Assignment 5

12541018 赵钊

Part 1

1.

决策变量:

$x_{jt} \in \{0, 1\}$，对于每个任务 $j \in J$ 和每个服务器 $t \in 1, \ldots, m$, 若任务 j 分配给服务器 t，则 $x_{jt} = 1$，否则为 0。

$C_{max} \geq 0$ 是一个连续变量，代表完工时间即最大负载。

目标函数: $\text{Minimize } C_{max}$

约束条件:

(1) 每个任务必须被分配给恰好一台服务器：$\sum_{t=1}^{m} x_{jt} = 1, \forall j \in J$

(2) 定义每台服务器的负载：对于每台服务器 t，其负载 $\sum_j p_j x_{jt}$ 不超过 C_{max}。

$\sum_{j=1}^{n} p_j \cdot x_{jt} \leq C_{max}, \forall t \in \{1, \ldots, m\}$

2.

PARTITION: 给定一组正整数 $a_1, a_2, \ldots, a_k$，是否存在一个子集 $S \subseteq \{1, \ldots, k\}$ 使得 $\sum_{i \in S} a_i = \sum_{i \notin S} a_i$

划分问题是 NP-Complete 的

给定一个划分问题的实例 $\{a_1, \ldots, a_k\}$，构造一个实例：

任务数量 $n = k$, 每个任务 j 的处理时间 $p_j = a_j$, 服务器数量 $m = 2$, 目标完工时间 C_{max}

Claim: 存在一个调度使得 $C_{max} = \frac{1}{2} \sum_{j=1}^{k} p_j$ 当且仅当原始的划分问题有解

Proof:

（⇒） 如果存在一个分配使得 $C_{max} = \frac{1}{2} \sum p_j$，那么两台服务器的负载都必须恰好等于 $S/2$（其中 $S = \sum p_j$），否则如果有一台负载小于 $S/2$，另一台就会大于 $S/2$，这直接对应了划分问题的一个解

（⇐） 如果划分问题有解，即存在子集 S 使得 $\sum_{i \in S} a_i = \sum_{i \notin S} a_i = S/2$，那么我们可以将 S 中的任务分配给服务器1，其余任务分配给服务器2。这样两台服务器的负载都等于 $S/2$，因此 $C_{max} = S/2$

由于我们可以将任意划分问题实例在多项式时间内归约到 $m = 2$ 的负载均衡问题，并且划分问题是 NP-Complete 的，因此 $m = 2$ 的负载均衡决策问题是 NP-Hard 的。因此，其优化版本也是 NP-Hard 的。

Part 2

1.

1)
设总处理时间 $P_{total} = \sum_{j=1}^{n} p_j$
由于 m 台服务器共享这些工作，即使在最优调度中，负载最大的服务器也至少承担平均负载：

$L^* \geq \frac{1}{m} \sum_{j=1}^{n} p_j$

2)
设最大任务处理时间为 $p_{max} = \max_j p_j$
任何一台服务器必须处理分配给它的任务，且同一时间只能运行一个任务。因此，即使一台服务器只运行这一个最大任务，它的负载也至少是 p_{max}, 所以：$L^* \geq p_{max}$

2.

设贪心算法得到的完工时间为 L_{Greedy}

设服务器 t 是贪心调度中负载最大的服务器，即 $L_{Greedy} = L_t$

考虑在贪心算法中，最后一个被分配到服务器 t 的任务 j

在任务 j 被分配的时刻，服务器 t 的负载是当时所有服务器负载中最小的，因为贪心策略总是选择负载最小的服务器

设分配 j 之前，服务器 t 的负载为 L_t^{before}，其他所有服务器的负载在那一刻都至少是 L_t^{before}

因此，在分配 j 之前，所有服务器的总负载至少为：$m \cdot L_t^{before}$

加上 p_j 后的总负载为：$m \cdot L_t^{before} + p_j \leq \sum_{i=1}^{n} p_i$

于是有：$L_t^{before} \leq \frac{1}{m} \sum_{i=1}^{n} p_i - \frac{p_j}{m}$

分配 j 后，服务器 t 的负载为：$L_{Greedy} = L_t^{before} + p_j$

结合上述：$L_{Greedy} \leq \frac{1}{m} \sum_{i=1}^{n} p_i - \frac{p_j}{m} + p_j = \frac{1}{m} \sum_{i=1}^{n} p_i + (1 - \frac{1}{m}) p_j$

利用下界：$\frac{1}{m} \sum p_i \leq L^*, p_j \leq L^*$ 且 $1 - \frac{1}{m} < 1$

代入得：$L_{Greedy} \leq L^* + L^* = 2L^*$

Part 3

DP 状态定义

令：$dp[i][x] = True\ or\ False$

表示是否存在一种方式，从前 i 个任务中选取一部分分配给服务器1，使得服务器1的总负载恰好为 x

$i \in \{0, 1, \ldots, n\}, x \in \{0, 1, \ldots, S\}$

递推关系

对处理时间为 p_i 的任务 i:

将任务 i 分配给服务器2：$dp[i][x] = dp[i-1][x]$

将任务 i 分配给服务器1：$dp[i][x] = dp[i-1][x-p_i]$

初始条件

$dp[0][0] = True, dp[0][x] = False(x > 0)$

伪代码

输入: n 个任务的处理时间 $p[1..n]$，总时间 $S = sum(p)$
输出: 最小完工时间 C_{max}

```
dp[0..S] = False
dp[0] = True

for i = 1 to n:
    for x = S down to p[i]:
        if dp[x - p[i]] is True:
            dp[x] = True

best_diff = S
for x = 0 to S:
    if dp[x] is True:
        T1 = x
        T2 = S - x
        C_candidate = max(T1, T2)
        if C_candidate < best_diff + S/2:
            best_C = C_candidate

return best_C
```

复杂度分析

时间复杂度：外层循环 n 次，内层循环至多 S 次。因此时间复杂度为：$O(n \cdot S)$, 其中 $S = \sum p_j$

空间复杂度：使用一维数组时，空间复杂度为 $O(S)$

Part 4

1.

数据结构：数组 `assignment[1..n]` 记录任务分配的服务器，数组 `load[1..m]` 记录各服务器负载

邻域大小：
一个任务可从当前服务器移到其他 $m-1$ 台服务器，共有：$|N(S)| = n \cdot (m-1)$

2.

只有从当前负载最大的服务器（关键机器）移出任务才可能降低完工时间，从非关键机器移出任务无法立即改进目标值

优化策略：仅评估从关键机器移出任务到其他服务器的移动。原始每轮评估 $O(nm)$ 次移动，优化后只需评估 $O(c \cdot n_{critical} \cdot m)$ 次，其中 c 是关键机器数通常很小，减少了大量计算量