**Polynomial schedule for jobs with identical process time:**

s Size of each job

p Process time of each job

S Total number of parallel machines.

a set of *n* jobs, where each job has a due date , size , and process time .

If we can ignore the machines, since there just no jobs that can fit this size, and this number of machines will always remain unused, hence we assume that S is dividable by s. Additionally, since all jobs have identical size and identical process time, we can do discretization by s for size and p for time, without changing optimal solution. In the discretized problem , which has polynomial solution in [1].

**References**

[1] Dessouky, M. I., Lageweg, B. J., Lenstra, J. K., & Velde, S. L. (1990). Scheduling identical jobs on uniform parallel machines. *Statistica Neerlandica*, *44*(3), 115-123.

**Definitions:**

s Size of each job

p Process time of each job

S Total number of parallel machines.

a set of *n* jobs, where each job has a due date , size , and process time .

Compact schedule - A schedule with no idle times, or idle machines.

**Algorithm:**

1. Order all the jobs according to EDD rule, denote this set by
2. For each job
   1. Try to add the job to the lower machine index, and then to lower possible time. For example, assume that s=1 and p=1, if the machines 1-2, at time 0 are already used, we will try to add the new job at machine 3, and only when we use all our machines at time 0, we will start placing jobs at time 1.
      1. If we cannot add the job add it to set
      2. Else add the job to set , with the added position.

**Theorem:** A schedule O generated by the above algorithm, will produce optimal results for jobs of identical sizes and identical process times.

**Proof:**

We denote , as any partial schedule of the above algorithm, containing the first k jobs from the set . It is important to notice that any schedule is a compact schedule.

At step 2.1 for each job k following properties are true:

Additionally if is late:

1. Schedule has the maximum number of early jobs possible before time , because it is a compact schedule.

If job is late:

We have three options:

* Replace job with some job from : because of property 1, we still will have at least one job late.
* Replace job with some job from : because of property 2, we cannot exchange with any job from , and having better schedule.
* Create a better schedule from the job jobs: Because of 3 we know that there is no better schedule possible until time

Hence we know that job must be late, and the schedule is optimal for the first jobs. This is true .