# Algorithmic Methods for Mathematical Models (AMMM)

## Lab Session 3 – More on Mixed Integer Linear Programs

In this third session we are going to slightly complicate our example of assigning tasks to computers in a data center.

In this case, we will assume that computers consist of a number of cores, whereas tasks consist of a number of threads. Each task must be assigned to a single computer, and threads must be assigned to a single core provided that it has enough capacity.

### 1. Problem statement

The *P3* problem can be formally stated as follows:

*Given:*

- The set $T$ of tasks. Each task $t$ consists of a set of threads $H(t)$. For each thread $h$ the amount of requested resources $r_h$ is specified.
- The set $C$ of computers. Each computer $c$ consists of a set of cores $K(c)$. All cores of each computer $c$ have the same capacity $r_c$.

*Find* the assignment of tasks to computers and threads to cores subject to the following constraints:

- Each thread is assigned to a single core.
- Each task is assigned to a single computer, i.e. all the threads of a task are assigned to cores of the same computer.
- The capacity of each core cannot be exceeded.

with the *objective* to minimize the highest loaded computer.

### 2. MILP formulation

The *P3* problem can be modeled as a Mixed Integer Linear Program. To this end, the following sets and parameters are defined:

| | |
|---|---|
| $T$ | Set of tasks, index $t$. |
| $C$ | Set of computers, index $c$. |
| $H$ | Set of threads, index $h$. |
| $H(t)$ | Subset of threads belonging to task $t$. |
| $K$ | Set of cores, index $k$. |
| $K(c)$ | Set of cores in computer $c$. |
| $r_h$ | Resources requested by thread $h$. |
| $r_c$ | Capacity of each core $k$ in computer $c$. |

The following decision variable is also defined:

$x_{tc}$ binary. Equal to 1 if task $t$ is served from computer $c$; 0 otherwise.
$x_{hk}$ binary. Equal to 1 if thread $h$ is served from core $k$; 0 otherwise.
$z$ positive real with percentage of load of the highest loaded computer.

Finally, the MILP model for the P3 problem is as follows:

$$\text{minimize} \quad z \tag{1}$$

subject to:

$$\sum_{k \in K} x_{hk} = 1 \quad \forall h \in H \tag{2}$$

$$\sum_{h \in H(t)} \sum_{k \in K(c)} x_{hk} = |H(t)| \cdot x_{tc} \quad \forall t \in T, c \in C \tag{3}$$

$$\sum_{h \in H} r_h \cdot x_{hk} \le r_c \quad \forall c \in C, k \in K(c) \tag{4}$$

$$z \ge \frac{1}{|K(c)| \cdot r_c} \cdot \sum_{h \in H} \sum_{k \in K(c)} r_h \cdot x_{hk} \quad \forall c \in C \tag{5}$$

## 3. Tasks

In pairs, do the following tasks and prepare a lab report.

a) Implement the *P3* model in OPL and solve it using CPLEX with the following data file.

**Table 1 Data file**

```
nTasks=4;
nThreads=8;
nCPUs=3;
nCores=9;

rh=[261.27 261.27 560.89 560.89 310.51 310.51 105.8 105.8];

rc=[505.67 503.68 701.78];

CK=[
    [1  1  1  0  0  0  0  0  0]
    [0  0  0  1  1  1  0  0  0]
    [0  0  0  0  0  0  1  1  1]
];


TH=[
    [1  1  0  0  0  0  0  0]
    [0  0  1  1  0  0  0  0]
    [0  0  0  0  1  1  0  0]
    [0  0  0  0  0  0  1  1]
];
```

Where matrix *CK* defines the cores belonging to each computer and matrix *TH* defines the threads belonging to each task.

NOTE: You will need some preprocessing to obtain the number of threads of each task and the number of cores in each computer.

b) Generate instances of increasing size with the instance generator script and use the *P3* model to solve them.

c) Modify the *P3* model to maximize the number of computers with all their cores empty (*P3a*).

d) Compare both models (*P3* and *P3a*) in terms of number of variables, constraints and execution time for the generated instances.

   Recall that you can tune the *epgap* param to control when cplex stops, e.g., use this in your main OPL code:

```
cplex.epgap = 0.01
```