Algorithmic Methods for Mathematical Models (AMMM)

Lab Session 1 - Linear Programs

In this first session we are going to learn how to implement Linear Program (LP) models using the IBM ILOG OPL language and solve them using IBM CPLEX. You can find complete information regarding both OPL and CPLEX in the *IBM ILOG CPLEX Optimization Studio Information Center*.

To start with this first session, let us state the problem we are going to solve. The problem is about assigning a set of tasks (processes) to a set of computers in a data center. We denote this problem as *P1*.

The following points impose constraints on the assignment:

- On the one hand, each process is requesting some amount of processing resources.
- On the other hand, each computer has a CPU with a known processing capacity.

We define the *load* of a computer as the amount of processing resources assigned to that computer with respect to its processing capacity. Since the power consumption of the computers increases heavily with the load, the objective consists in distributing the tasks among all the computers so to minimize the load of the computer running at the highest load.

1. Problem statement

The *P1* problem can be formally stated as follows:

Given:

- The set T of tasks. For each task t the amount of processing resources requested r_t is specified.
- The set C of computers. For each computer c the available processing capacity r_c is specified.

Find the assignment of tasks to computers subject to the following constraints:

- Each task is assigned to some computers so that the entire requested processing load is served. Note that this means that a single task can consume processing capacity of several computers.
- The processing capacity of each computer cannot be exceeded.

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

2. LP formulation

The *P1* problem can be modeled as a Linear Program. To this end, the following sets and parameters are defined:

T Set of tasks, index t.
C Set of computers, index c.
r_t Resources requested by task t.
r_c Available capacity of computer c.

The following decision variables are also defined:

x_{tc} positive real with the ratio of resources requested by task t that are served from computer c.
z positive real with the load of the highest loaded computer.

Note that the variable z stores the load of the highest loaded computer, that is:

$$z = \max_{c \in C} \left\{ \frac{1}{r_c} \cdot \sum_{t \in T} r_t \cdot x_{tc} \right\} \tag{1}$$

Finally, the LP model for the *P1* problem is as follows:

$$\text{minimize} \quad z \tag{2}$$

subject to:

$$\sum_{c \in C} x_{tc} = 1 \quad \forall t \in T \tag{3}$$

$$\sum_{t \in T} r_t \cdot x_{tc} \leq r_c \quad \forall c \in C \tag{4}$$

$$z \geq \frac{1}{r_c} \cdot \sum_{t \in T} r_t \cdot x_{tc} \quad \forall c \in C \tag{5}$$

3. Simple implementation in OPL

To implement the model for *P1* in OPL, create a new project in the ILOG Optimization Studio. Two files are needed: one for the model and another one for the data.

The model file contains the data definition, variables, expressions and constraints. It also may contain processing scripting blocks for pre and post processing data and results, among others.

The data file contains specific data for a problem instance. Note that by splitting data and model into two different files we can run the same model for different instances represented by different data files.

To run a model, the ILOG Optimization Studio uses configurations which can contain both, the model and the data files.

The next code implements the *P1* model defined in equations (2)-(5).

Table 1 Model file

```
int nTasks=...;
int nCPUs=...;

range T=1..nTasks;
range C=1..nCPUs;

float rt[t in T]=...;
float rc[c in C]=...;

dvar float+ x_tc[t in T, c in C];
dvar float+ z;

// Objective
minimize z;

subject to{

// Constraint 1
forall(t in T)
   sum(c in C) x_tc[t,c] == 1;

// Constraint 2
forall(c in C)
   sum(t in T) rt[t]* x_tc[t,c] <= rc[c];

// Constraint 3
forall(c in C)
   z >= (1/rc[c])*sum(t in T) rt[t]* x_tc[t,c];

}
```

A simple pre-processing script to write the amount of load that is requested to the datacenter is:

Table 2 Pre-processing block

```
execute {
   var totalLoad=0;

   for (var t=1;t<=nTasks;t++)
      totalLoad += rt[t];

   writeln("Total load "+ totalLoad);
};
```

Similarly, a simple post-processing script to print the percentage of load of each computer is:

Table 3 Post-processing block

```
execute {

   for (var c=1;c<=nCPUs;c++) {
      var load=0;
      for (var t=1;t<=nTasks;t++)
         load+=(rt[t]* x_tc[t][c]);
      load = (1/rc[c])*load;
      writeln("CPU " + c + " loaded at " + 100*load + "%");
   }
};
```

Note that pre-processing and post-processing blocks must be placed in the model file before and after, respectively, of the objective function.

Finally, an example of data file is:

Table 4 Data file

```
nTasks=4;
nCPUs=3;

rt=[261.27 560.89 310.51 105.80];
rc=[505.67 503.68 701.78];
```

4. Advanced implementation in OPL

Many times it is useful to be able to do some more advanced implementation to, for instance, solve several models in sequence passing data from one model to the next. To do that, we need to define a main file similarly to the next one.

Table 5 Main file

```
main {

    var src = new IloOplModelSource("P1.mod");
    var def = new IloOplModelDefinition(src);
    var cplex = new IloCplex();
    var model = new IloOplModel(def,cplex);
    var data = new IloOplDataSource("P1.dat");
    model.addDataSource(data);
    model.generate();

    cplex.epgap=0.01;

    if (cplex.solve()) {
      writeln("Max load " + cplex.getObjValue() + "%");

      for (var c=1;c<=model.nCPUs;c++) {
         var load=0;
```

```
          for (var t=1;t<=model.nTasks;t++)
             load+=(model.rt[t]* model.x_tc[t][c]);
          load = (1/model.rc[c])*load;
          writeln("CPU " + c + " loaded at " + load + "%");
       }
    }
    else {
        writeln("Not solution found");
    }

    model.end();
    data.end();
    def.end();
    cplex.end();
    src.end();
};
```

To run this script, let us create a new configuration in the ILOG Optimization Studio. In contrast to the first configuration which contained both, the model and the data files, in this configuration only the main file must be placed.

5. Tasks

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

a) Explain the *P1* model in eq. (2)-(5). Specifically, define each of the constraints.

b) Explain how equation (1) is implemented in the model *P1*. Specifically, why z is guaranteed to be equal to the load of the highest loaded computer?

c) Implement the *P1* model in OPL following the steps in section 3 and solve it using CPLEX with the provided input data.

d) Write a pre-processing block to check whether computers have enough total capacity to serve the resources requested by all the tasks.

e) Implement the *P1* model in OPL following the steps in section 4 and solve it using CPLEX witth the provided input data.

f) Analyze the optimal solution obtained in e): Would it be possible to reduce the capacity of all computers uniformly by the same percentage? If so, which percentage?