21307077 凌国明 嵌入式系统作业2

问题描述

设有18个进程，目前有11个cpu，每个进程在不同cpu上的运行时间如下表（具体见time.mat）

耗时	进程1	进程2	进程3	进程4	.....	进程18
Cpu1	17	19	3	19	.....	19
Cpu2	16	20	14	1	.....	17
Cpu3	14	7	20	1	.....	4
Cpu4	3	10	20	7	.....	6
.....	.....	.....	.....	.....	.....	.....
Cpu11	11	6	15	4	.....	11

对应的能耗如下表（具体见cost.mat）

功耗	进程1	进程2	进程3	进程4	.....	进程18
Cpu1	17	16	13	8	.....	4
Cpu2	5	4	5	9	.....	15
Cpu3	5	3	6	7	.....	11
Cpu4	11	5	10	13	.....	15
.....	.....	.....	.....	.....	.....	.....
Cpu11	8	12	6	6	.....	1

请根据所学建立对应的数学模型，并利用matlab中的ILP相关函数进行求解以下问题

1. 根据给定的表格求最小功耗的分配方案，要求每个CPU最多只能运行2个进程；
2. 根据给定的表格求最大运行总时间的分配方案，要求每个CPU最多只能运行2个进程；
3. 在时间和功耗代价相当的情况下，考虑兼顾2者最好的分配方案，每个CPU最多只能运行2个进程；
4. 根据表格建立运行时间最均衡的分配方案的相关数学模型对应求解式，除每个CPU只有一个外，无约束条件，不需要进行求解

Matlab 中的 ILP 函数

$$min\ f^T x$$

$$s.t. \begin{cases} x(intcon)\ are\ integers \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub \end{cases}$$

$f, x, b, beq, lb, ub, intcon$是向量； A和Aeq是矩阵

数学建模

x_{ij} 的建模

天然地，考虑设置一个 $11 * 18$ 的分配矩阵 x，使得 x_{ij} 表示第 j 个进程是否在第 i 个 cpu 上运行。$x_{ij} = 1$ 表示第 j 个进程在第 i 个 cpu 上运行，$x_{ij} \neq 1$ 表示第 j 个进程不在第 i 个 cpu 上运行。

ILP 函数中，f要求是一个向量，那么我们将 x 展平成向量，$x_{ij} = f_{(i-1)*18+j}$

A 和 b 的建模

ILP 函数中，A 和 b 表示矩阵形式的不等式约束。在我们的问题中，不等式约束表现在每个 cpu 最多运行两个进程，那么根据以上 x_{ij} 的建模，我们可以得出 A 是以下形式的，$11 * 198$ 维的矩阵

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 1 & 1 & \cdots & 0 & 0 & 0 \\ & & & & & \vdots & & & & & \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 1 & 1 \end{bmatrix}$$

```
A = zeros(11, 11 * 18); % 不等式约束矩阵
for i = 1:11
    A(i, (i-1) * 18 + 1 : i * 18) = ones(1, 18);
end
```

b 则是 $11 * 1$ 大小的，每个值为 2 的矩阵

Aeq 和 beq 的建模

ILP 函数中，Aeq 和 beq 表示矩阵形式的等式约束。在我们的问题中，等式约束表现在每个进程都要在 cpu 上执行，且只能在一个 cpu 上执行，那么根据以上 x_{ij} 的建模，我们可以得出 A 是以下形式的，$18 * 198$ 维的矩阵：

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 1 & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 1 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 1 & \cdots & 0 & 0 & 1 \\ & \ddots & & & & \ddots & & & & \ddots & \end{bmatrix}$$

```
Aeq = repmat(eye(18, 18), 1, 11); % 等式约束矩阵
```

以上矩阵本质上是 11 个 $18 * 18$ 的 eye 矩阵的拼接
beq 则是 $18 * 1$ 大小的，每个值为 1 的矩阵

问题一

```
cost = load("cost.mat").cost;

% 创建问题的线性整数规划模型
f = cost'; % 目标函数，最小化总功耗
f = f(:); % 将f转换为列向量

A = zeros(11, 11 * 18); % 不等式约束矩阵
for i = 1:11
    A(i, (i-1) * 18 + 1 : i * 18) = ones(1, 18);
end
b = 2 * ones(11, 1);

Aeq = repmat(eye(18, 18), 1, 11); % 等式约束矩阵
beq = ones(18, 1);

lb = zeros(11 * 18, 1); % 变量下界，所有元素都为0
ub = ones(11 * 18, 1); % 变量上界，所有元素都为1

% 使用intlinprog函数求解线性整数规划问题
[x, fval] = intlinprog(f, 1:11 * 18, A, b, Aeq, beq, lb, ub);
% 将结果转换为矩阵形式
allocation_matrix = reshape(x, 11, 18);

% 输出最小功耗和分配方案
disp(['最小功耗: ', num2str(fval)]);
disp('分配方案: ');
disp(allocation_matrix);
save("allocation_1.mat", "allocation_matrix");
```

```
最小功耗：44
分配方案：
     0     0     0     0     0     0     1     0     0     0     0     1     0     0     0     0     0     0
     0     0     0     0     1     0     0     0     0     0     1     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     1     0     0     1     0     1     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0
     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0
     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1
```

问题二

```
time = load("time.mat").time;

% 创建问题的线性整数规划模型

f = -time'; % 目标函数，最大化时间
f = f(:); % 将f转换为列向量

A = zeros(11, 11 * 18); % 不等式约束矩阵
for i = 1:11
    A(i, (i-1) * 18 + 1 : i * 18) = ones(1, 18);
end
b = 2 * ones(11, 1);

Aeq = repmat(eye(18, 18), 1, 11); % 等式约束矩阵
beq = ones(18, 1);

lb = zeros(11 * 18, 1); % 变量下界，所有元素都为0
ub = ones(11 * 18, 1); % 变量上界，所有元素都为1

% 使用intlinprog函数求解线性整数规划问题
[x, fval] = intlinprog(f, 1:11 * 18, A, b, Aeq, beq, lb, ub);
% 将结果转换为矩阵形式
allocation_matrix = reshape(x, 11, 18);

% 输出最小功耗和分配方案
disp(['最大时间：', num2str(-fval)]);
disp('分配方案：');
disp(allocation_matrix);
save("allocation_1.mat", "allocation_matrix");
```

```
最大时间：333
分配方案：
     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0
     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1
     0     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     1     0     0     0     0     1     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     0
     0     1     0     0     0     0     0     0     0     0     0     1     1     0     0     0     0     0
     0     0     0     1     0     0     0     0     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     1     0     1     0     0     0     0     0     0     0     0     0     0
```

问题三

```
time = load("time.mat").time;
cost = load("cost.mat").cost;

% 创建问题的线性整数规划模型

f = time' + cost'; % 目标函数，最小化功耗和时间的和
f = f(:); % 将f转换为列向量

A = zeros(11, 11 * 18); % 不等式约束矩阵
for i = 1:11
    A(i, (i-1) * 18 + 1 : i * 18) = ones(1, 18);
end
b = 2 * ones(11, 1);

Aeq = repmat(eye(18, 18), 1, 11); % 等式约束矩阵
beq = ones(18, 1);

lb = zeros(11 * 18, 1); % 变量下界，所有元素都为0
ub = ones(11 * 18, 1); % 变量上界，所有元素都为1

% 使用intlinprog函数求解线性整数规划问题
[x, fval] = intlinprog(f, 1:11 * 18, A, b, Aeq, beq, lb, ub);
% 将结果转换为矩阵形式
allocation_matrix = reshape(x, 11, 18);

% 输出最小功耗和分配方案
disp(['优化函数最小值: ', num2str(fval)]);
disp('分配方案: ');
disp(allocation_matrix);
save("allocation_1.mat", "allocation_matrix");
```

```
优化函数最小值：162
分配方案：
     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     0
     0     0     0     1     0     0     1     0     0     0     1     1     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0     1
     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0     0
     0     1     0     1     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     1     0     0     0     0     0     1     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0
     0     0     1     0     0     0     0     0     1     0     0     0     0     0     0     0     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1     0     0     0
     0     0     0     0     0     0     0     0     1     0     0     0     0     0     0     1     0     0
     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     0     1
```

问题四

天然地，考虑设置一个 $11 * 18$ 的分配矩阵 x，使得 x_{ij} 表示第 j 个进程是否在第 i 个 cpu 上运行。$x_{ij} = 1$ 表示第 j 个进程在第 i 个 cpu 上运行，$x_{ij} \neq 1$ 表示第 j 个进程不在第 i 个 cpu 上运行。

每个进程都要被一个 cpu 执行，可以表示为以下约束条件

$$\sum_{i=1}^{11} x_{ij} = 1$$

要求运行时间最均衡，可以转化为最小化最大运行时间，则优化函数如下

$$\max_{i} \sum_{j=1}^{18} x_{ij} \cdot t_{ij}$$

使得以上优化函数取得最小值，就可以求得解。