Aperiodic Scheduling

Earliest Deadline Due, Latest Deadline First, Earliest Deadline First

Exercise class 4

Presenter:
Jürgen Mattheis

In cooperation with:
Pascal Walter

Based on the lecture of:
Marco Zimmerling

December 6, 2022

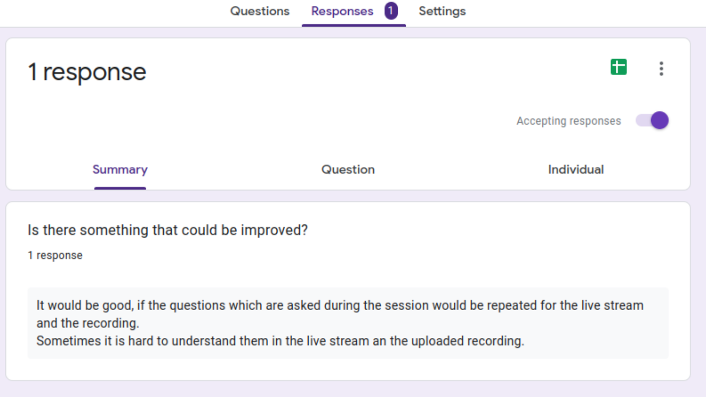
University of Freiburg, Chair for Embedded Systems

Gliederung

Organisation

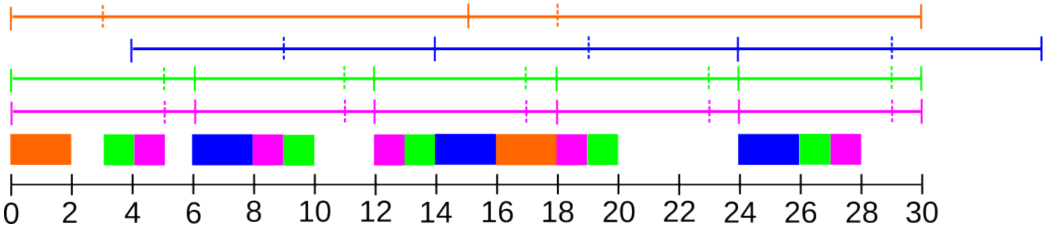
Organisation I

- feedback for me: `https://forms.gle/f3YN8EFrZ1vsfPoC6`



Organisation I

- Question: Periodic schedule for framesize $f = 2$ for task 3 of sheet 2



Overview Aperiodic Task Scheduling

Overview Aperiodic Task Scheduling

	Equal arrival times non preemptive	Arbitrary arrival times preemptive
Independent tasks	EDD (Jackson)	EDF (Horn)
Dependent tasks	LDF (Lawler)	EDF* (Chetto)

Overview Aperiodic Task Scheduling

- Lateness: $L_i = f_i - d_i$
- Maximum lateness: $L_{\max} = \max_i (f_i - d_i)$
  - this a a metric to compare schedules

Task 1

Task 1

Earliest Deadline Due

Task 1.1:

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

$(\forall J_i \in J : a_i = 0)$

Requirements 1.1:

- *non-preemptive*
- *tasks have same arrival times (synchronous arrivals)*
- *tasks are independent*
- *$min(D_i)$ for all remaining J_i (d_i if $\forall J_i \in J : a_i = c \wedge c \neq 0$)*
- *minimizing the maximum lateness*

Task 1

Earliest Deadline Due

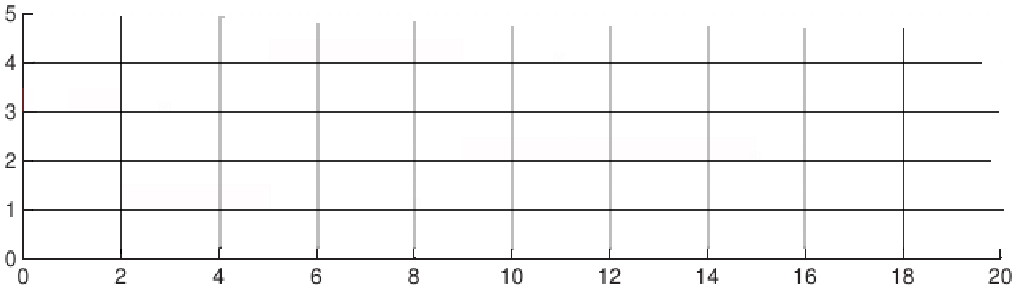


Figure 1: EDD schedule.

Task 1

Earliest Deadline Due

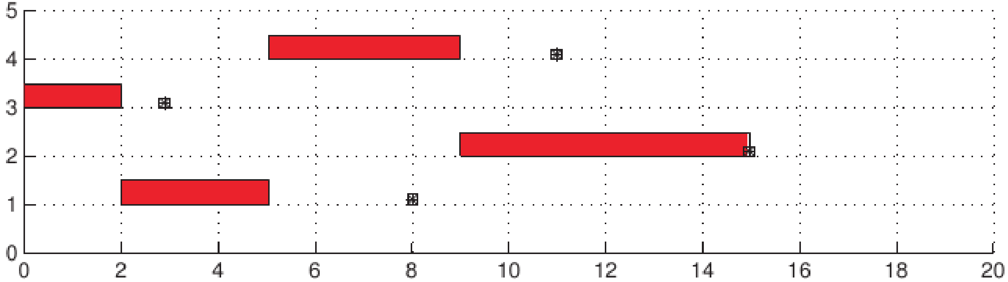


Figure 1: EDD schedule.

Task 2

Task 2

Latest Deadline First

Task 2.1:

	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

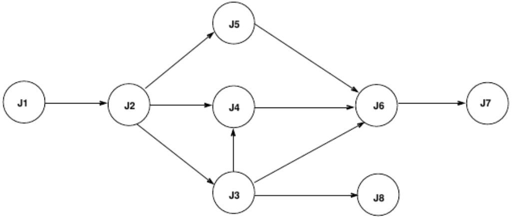


Figure 2: Precedence graph.

Task 2

Latest Deadline First

Requirements 2.1:

- *is non-preemptive*
- *synchronous task activations*
- *tasks are dependent, use precedence graph, going from tail to head*
- $max(D_i)$ $(d_i \text{ if } \forall J_i \in J : a_i = c \wedge c \neq 0)$ *for all tasks* J_i *without successors or whose successors have been all selected in the precedence graph inserted into the queue to be executed last*
- *at runtime, tasks are extracted from the head of the queue: the first task inserted in the queue will be executed last*
- *minimizes the maximum lateness*

Task 2

Latest Deadline First

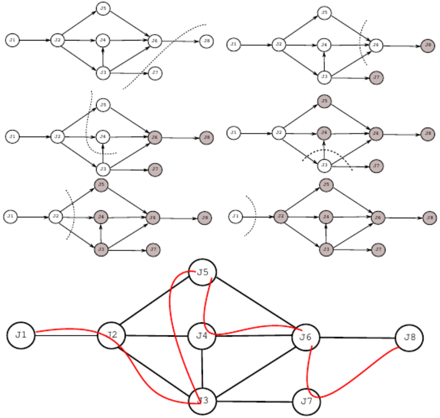


Figure 4: The LDF algorithm proceeds as depicted (figures left to right)

Task 2

Latest Deadline First

- queue of tasks: (, , , , , , ,)

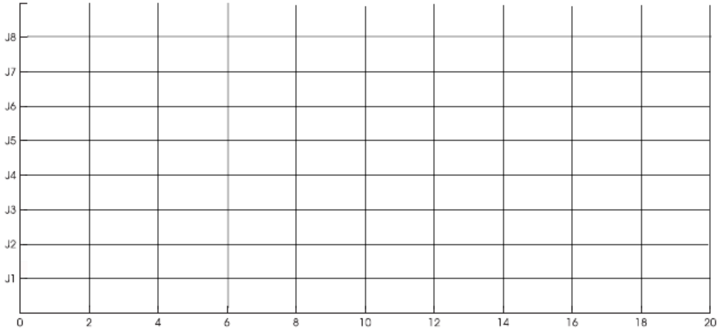


Figure 3: LDF schedule.

Task 2

Latest Deadline First

- queue of tasks: $(J_1, J_2, J_3, J_5, J_4, J_6, J_7, J_8)$

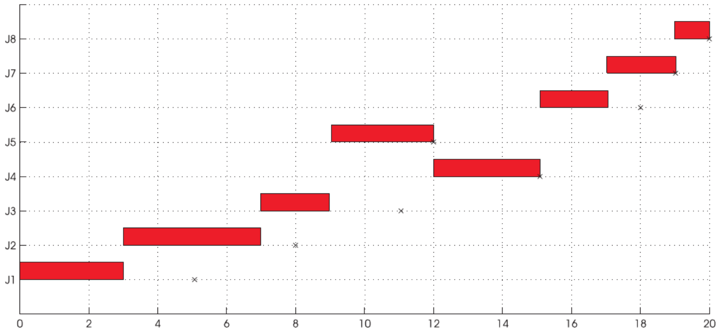


Figure 3: LDF schedule.

Task 3

Task 3

Earliest Deadline First

Task 3.1:

	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

Requirements 3.1:

- *is preemptive*
- *arbitrary arrival times*
- *the tasks are independent*
- *minimizes the maximum lateness*
- *$\min(d_i)$ for all remaning tasks J_i that have already arrived (are ready) and not finished every time the arrival time of a task is reached*

Task 3

Earliest Deadline First

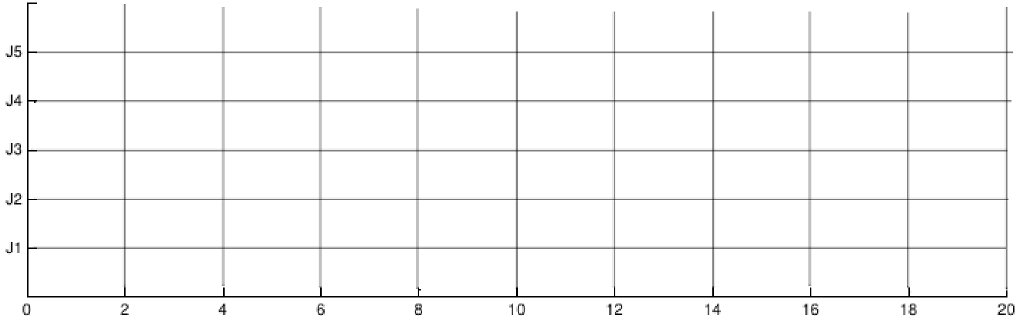


Figure 5: EDF schedule

Task 3

Earliest Deadline First

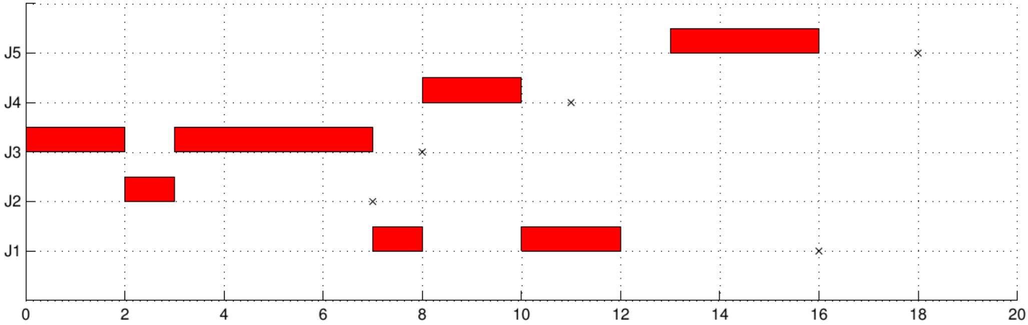


Figure 5: EDF schedule

Task 3

Earliest Deadline First

Task 3.2:

- at time $t = 3$, a new task J_x arrives with execution time $C_x = 2$ and deadline $d_x = 10$.
- still guarantee the schedulability of task set?

Requirements 3.2:

- *acceptance test:* $\forall i = 1, \ldots, n \quad t + \sum_{k=1}^{i} c_k(t) \leq d_i$
  - *n is the number of active tasks*

Task 3

Earlist Deadline First

Requirements 3.3:

```
Algorithm: EDF_guarantee (J, J_new)
{       J'=J∪{J_new};  /* ordered by deadline */
        t = current_time();
        f_0 = t;
        for (each J_i∈J') {
                f_i = f_i-1 + c_i(t);
                if (f_i > d_i) return(INFEASIBLE);
        }
        return(FEASIBLE);
}
```

Task 3 I

Earliest Deadline First

- check: with J_x schedule still feasible
- compute ahead the finishing times in consideration of J_x
- consider the tasks in order of increasing deadlines:

	J_2	J_3	J_X	J_4	J_1	J_5
a_i	2	0	3	8	0	13
C_i	1	6	2	2	3	3
d_i	7	8	10	11	16	18

- test performed every time a new task arrives among all active task

Task 3 II

Earliest Deadline First

- until $t = 3$ everything is feasible
  - at $t = 0$ we have active tasks (arrived but not finished): $\{J_1, J_3\}$
    - Put $t = 0$
    - Task $J_3 : f_1 = t + c_3(0) = 0 + 6 = 6 \leq 8$ ✓
    - Task $J_1 : f_2 = f_1 + c_1(0) = 6 + 3 = 9 \leq 16$ ✓
  - at $t = 2$ we have active tasks: $\{J_1, J_2, J_3\}$
    - Put $t = 2$
    - Task $J_2 : f_1 = t + c_3(2) = 2 + 1 = 3 \leq 7$ ✓

Task 3 III

Earliest Deadline First

- Task $J_3 : f_2 = f_1 + c_2(2) = 3 + 4 = 7 \leq 8$ ✓
- Task $J_1 : f_3 = f_2 + c_1(2) = 7 + 3 = 9 \leq 16$ ✓

- task J_2 finishes before its deadline at $t = 3$

- at $t = 3$ we have active tasks: $\{J_1, J_3, J_x\}$
  - Put $t = 3$
  - Task $J_3 : f_1 = t + c_3(3) = 3 + 4 = 7 \leq 8$ ✓
  - Task $J_x : f_2 = f_1 + c_x(3) = 7 + 2 = 9 \leq 10$ ✓
  - Task $J_1 : f_3 = f_2 + c_1(3) = 9 + 3 = 12 \leq 16$ ✓

Task 3 IV

Earliest Deadline First

- 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 three active tasks: $\{J_x, J_4, J_1\}$
  - Put $t = 8$
  - Task $J_x : f_1 = t + c_x(8) = 8 + 1 = 9 \leq 10$ ✓
  - Task $J_4 : f_2 = f_1 + c_4(8) = 9 + 2 = 11 \leq 11$ ✓
  - Task $J_1 : f_3 = f_2 + c_1(8) = 11 + 3 = 14 \leq 16$ ✓
  - Thus, at $t = 8$ all tasks in the system are feasible.

Task 3 V

Earliest Deadline First

- next task J_5 arrives at $t = 13$. At $t = 13$, we have two active tasks: $\{J_1, J_5\}$
  - Put $t = 13$
  - Task $J_1 : f_1 = t + c_1(13) = 13 + 1 = 14 \leq 16$ ✓
  - Task $J_5 : f_2 = f_1 + c_5(13) = 14 + 3 = 17 \leq 18$ ✓
  - hence, the whole schedule is feasible

Task 3

Earliest Deadline First

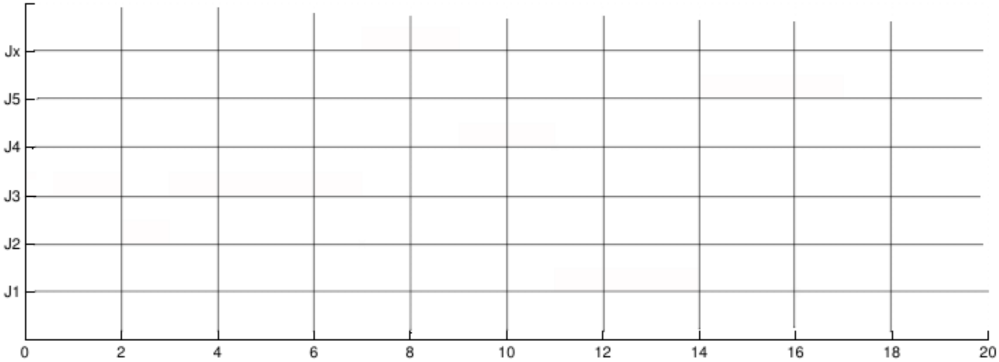


Figure 6: EDF schedule (with J_x)

Task 3

Earliest Deadline First

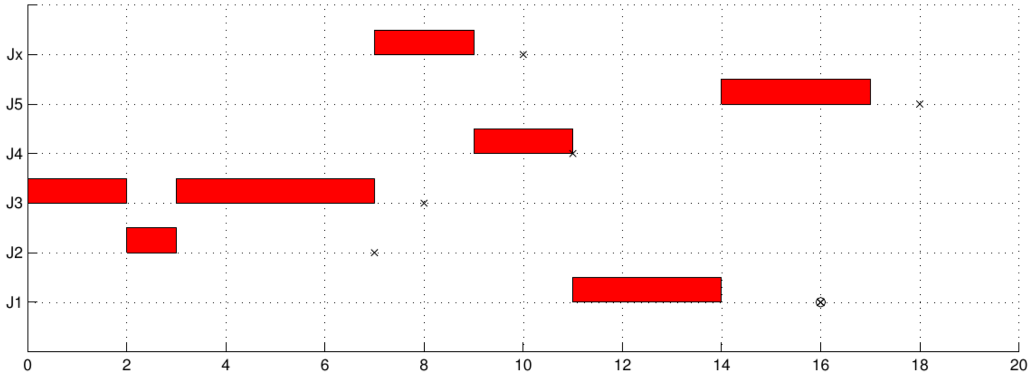


Figure 6: EDF schedule (with J_x)

Task 4

Task 4 I

EDF* - Example

Task 4.1:

Given tasks A, B, C, D, E, F, G with precedences $A \to C$, $B \to C$, $C \to E$, $D \to F$, $B \to D$, $C \to F$, $D \to G$.

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

We will now prepare the tasks for EDF*

Task 4

EDF*

Requirements 4.1:

- *preemptive*
- *arbitrary arrival times*
- *tasks are dependent*
- *release time and deadline of individual tasks are modified such that all the precedence constraints are satisfied*
- *scheduling problem is transformed into a problem without precedence constraints, which can then be handled by a "normal" EDF scheduler*

Task 4 I

EDF* - Transformation

- EDF* transforms the arrival time and deadline of every task in the following way:

  Deadline:

  1. Task must finish the execution time within its deadline: $f_i \leq d_i$
  2. Task must not finish the execution time later than the maximum start time of its successor(s): $f_i \leq d_j - C_j$

  $\rightarrow$ $d_i^* = min(d_i, min(d_j^* - C_j : J_i \rightarrow J_j))$

Task 4 II

EDF* - Transformation

Arrival time

1. Task must start the execution not earlier than its release time: $s_j \geq r_j$
2. Task must not start execution earlier than the minimum finishing time of its predecessor(s): $s_j \geq r_i + C_i$

$\rightarrow$ $r_j^* = max(r_j, max(r_i^* + C_i : J_i \rightarrow J_j))$

Task 4

EDF* - Precedence graph example

Given the precedences $A \to C$, $B \to C$, $C \to E$, $D \to F$, $B \to D$, $C \to F$, $D \to G$ we first draw the precedence graph:

Task 4

EDF* - Precedence graph example

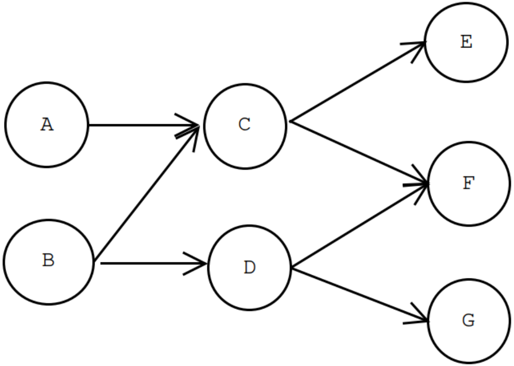


Figure 1: Task 4: precedence graph

Task 4 I

EDF* - Transformation example

- $r_A^* = r_A$, $r_B^* = r_B$
- $r_C^* = \max\{r_C, \max\{r_A^* + C_A, r_B^* + C_B\}\} = \max\{0, \max\{3, 2\}\} = 3$
- $r_D^* = \max\{r_D, r_B^* + C_B\} = \max\{0, 2\} = 2$
- $r_F^* = \max\{r_F, \max\{r_C^* + C_C, r_D^* + C_D\}\} = \max\{0, \max\{7, 5\}\} = 7$
- $r_E^* = \max\{r_E, r_C^* + C_C\} = \max\{0, 7\} = 7$
- $r_G^* = \max\{r_G, r_D^* + C_D\} = \max\{0, 5\} = 5$

Task 4 II

EDF* - Transformation example

- $d_E^* = d_F^* = d_G^* = 20$
- $d_C^* = \min\{d_C, \min\{d_E^* - C_E, d_F^* - C_F\}\} = \min\{20, \min\{18, 15\}\} = 15$
- $d_D^* = 15$
- $d_A^* = 11$
- $d_B^* = 11$

We now successfully have transformed the problem into one without precedence and can simply use EDF!

Task 4

EDF* - Transformation example

- 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

Task 4 I

EDF* - Schedule

Task 4.2:

- determine a resulting EDF* schedule
- for this schedule, compute the average of all response times of the tasks

Requirements 4.2:

- *average response time:* $\overline{t_r} = \frac{1}{n} \sum_{i=1}^{n} (f_i - r_i)$
  - *metric to compare schedules*

Task 4

EDF* - Schedule

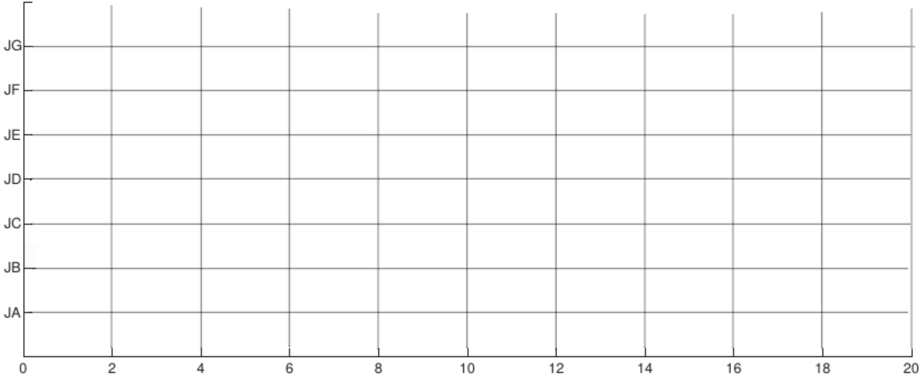


Figure 7: EDF* schedule

Task 4

EDF* - Schedule

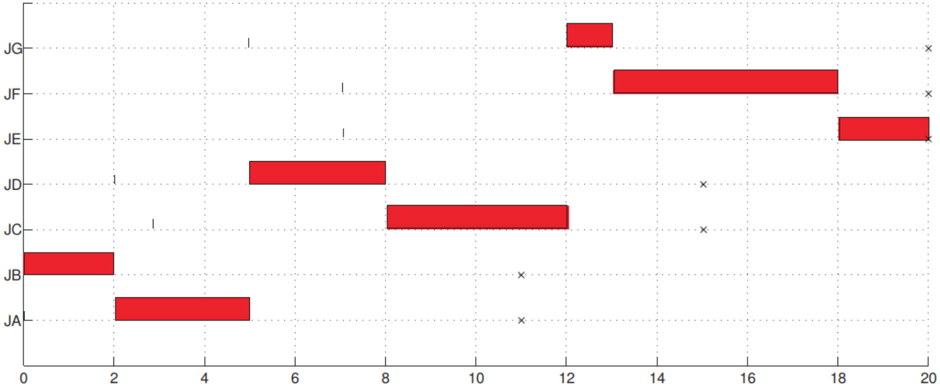


Figure 7: EDF* schedule

Task 4 I

EDF* - Schedule

Solution 4.2:

- $\overline{t_r} = \frac{1}{7}\sum_{i=1}^{7}(f_i - r_i) = \frac{5+2+9+6+13+11+8}{7} = \frac{54}{7} = 7.714$

Task 4 I

EDF* - Schedule

Task 4.3:

- additional precedence constraint $E \rightarrow A$
- still a feasible schedule for the task set

Requirements 4.3:

- *Precedence relations between tasks can be described through an acyclic directed graph G where tasks are represented by nodes and precedence relations by arrows. G induces a partial order on the task set.*

Task 4 II

EDF* - Schedule

Solution 4.3:

- *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

Task 5 I

Earliest Deadline First – Star

Task 5.1:

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

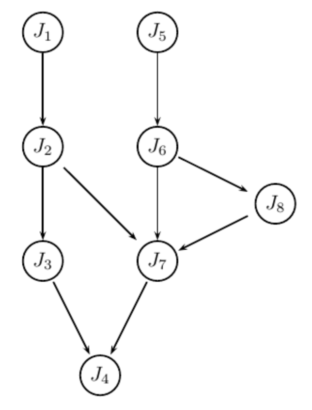
	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
r_i	0	3	4	0	0	2	0	2
d_i	3	8	15	15	10	10	10	11
C_i	1	3	3	3	1	1	2	1

- construct precedence graph

Task 5 II

Earliest Deadline First – Star

Solution 5.1:



Task 5 I

Earliest Deadline First - Star

Task 5.2:

- apply EDF* algorithm for modified arrival times and deadlines

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

Task 5 II

Earliest Deadline First - Star

- $r_1^* = r_1 = 0, \ r_5^* = r_5 = 0$
- $r_2^* = \max\{r_2, r_1^* + C_1\} = \max\{3, 0 + 1\} = 3$
- $r_6^* = \max\{r_6, r_5^* + C_5\} = \max\{2, 0 + 1\} = 2$
- $r_3^* = \max\{r_3, r_2^* + C_2\} = \max\{4, 3 + 3\} = 6$
- $r_8^* = \max\{r_8, r_6^* + C_6\} = \max\{2, 2 + 1\} = 3$
- $r_7^* = \max\{r_7, \max\{r_2^*+C_2, r_6^*+C_6, r_8^*+C_8\}\} = \max\{0, \max\{3+3, 2+1, 3+1\}\} = 6$
- $r_4^* = \max\{r_4, \max\{r_3^* + C_3, r_7^* + C_7\}\} = \max\{0, \max\{6 + 3, 6 + 2\}\} = 9$

Task 5 III

Earliest Deadline First - Star

- $d_4^* = d_4 = 15$
- $d_3^* = \min\{d_3, d_4^* - C_4\} = \min\{15, 15 - 3\} = 12$
- $d_7^* = \min\{d_7, d_4^* - C_4\} = \min\{10, 15 - 3\} = 10$
- $d_2^* = \min\{d_2, \min\{d_3^* - C_3, d_7^* - C_7\}\} = \min\{8, \min\{12 - 3, 10 - 2\}\} = 8$
- $d_1^* = \min\{d_1, d_2^* - C_2\} = \min\{3, 8 - 3\} = 3$
- $d_8^* = \min\{d_8, d_7^* - C_7\} = \min\{11, 10 - 2\} = 8$
- $d_6^* = \min\{d_6, \min\{d_7^* - C_7, d_8^* - C_8\}\} = \min\{10, \min\{10 - 2, 8 - 1\}\} = 7$

Task 5 IV

Earliest Deadline First - Star

- $d_5^* = \min\{d_5, d_6^* - C_6\} = \min\{10, 7 - 1\} = 6$

Task 5 V

Earliest Deadline First - Star

Solution 5.2:

	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

Task 5 I

Earliest Deadline First - Start

Task 5.3:

- dual-core platform
- at any time t, both cores execute the two ready tasks ($r_i* \leq t$) with earliest deadlines
- a single task cannot be executed on two cores simultaneously
- construct schedule:

Task 5 II

Earliest Deadline First - Start

Task 5.3:

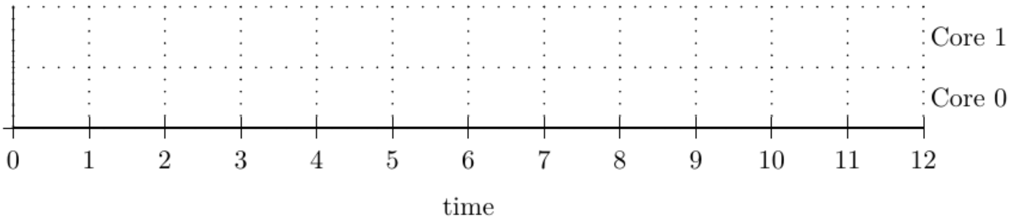


Figure 8: EDF schedule for part (3)

Task 5 III

Earliest Deadline First - Start

Solution 5.3:

- *one of several solutions since the tasks can run either on core 0 or core 1*

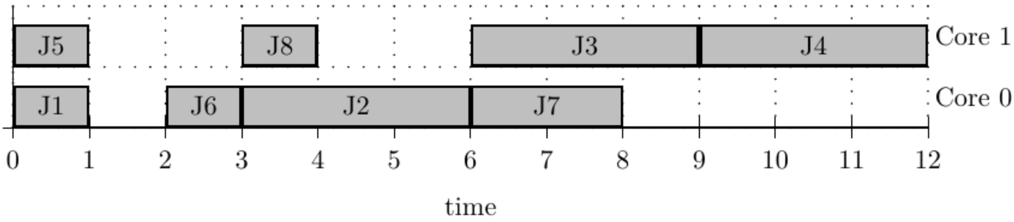


Figure 9: One solution for part (3)

Task 5 I

Earliest Deadline First - Start

Task 5.4:

- Will executing on the quad-core platform with the same scheduling rule reduce the completion time of the application?
  - 4 cores execute the four ready tasks with earliest deadlines

Solution 5.4:

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