Max Wisniewski

Dozent : Panos Giannopoulos

Task 3: (Makespan - Problem)

In this exercise we should prove the following Lemma on the Makespan-Problem.

Lemma 2.8: *For any input to the problem of minimizing the makespan on identical parallel maschines for which the processing requirements of each job is more than one-third the optimal makespan, the longest processing time rule computes an optimal schedule.*

Proof:

Let m be the number of machines, and $J = \{1, ..., n\}$ the jobs, where $p_i, i \in J$ is the processing-time. c_i, $i \in J$ is the time at which Job i is finished. Let $C_{\max} = \max_{j \in J} c_j$ the value of the optimal solution and $C^*_{\max}$ the computationtime of the longest processing rule. Let $e : \{1..m\} \to 2^J$ be the function, that states, which machine executes which jobs.

First obeserve, that in the optimal solution each machine can execute at most two jobs. Next we can assume, that the in the optimal solution the machines that executes two jobs, the longes one will allways be executed first.

Now we show, that we can modify the optimal solution s.t. we have a solution, that satisfies the longest processing time rule.

The Idea of the following algorithm is, that we look at two machines. If the second job of the first machine has a bigger executiontime than the first job of the seconde machine. (Assuming the $p_{first} > p_{seconde}$). Than we put the second job of the first machine as the first job of the second machine and take the second job of the second machine and make it to the second job of the first machine. After this change at least these two machines satisfy the longest processing time rule.

```
WHILE (∃ i,j : p_{e(i)[1]} > p_{e(j)[0]} DO
    save := e(j)[1];
    e(j)[1] := e(j)[0];
    e(j)[0] := e(i)[1];
    e(i)[1] := save;
```

After this execution on the specific optimal schedule there hold two claims. The new schedule is $C'_{\max}$.

Lemma 1. $C'_{\max} \leq C_{\max}$

Lemma 2. *If you take the reordered schedule from lemma 1 you can exchange the following jobs on the machines and the maximal computation time is not worsen. If i is the last job on one machine and j is the last job on another machine, and $p_i > p_j \wedge c_i - p_i > c_j - p_j$ holds, the two jobs are exchanged from one machine to another.*

Lemma 3. $C'_{\max}$ *after lemma 2 satisfies the longest processing time rule.*

Proof 1. We show, that in after each iteration $C_{\max}$ is monotonically decreasing. Let (k_1, k_2) and (t_1, t_2) be the jobs handeled by the two choosen processes. We assume, that the first one is greater, namely $k_1 \geq k_2 > t_1 \geq t_2$. The complete runtime of the first process is $k_1 + k_2$ and the seconde one runs in $t_1 + t_2$. After the exchange the first process handels $k_1 + t_2$ and the second one handels are $k_2 + t_1$. As we can see, both processes have with the underlaying mentioned relation, that $k_1 + t_2 \leq k_1 + k_2$ and $k_2 + t_1 \leq k_1 + t_2$.

We conclude, that after each iteration the solution gets better.

Proof 2. For the computation time holds $C_{\max} \geq c_i$ and $C_{\max} \geq c_j$. Now we substitute the conditions for the change in the formula $c_{\max} \geq c_i - p_i + p_i > c_i - p_i + p_j$. The time on this machine is lessen. An for other machine $C_{\max} \geq c_j - p_j + p_j > c_j - p_j + p_i$ holds, because $c_j - p_j < c_i - p_i$.

Proof 3. We assume there is a job s that is longer than a job, that started earlier we name t. Because every job starts either at point zero or as the second the job s has to be the second one and the job t has to be the first one.

But under these circumstances the algorithm would have exchanged the the jobs of these machines. Therefore there could not be two jobs as assumed and the lemma holds.

Considering these three lemmas, we conclude from lemma 1 and 2 that the reordering will not change the schedule, because otherwise there would be a smaller solution than the minimum.

With lemma 3 we conclude, that the longest processing time rule will infact delifer an optimal solution. That is because the longest processing time rule algorithm will deliver always the same result in the value . So this solution will always have the optimal computation time.