CSC 373 - Algorithm Design, Analysis, and Complexity **Summer 2016**

Lalla Mouatadid

2-Approximation Minimum Makespan Scheduling

The first approximation technique we have seen was through rounding and relaxation of IPs and LPs. In this lecture, we'll see an example of a greedy algorithm that guarantees a constant factor approximation ratio.

The problem we are interested is the **Minimum Makespan Scheduling Problem**, defined as follows:

Suppose we have n jobs each of which take time t_i to process, and m identical machines on which we schedule the jobs. Jobs cannot be split between machines. For a given scheduling, let A_j be the set of jobs assigned to machine j. Let $T_j = \sum\limits_{i \in A_j} t_i$ be the load of machine j. The *minimum makespan scheduling problem* asks for an assignment of jobs to machines that minimizes the **makespan**, where the makespan is defined as the maximum load over all machines (i.e. $\max\limits_j T_j$).

The greedy algorithm we came up with in class was to sort the jobs so that $t_1 \geq t_2 \geq ... \geq t_n$, and iteratively allocate the next job to the machine with the least load:

Algorithm 1 Greedy Approximation Algorithm for Job Scheduling on identical Machines

```
A_j ← ∅, T_j ← 0, ∀j
for i = 1 ... n do
    j ← min_k T_k
    A_j = A_j ∪ {i}
    T_j = T_j + t_i
end for
```

Claim: The algorithm above is a 2-approximation.

Proof. Let T^* denote the optimal makespan. We came up with the following two facts:

$$T^* \geq \max_i t_i \tag{1}$$

$$T^* \geq \frac{1}{m} \sum_j^m T_j \tag{2}$$

$$= \frac{1}{m} \sum_i^n t_i \tag{3}$$

Now consider machine j with maximum load T_j. Let i be the last job scheduled on machine j. When i was scheduled, j had the smallest load, so j must have had load smaller than the average load. Then,

$$\begin{aligned} T_j &= (T_j - t_j) + t_j \\ &\leq \frac{1}{m} \sum_i^n t_i + \max_i t_i \\ &\leq 2T^* \end{aligned}$$

Where the first inequality follows from Facts (1) and (3) above. The analysis above shows that the algorithm is a 2-approximation algorithm. □