

Algorithm-HW6

🕒 Created	@2025年12月28日 20:13
📖 Class	Fundamentals of Computer Science

讲义第2题

解空间为 $n!$ 种可能的排列，可以用一个排列树表示，解向量为 $X = (x_1, x_2, \dots, x_n)$ ，其中 x_i 表示第 i 个任务分配给第 x_i 个人。设置当前已分配任务的费用总和 `current_cost = 0`，当前最优总费用 `best_cost = ∞`。从排列树的根节点（第1个任务）开始，尝试将其分配给第 j 个人 ($j = 1, \dots, n$)。在每一层搜索时，利用**约束函数**检查该分配是否合法，利用**限界函数**检查当前路径是否可能产生比 `best_cost` 更小的解。当搜索到达叶子节点（即 n 个任务分配完毕）且 `current_cost < best_cost` 时，更新 `best_cost = current_cost`。撤销当前的分配选择（回退到上一层），尝试其他可能的分支。

约束函数： $x_i \notin \{x_1, x_2, \dots, x_{i-1}\}$

限界函数： $current_cost + lower_bound(remaining_tasks) < best_cost$

讲义第3题

解空间为一棵高度为 n 的 k 叉树。每一个层级代表一个任务 i ，每一个分支代表将该任务分配给第 j 台机器 ($1 \leq j \leq k$)。创建一个数组 `sum[k]`，用于记录每台机器当前的累计工作时间，初始值均为 0。设置当前最短完成时间 `best_time = ∞`。从第 1 个任务开始，依次尝试将其分配给 k 台机器中的每一台。将任务 i 分配给机器 j 后，更新 `sum[j] = sum[j] + t_i`。在搜索过程中，计算当前所有机器中完成时间最长的那台机器的时间 `current_max = max(sum)`。如果 `current_max >= best_time`，则停止向下搜索，回溯。当搜索到叶子节点（即 n 个任务全部分配完毕）时，如果 `current_max < best_time`，则更新 `best_time = current_max`。撤销当前分配，令 `sum[j] = sum[j] - t_i`，尝试将任务分配给下一台机器。

约束函数：通常**没有硬性的可行性约束**。

限界函数： $\max_{1 \leq j \leq k} \{sum_j\} \geq best_time$

讲义第5题

解空间是一棵高度为 n 的 m 叉树。每一层 i 代表一个部件，每个分支 j 代表选择第 j 个供应商。设定当前总价格 `cp = 0`，当前总重量 `cw = 0`。设定当前最小重量 `best_w =`

∞ 。设定价格限制为 c 。从第 1 个部件开始，依次尝试从 m 个供应商中购买该部件。选择第 i 个部件的供应商 j 后，更新 $cp = cp + c_{ij}$ 且 $cw = cw + w_{ij}$ 。在每一层搜索时，利用**约束函数**检查价格是否超标，利用**限界函数**检查当前重量是否已经超过已知最优解。当搜索到叶子节点 ($i > n$) 时，说明已完成所有部件的选择。此时的 cw 即为一个可行解，更新 $best_w = cw$ 。撤销当前选择，令 $cp = cp - c_{ij}$ ， $cw = cw - w_{ij}$ ，尝试下一个供应商分支。

约束函数： $cp + c_{ij} \leq c$

限界函数： $cw + w_{ij} < best_w$