Embedded Systems

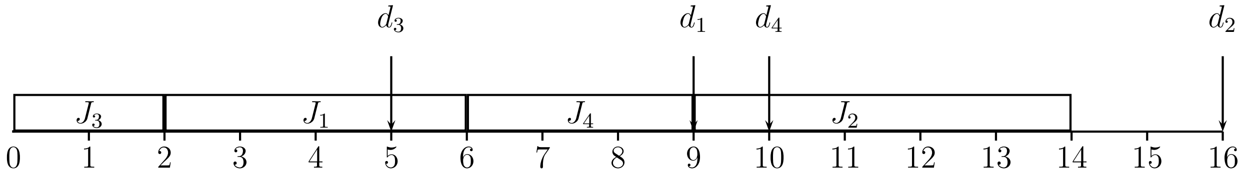
Assignment 8

Jan Hendrik Dithmar
2031259

Pascal Gwosdek
2505221

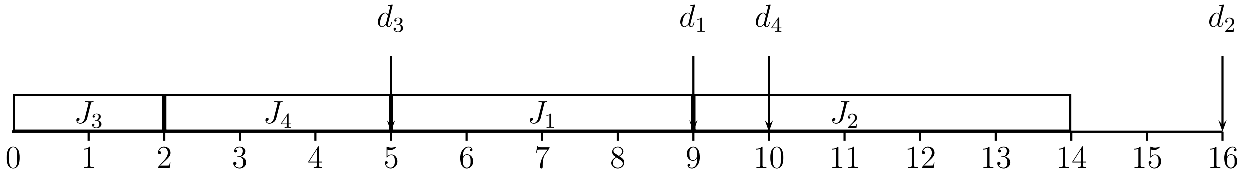
8.1 EDD-scheduling

Here is the schedule:



$$\left.\begin{array}{l} L_3 = 3 \\ L_1 = 3 \\ L_4 = 1 \\ L_2 = 2 \end{array}\right\} \Rightarrow L_{max} = 3$$

If we swap J_1 and J_4, we get:



$$\left.\begin{array}{l} L_3 = 3 \\ L_4 = 5 \\ L_1 = 0 \\ L_2 = 2 \end{array}\right\} \Rightarrow L_{max} = 5$$

The maximal lateness is increased. This means that the first schedule is better.
Now let's have a look if we modify the last schedule in such way that we additionally swap J_3 and J_4. Then you can easily see, that L_4 increases to 7 and L_3 decreases to 0, which leads to a maximal lateness $L_{max} = 7$. The problem is now even worser.
As the schedule above shows, it is possible to schedule the given set of tasks with respect to the criteria $1 \mid sync \mid L_{max}$.

8.2 Scheduling

a. First observation: If $a_i = 0 \quad \forall i = 1, ..., n$ we have for $\overline{R}$:

$$\overline{R} = \frac{1}{n} \sum_{i=1}^{n} f_i$$

To minimize $\overline{R}$, we have to ensure small finishing times $\forall J_i, i = 1, ..., n$. Due to the fact that there are no deadlines mentioned, we assume that the deadlines of the jobs never hurt us. We describe later in the algorithm what to do if we have to consider the deadlines.
To get a very small result for the sum, we have to ensure that each summand is as small as possible. Due to this, we have to take the jobs with the smallest C_i's first.

Algorithm:

I Build a FIFO-queue Q

1. If we don't have to consider the deadlines, set the deadlines $d_i \quad \forall i = 1, ..., n$ to $\sum_{i=1}^{n} C_i$.
   If the deadlines are important, don't change the d_i's; just take the given values.
2. Build a dependence graph, s.t. the sum of the computation times along a path is smaller or equal than the deadline of the job you want to add to the graph right now. If you get stuck, mark the path as bad.
3. Delete all the bad paths from the graph.
4. Compute the sum of the finishing times for each path in the dependence graph.
5. Pick the path with the smallest sum of finishing times and build Q by going along this path.

II Schedule according to Q.

b. The algorithm is not deterministic, because the building of the dependence graph is non-deterministic.

Proof:

- Let J_s be the job with the smallest computation time. Let J_t be the job with the computation time which is superior to the computation time of J_s. All other computation times are higher than the ones from J_s and J_t.
- Let $d_s = d_t = C_s + C_t$ and $f_j = \sum_{i=3}^{n} C_i \quad \forall j = 3, ..., n$ wlog.
- Let σ be a schedule which starts with J_s, followed by J_t.

$$\Rightarrow \overline{R} = \frac{1}{n} \cdot \left(f_s + f_t + \sum_{i=3}^{n} f_i \right)$$

- Let σ^* be a schedule which starts with J_t, followed by J_s.

$$\Rightarrow \overline{R}^* = \frac{1}{n} \cdot \left(f_t^* + f_s^* + \sum_{i=3}^{n} f_i^* \right)$$

- For the finishing times we have the following:

$$f_s = C_s \qquad f_t = f_s + C_t$$

  and

$$f_t^* = C_t \qquad f_s^* = f_t^* + C_s$$

- Looking at that we obviously get the following:

$$f_s + f_t < f_s^* + f_t^*$$

- Since all other computation times are higher that C_t, we have

$$\sum_{i=3}^{n} f_i \leqslant \sum_{i=3}^{n} f_i^*$$

$\Rightarrow$ The sum – and obviously the average response time – is bigger after swapping J_s and J_t in that way that J_t is executed before J_s.

$\Rightarrow$ The algorithm is optimal.