Explanation of the set of instances

===================================

There are four directories. One for small instances and another for large instances. Then there are two directories with small and large calibration instances, respectively.

There are 1620 small instances

There are 1440 large instances

There are 162 small calibration instances

There are 144 large calibration instances

There are 7 controlled factors in the generation of the instances which are in the name of the instance:

Number of jobs (n)

Number of stages (i)

Number of identical parallel machines per stage (mi)

Tardiness factor (T)

Range of due dates factor (R)

Width of due date window (W)

Replica number (r)

Therefore, the name of the instance is:

Instance\_n\_i\_mi\_T\_R\_W\_Repr.txt

For example

Instance\_50\_5\_3\_0,2\_0,2\_10\_Rep0.txt is an instance with 50 jobs, 5 stages, 3 machines per stage 0.2 Tardiness factor, 0.2 Range of due dates factor, 10 width of due window and replica 0.

a) small instances

The following factor combinations are tested (all combinations)

n: 10, 15 and 20

i: 2,3 and 4

mi: 2 and 3

T: 0.2, 0.4 and 0.6

R: 0.2, 0.6 and 1.0

W: 10 and 20

r: 5 replicates

In total there are 3\*3\*2\*3\*3\*2\*5=1620 small instances

b) large instances

n: 50, 100, 150 and 200

i: 5 and 10

mi: 3 and 5

T: 0.2, 0.4 and 0.6

R: 0.2, 0.6 and 1.0

W: 10 and 20

r: 5 replicates

In total there are 4\*2\*2\*3\*3\*2\*5=1440 small instances

c) small calibration instances

We generate a 10% of the 1620 small instances choosing at random values for n, i, mi, T, R and W from the small instances factors. Different seed is used. In total 162 small calibration instances.

d) large calibration instances

The same procedure from the small calibration instances is applied to generate a 10% of the 1440 large instances (144 instances)

Explanation of the contents of an instance file

===============================================

An instance file looks like this:

HFSDDW

10 4 2

0 72 1 81

0 77 1 56

0 21 1 56

0 90 1 44

0 97 1 28

0 29 1 47

0 63 1 47

0 98 1 4

0 86 1 99

0 68 1 32

LBCmax: 367

RELDUE

-1 317 2 5

-1 319 3 9

-1 330 6 7

-1 259 1 4

-1 308 4 9

-1 295 5 7

-1 326 2 3

-1 307 9 8

-1 297 4 5

-1 263 2 2

DDW

301 333

293 345

307 353

246 272

302 314

268 322

297 355

282 332

270 324

245 281

The first line is the tag identifying the problem. For all this benchmark is HFSDDW

The second line contains three integers, number of jobs n, total number of machines (n\*mi) and the number of stages i

Then we have n rows where each row has 2\*i columns. The first row is the number of the first stage (0 indexed) followed by the processing time of the first job on the first stage... then the second stage (1) and the processing time of the job in the second stage and so on.

Then we have a tag LBCmax followed by the best calculated lower bound for the makespan in this problem.

Then we have a tag RELDUE indicating that next we have n lines with four columns. The first column is the release date (-1) for this problem as it is not used. Then we have the due date of the first job, then the weight for the earliness and the weight for the tardiness.

Lastly we have a tag DDW indicating that next we have n lines with two columns. The first column is d- or the opening of the due window and the second is d+ or the closing or the due window. The due date is contained within these two values

Explanation of the txt files with the best solutions

====================================================

All the best solutions are in the same file. For each instance we have something like this:

Instance: 0 Jobs: 10 Stages: 2 UpperBound: 613

8 0 3 4 5 7 1 2 6 9

0 8 3 4 5 1 7 2 6 9

Instances are numbered from 0 to the total of instances-1 on each benchmark. As an indication we have the number of jobs and the number of instances. Upper bound is the best solution known for the total weighted earliness tardiness deviation from the due window.

Below we have one line per stage indicating the job sequence at each that stage (jobs are 0 indexed). It is the order in which jobs have to be launched at the stage. The First Available Machine rule is to be applied to know to which machine each job is assigned to.

To know the completion times for each job one has to decode the solution and insert idle time as indicated in the paper.