Multi-Machine Assignment Problem

Problem definition: Given a number of tasks to schedule and a number of machines, each task can be done on any of these machines with specified duration and cost. In addition, each task has a release date and due date, meaning the task must be performed during period between these two dates.

The objective is to minimise total cost by choosing the machines to process these tasks. To load this project, search for sched_mmasp in the ILOG CPLEX Studio,

Now we explain the mod file in detail:

define the machine range.

```
// Number of Machines (Packing + Manufacturing)
int nbMachines = ...;
range Machines = 1..nbMachines;
```

define the job range

```
// Number of Jobs
int nbJobs = ...;
range Jobs = 1..nbJobs;
```

define the duration, cost for each pair of task and machine, release and due dates for each task.

```
int duration[Jobs,Machines] = ...;
int cost    [Jobs,Machines] =...;
int release [Jobs] = ...;
int due     [Jobs] = ...;
```

define the interval variable for each task in the range of release and due date.

define the interval variables for each pair of task and machine with task duration. For each task, we choose one of these interval variable meaning one machine will perform this task.

```
dvar interval task[j in Jobs] in release[j]..due[j];
dvar interval opttask[j in Jobs][m in Machines] optional size duration[j][m];
```

define sequence of tasks on each machine

```
dvar sequence tool[m in Machines] in all(j in Jobs) opttask[j][m];
```

minimise total costs - this is achieved by using presenceOf function on the optional variables.

The function return1 if the variable is present.

```
// Minimize the total processing cost (24)
minimize
  sum(j in Jobs, m in Machines) cost[j][m] * presenceOf(opttask[j][m]);
```

define constraints that each task interval is associated with one of the machine/task intervals.

```
 // Each job needs one unary resource of the alternative set s (28
forall(j in Jobs)
          alternative(task[j], all(m in Machines) opttask[j][m]);
```

define constraints that for each machine, tasks on this machine cannot overlap

```
// No overlap on machines
 forall(m in Machines)
          noOverlap(tool[m]);
```