Open-Shop Scheduling

“This problem can be described as follows: a finite set of operations has to be processed on a given set of machines. Each operation has a specific processing time during which it may not be interrupted. Operations are grouped in jobs, so that each operation belongs to exactly one job. Furthermore, each operation requires exactly one machine for processing.”

To load this project, search for sched_openshop in the ILOG CPLEX Studio

Define job and machine ranges.

```
int nbJobs = ...;
int nbMchs = ...;
range Jobs = 0..nbJobs-1;
range Mchs = 0..nbMchs-1;
```

Load operation durations for each job on each machine.

```
int OpDurations[j in Jobs][m in Mchs] = ...;
```

Define decision variables: interval variable for each operation of each job; sequence of operations on each machine; sequence of operations for each job

```
dvar interval itvs[j in Jobs][m in Mchs]  size OpDurations[j][m];
dvar sequence mchs[m in Mchs] in all(j in Jobs) itvs[j][m];
dvar sequence jobs[j in Jobs] in all(m in Mchs) itvs[j][m];
```

Define CP search limit.

```
execute {
                cp.param.FailLimit = 10000;}
```

Objective is to minimise max completion time of all operations

```
minimize max(j in Jobs, m in Mchs) endOf(itvs[j][m]);
```

Define constraints that no operations for each job should overlap – meaning job can only appear on one machine at a time. Note that if no transition between adjacent operations is constrained, we can just use noOverlap on the set of interval variables; no need to define this sequence variable. Sequence variables are most useful when setup time is modelled between adjacent tasks.

```
forall (j in Jobs)
  noOverlap(jobs[j]);
```

or

```
forall (j in Jobs)
    noOverlap(all(m in Mchs)itvs[j][m]);
```

Define constraints that no operations on each machine should overlap

```
forall (m in Mchs)
  noOverlap(mchs[m]);
```