Networked Embedded Systems Lab
Prof. Marco Zimmerling

Introduction to Embedded Systems – WS 2022/23

Sample Solution to Exercise 3: Aperiodic Scheduling

Task 1: Earliest Deadline Due

Check whether the Earliest Deadline Due (EDD) algorithm produces a feasible schedule for the following task set, given that all tasks are synchronous and arrive at time $t = 0$.

	J_1	J_2	J_3	J_4
C_i	3	6	2	4
D_i	8	15	3	11

Solution to Task 1:

EDD (Earliest Deadline Due) is a scheduling algorithm that minimizes the maximum lateness. The *Jackson's rule* says that given a set of n independent tasks, any algorithm that executes the tasks in order of non-decreasing deadlines is optimal with respect to minimizing the maximum lateness.

Assumptions about the task set for applying EDD:

- tasks have same arrival times (synchronous arrivals)
- tasks are independent

EDD is non-preemptive. The EDD produces a feasible schedule given in Figure 1.

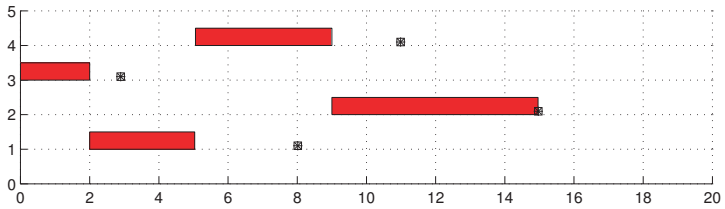


Figure 1: EDD schedule.

The tasks are scheduled in the order of non-decreasing deadlines as follows: J_3, J_1, J_4, J_2.

Task 2: Latest Deadline First

Given the precedence graph in Figure 2 and the following table of task execution times and deadlines, determine a Latest Deadline First (LDF) schedule. Is this schedule feasible?

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
C_i	3	4	2	3	3	2	2	1
D_i	5	8	11	15	12	18	19	20

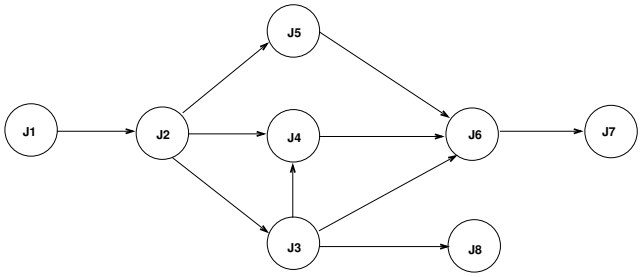


Figure 2: Precedence graph.

Solution to Task 2:

LDF (Latest Deadline First) is a scheduling algorithm that minimizes the maximum lateness. It assumes synchronous task activations and is non-preemptive.

The algorithm to produce an LDF schedule proceeds in two stages: firstly, a precedence graph is constructed. Going from tail to head: among the tasks without successors or whose successors have been all selected, LDF selects the tasks with the latest deadline to be scheduled last. At runtime, tasks are extracted from the head of the queue: the first task inserted in the queue will be executed last.

The queue of tasks scheduled with LDF is as follows: $J_1 - J_2 - J_3 - J_5 - J_4 - J_6 - J_7 - J_8$. The LDF schedule is presented in Figure 3. On Figure 4 you can find a sketch of how the LDF algorithm proceeds.

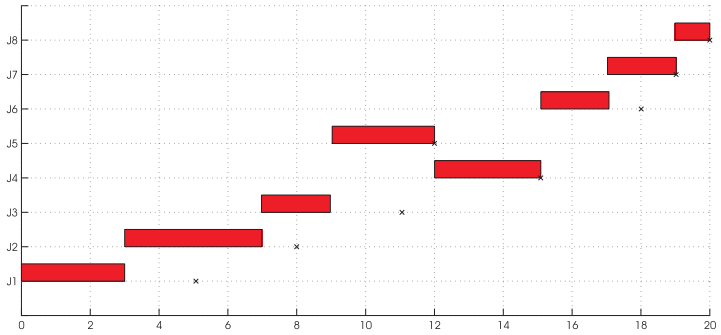


Figure 3: LDF schedule.

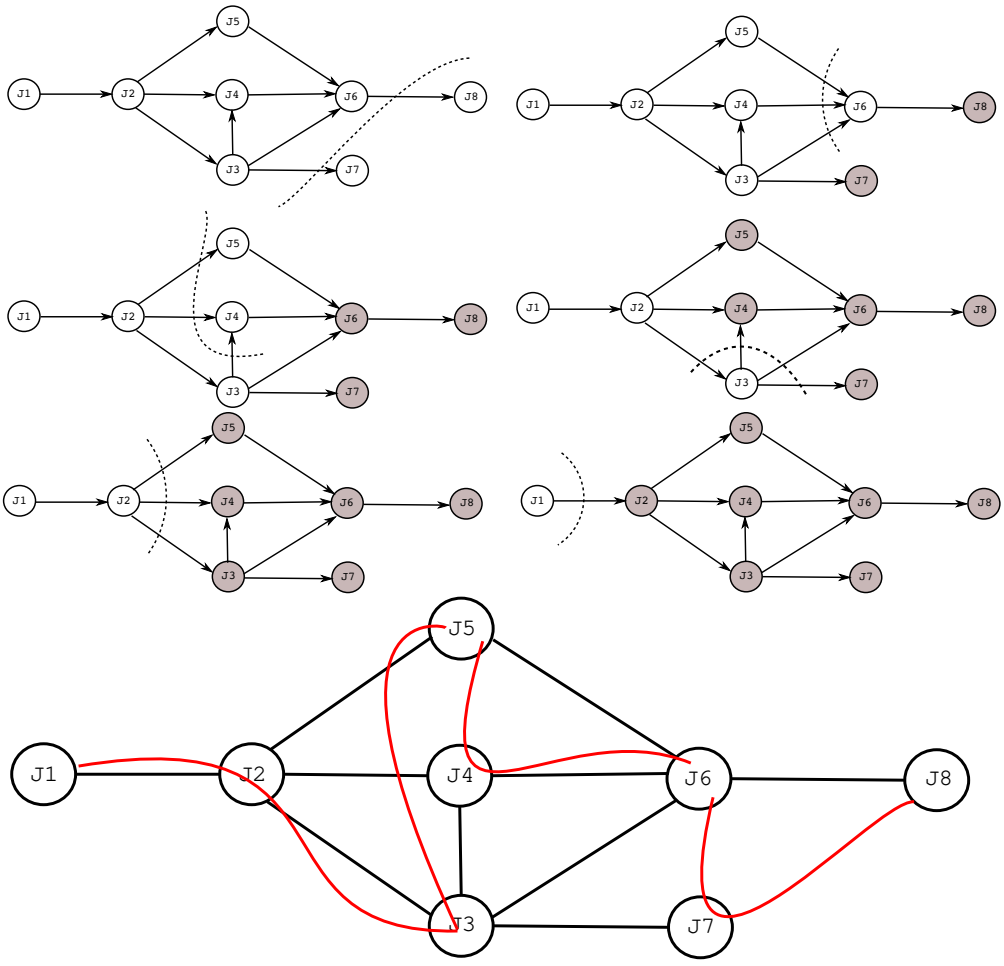


Figure 4: The LDF algorithm proceeds as depicted (figures left to right)

Task 3: Earliest Deadline First

In the following table, five tasks with arrival times, execution times and deadlines are given.

	J_1	J_2	J_3	J_4	J_5
a_i	0	2	0	8	13
C_i	3	1	6	2	3
d_i	16	7	8	11	18

(1) Determine a Earliest Deadline First (EDF) schedule. Is this schedule feasible?

(2) At time $t = 3$, a new task J_x arrives with execution time $C_x = 2$ and deadline $d_x = 10$. Can you guarantee the schedulability of the task set with this new task?

Solution to Task 3:

EDF (Earliest Deadline First) is optimal in the sense of feasibility (it minimizes the maximum lateness under following assumptions: scheduling algorithm is preemptive, the tasks are independent, and may have arbitrary arrival times). The *Horn's rule* says that given a set of n independent tasks with arbitrary arrival times any algorithm that at any instant executes the task with earliest absolute deadlines among the ready tasks is optimal with respect to the maximum lateness.

(1) The EDF schedule is feasible, and the respective schedule is shown in the Figure 5.

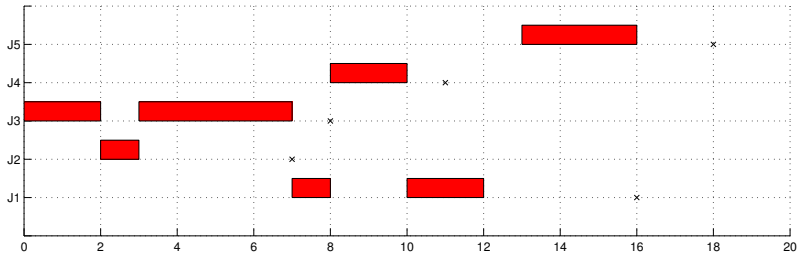


Figure 5: EDF schedule

(2) The arrival of a new task J_x at time point $t = 3$ still maintains the schedule feasible. We can check this by computing ahead the finishing times of the tasks (with respect to the interesting time points). In the analysis, we consider the tasks in order of increasing deadlines.
We assume an on-line scenario, i.e. in the schedulability test we consider only tasks currently present in the system. The test is performed every time a new task arrives.
From the schedule in the preceding sub-problem, we know that before task J_X arrives at time $t = 3$ we have:
- tasks J_1, J_2 and J_3 are feasible
- task J_2 finishes before its deadline at $t = 3$

That is, up to $t = 3$ all active tasks in the system are feasible.
At $t = 3$ we have three tasks in the system: J_1, J_3 and the new task, J_x. For them we perform the schedulability test:
Put $f_0 = t = 3$ (We have absolute deadlines in the problem specification)
Task J_3: $f_1 = f_0 + c_3(3) = 3 + 4 = 7 \leq 8$ (ok)
Task J_x: $f_2 = f_1 + c_x(3) = 7 + 2 = 9 \leq 10$ (ok)
Task J_1: $f_3 = f_2 + c_1(3) = 9 + 3 = 12 \leq 16$ (ok)
Thus, at $t = 3$ all tasks in the system are feasible.

The next task to arrive is J_4. It arrives at $t = 8$. At $t = 8$, we have thee active tasks: J_x, J_4 and J_1. The schedulability test at $t = 8$ proceeds as follows:

Put $f_0 = t = 8$

Task J_x: $f_1 = f_0 + c_x(8) = 8 + 1 = 9 \leq 10$ (ok)

Task J_4: $f_2 = f_1 + c_4(8) = 9 + 2 = 11 \leq 11$ (ok)

Task J_1: $f_3 = f_2 + c_1(8) = 11 + 3 = 14 \leq 16$ (ok)

Thus, at $t = 8$ all tasks in the system are feasible.

Finally, task J_5 arrives at $t = 13$. At $t = 13$, we have two active tasks: J_1 and J_5. The schedulability test at $t = 13$ proceeds as follows:

Put $f_0 = t = 13$

Task J_1: $f_1 = f_0 + c_1(13) = 13 + 1 = 14 \leq 16$ (ok)

Task J_5: $f_2 = f_1 + c_5(13) = 14 + 3 = 17 \leq 18$ (ok)

Hence, the whole schedule is feasible. It is given in the figure below:

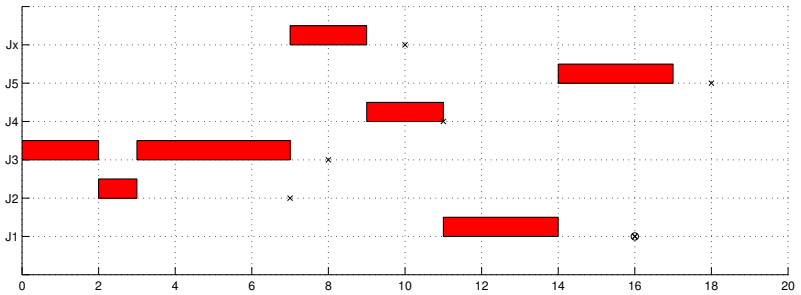


Figure 6: EDF schedule (with J_x)

Task 4: Earliest Deadline First – Star

Given are seven tasks A, B, C, D, E, F, G with following precedence constraints:

$$A \longrightarrow C, \quad B \longrightarrow C, \quad C \longrightarrow E, \quad D \longrightarrow F, \quad B \longrightarrow D, \quad C \longrightarrow F, \quad D \longrightarrow G$$

All tasks arrive at time $t_0 = 0$, have a common deadline $d = 20$ and the following execution times:

	A	B	C	D	E	F	G
C_i	3	2	4	3	2	5	1

(1) Construct the precedence graph for this task set. Then, modify the release times and deadlines so that EDF* can be used for its scheduling.

(2) Determine a resulting EDF* schedule. For this schedule, compute the average of all response times of the tasks.

(3) Assume the additional precedence constraint $E \longrightarrow A$. Is there still a feasible schedule for the above task set? Justify your answer.

Solution to Task 4:

EDF* schedules tasks with precedence constraints. Release time and deadline of individual tasks are modified such that all the precedence constraints are satisfied. By doing this, the scheduling problem is transformed into a problem without precedence constraints, which can then be handled by a "normal" EDF scheduler.

There are several points to take into consideration while modifying tasks' release times and deadlines:

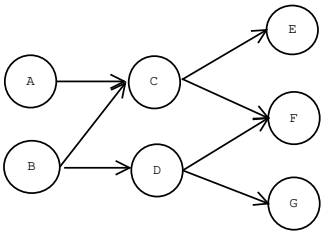
Release time modification:

- task must start the execution not earlier than its release time
- task must start the execution not earlier than the minimum finishing time of its predecessors

Deadline modification:

- task must finish within its deadline
- task must finish not later than the maximum start time of its successors

(1) The precedence graph for the given task set is:



The modified release times and deadlines are:

	A	B	C	D	E	F	G
r_i^*	0	0	3	2	7	7	5
d_i^*	11	11	15	15	20	20	20

In particular, the release time modification proceeds as follows:

for initial nodes $\{A, B\}$ set $r_A^* = r_A$, $r_B^* = r_B$

$r_C^* = \max\{r_c, \max\{r_A^* + C_A, r_B^* + C_B\} = \max\{0, \max\{3, 2\}\} = 3$

$r_D^* = \max\{r_D, r_B^* + C_B\} = \max\{0, 0 + 2\} = 2$

$r_F^* = \max\{r_F, r_C^* + C_C, r_D^* + C_D\} = \max\{0, 3 + 4, 2 + 3\} = 7$

$r_E^* = \max\{r_E, r_C^* + C_C\} = \max 0, 3 + 4 = 7$

$r_G^* = \max\{r_G, r_D^* + C_D\} = \max 0, 2 + 3 = 5$

The deadline modification proceeds as follows:

for the terminal nodes $\{E, F, G\}$ set $d_E^* = d_F^* = d_G^* = 20$

$d_C^* = \min\{d_C, d_E^* - C_E, d_F^* - C_F\} = \min\{20, 20 - 2, 20 - 5\} = 15$

$d_D^* = 15$

$d_A^* = 11$

$d_B^* = 11$

(2) One resulting EDF* schedule, based on the modified release times and deadlines, is presented below.

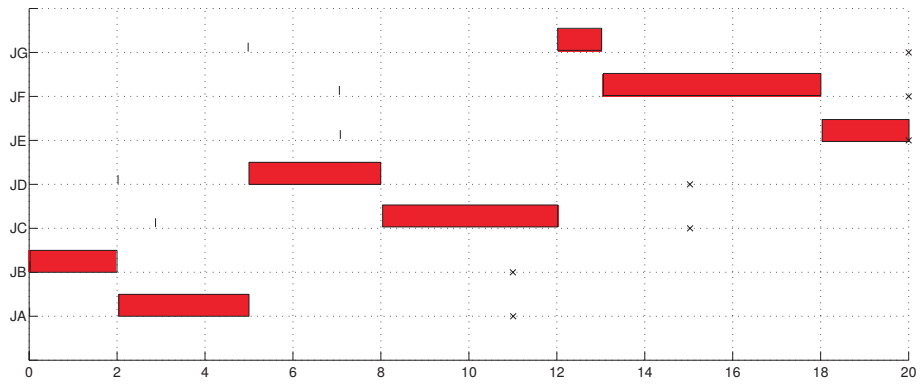


Figure 7: EDF* schedule

The response time of a task is defined as the difference between its finishing time and release time. Therefore, the average of all response times for the above schedule is computed as follows:

$$\overline{t_r} = \frac{1}{7}\sum_{i=1}^{7}(f_i - r_i) = \frac{5+2+12+8+20+18+13}{7} = 11.14$$

Note that for the computation of a task's response time we consider its original (not modified) release time. Note also that there are different correct schedules, and the corresponding response times may be different.

(3) No, the task set is no longer schedulable. Under the new conditions, the constraints among tasks A, C and E introduce a cycle in the precedence graph. As a result, none of the three tasks can be executed as first and therefore, no feasible schedule exists.

Task 5: Earliest Deadline First – Star

Given are eight aperiodic tasks, J_1 to J_8, with their arrival times, deadlines, and execution times as shown in the table below. Task precedence constraints are as follows:

$$J_1 \to J_2,\ J_2 \to J_3,\ J_3 \to J_4,\ J_5 \to J_6,\ J_6 \to J_7,\ J_6 \to J_8,\ J_2 \to J_7,\ J_7 \to J_4,\ J_8 \to J_7.$$

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
r_i	0	3	4	0	0	2	0	2
d_i	3	8	15	15	10	10	10	11
C_i	1	3	3	3	1	1	2	1

(1) Construct the precedence graph.

(2) Using the EDF* algorithm, modify the arrival times and deadlines of the tasks in order to make the tasks schedulable under EDF. Enter the modified arrival times and deadlines in Table 1.

Table 1: Modified arrival times and deadlines

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
r_i^*								
d_i^*								
C_i	1	3	3	3	1	1	2	1

(3) Assume that the application is executed on a dual-core platform. At any time t, both cores execute the two ready tasks ($r_i^* \leq t$) with earliest deadlines (Note: A single task cannot be executed on two cores simultaneously). Using the arrival times and deadlines obtained in (2), construct an EDF schedule in Figure 8.

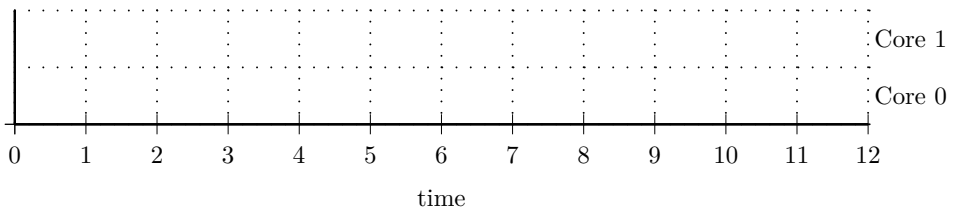
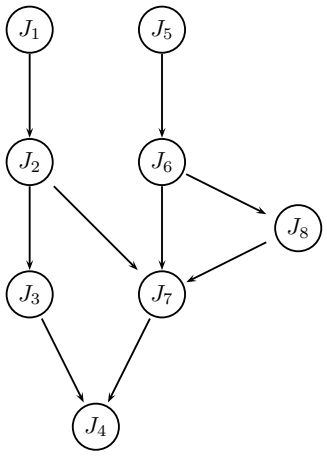


Figure 8: EDF schedule for part (3)

(4) Now assume that the application is executed on a quad-core platform with the same scheduling rule (4 cores execute the four ready tasks with earliest deadlines). Will executing on the quad-core platform reduce the completion time of the application? Justify your answer with an explanation.

Solution to Task 5:

(1) The precedence graph:



(2) The modified arrival times and deadlines:

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
r_i^*	0	3	6	9	0	2	6	3
d_i^*	3	8	12	15	6	7	10	8
C_i	1	3	3	3	1	1	2	1

(3) There are several solutions since the tasks can run either on core 0 or core 1, and one can be seen in Figure 9. Makespan of all solutions should be 12.

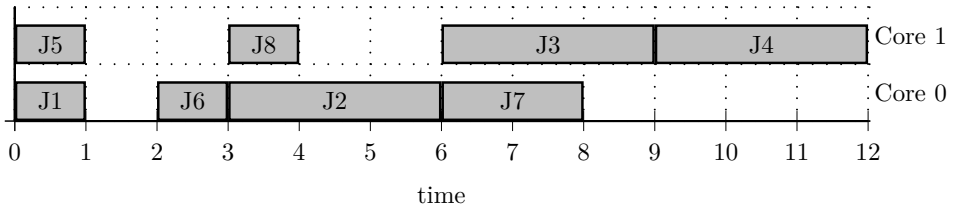


Figure 9: One solution for part (3)

(4) No. J_4 cannot be started earlier than time 9. Therefore, the minimum finish time of the application is 12 (the finish time for the dual core platform).