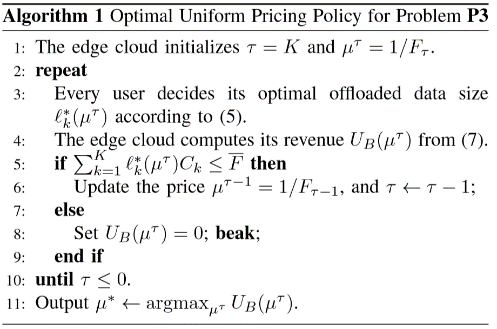
本文考虑了两种最优定价策略，即统一定价策略和差异化定价策略。

**A. Uniform Pricing**

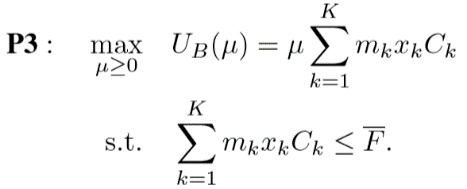
µ = µ1= … = µK， 则目标函数U\_k是l\_k的线性分段函数，

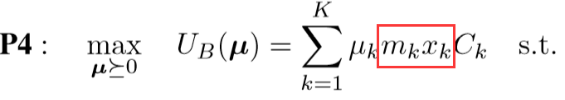
Problem P2 的最优解为:



如果用户k的CPU频率F\_k≤1/µ，则user k倾向于将m\_k bit数据加载到边缘云，否则将所有bits留给本地计算。

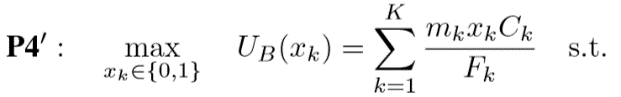
P1改写为P3：



**B. Differentiated Pricing**

Problem P1 can be rewritten as ：

x\_k = 1 ， 。

x\_k = 0 ， and earns no revenue.

P4相当于P4′：

P4′实际上是一个二进制背包问题。应用动态规划[15]解决上述二元背包问题。

**评估：**

**实验参数：**

B = 1 MHz, N\_0 = -174 dBm/Hz, h\_k in [-50,-30] dBm, F\_k selected from the set{0.1,0.2,…,1}GHz, C\_k ∈ [500,1500] cycles/bit, R\_k ∈ [100,500] KB, p\_k = 0.1 W, P\_{B,k} = 1 W, F = 6×109 cycles/slot, α\_k = 0.2, f\_C = 100 GHz.

**实验结果：**

**比较：**两种定价方案 + 所有输入数据都在用户处进行本地计算的方案

**指标：**计算容量对平均延迟和收入的影响

用户数量对平均延迟和收入的影响

**附：**

**Stackelberg博弈（斯塔克伯格博弈）的基本思想**：

Stackelberg Game是一个两阶段的完全信息动态博弈，博弈的time是序贯的。

主要思想是双方都是根据对方可能的策略来选择自己的策略以保证自己在对方策略下的利益最大化，从而达到纳什均衡。在该博弈模型中，先作出决策的一方被称为leader，在leader之后，剩余的players根据leader的决策进行决策，被称为followers，然后leader再根据followers的决策对自己的决策进行调整，如此往复，直到达到纳什均衡。

<https://en.wikipedia.org/wiki/Stackelberg_competition>