4. Scheduling in Echtzeit-Systemen

4.1. Grundlagen

- **Taskbeschreibung**

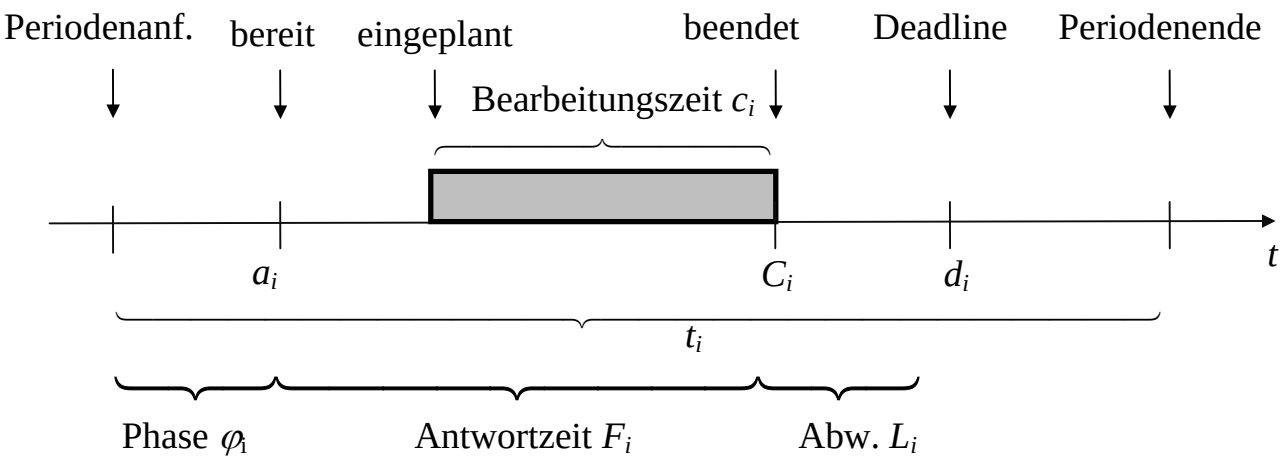
– ***Task:*** Planungseinheit. $T = \{\tau_1, ..., \tau_n\}$

– ***Taskparameter***

Anforderungszeit, Bereitzeit (release time)	a_i
Bearbeitungs-, Ausführungszeit (execution, processing time)	c_i
Zeitschranke, Frist (deadline)	d_i
Periodendauer	t_i
Phasenverschiebung (phase)	φ_i
Priorität	p_i
Taskabhängigkeiten (Präzedenzrelation)	R

– ***Bewertungsgrößen***

Beendigungszeit (computation time)	C_i
Antwortzeit (flow time)	$F_i = C_i - a_i$
Deadline-Abweichung (lateness)	$L_i = C_i - d_i$
Deadline-Überschreitung (tardiness)	$T_i = \max(L_i, 0)$



- **Bewertungsmaße**

Gesamtzeit	$C_{max} = \max C_i$
mittlere Antwortzeit	$\overline{F} = \frac{1}{n} \cdot \sum_{i=1}^{n} F_i$
maximale Deadline-Abweichung	$L_{max} = \max L_i$
mittlere Deadline-Überschreitung	$\overline{T} = \frac{1}{n} \cdot \sum_{i=1}^{n} T_i$
keine Deadline-Überschreitung	$T_i = 0 \quad \forall \tau_i \in T$

- **Begriffe**

– ***Ablaufplan (Schdule***) Zuordnung der Tasks zu Prozessoren bei Berücksichtigung der Task-Parameter

– ***Ausführbarer Ablaufplan:*** Ablaufplan, bei dem alle Tasks stets ihre Zeitschranke einhalten.

– ***Scheduling-Verfahren (-Algorithmus):*** Algorithmus, der bei gegebener Taskbeschreibung für jede Taskmenge einen Ablaufplan bestimmt.

– ***Admission:*** Verfahren, das für eine Taskmenge und ein Scheduling-Verfahren die Existenz eines *ausführbaren* Ablaufplans entscheidet.

– ***Optimalität bzgl. Einplanbarkeit***

eines Scheduling-Verfahrens in einer Klasse $\mathbb{C}$ von Verfahren: führt nur dann zu keinem ausführbaren Ablaufplan, wenn auch kein anderes Verfahren aus $\mathbb{C}$ einen ausführbaren Ablaufplan erzeugt.

- **Klassifikation**

zeit-/ereignisgetrieben

statisch/dynamisch

Entziehbarkeit

Ein-/Mehrprozessorsysteme

4.2. Ratenmonotones Scheduling RMS

- **Modellannahmen**

  (1) Anforderungen an Tasks sind periodisch mit konstanter Periodendauer t_i, d.h. mit konstanter Anforderungsrate $r_i = t_i^{-1}$.

  (2) τ_i ist bereit zu Periodenbeginn, Ausführungszeit c_i ist konstant und höchstens gleich t_i.

  (3) Die Deadline einer Task ist gleich deren Periodenende.

  (4) Die Tasks besitzen feste Prioritäten.

  (5) Die Tasks sind voneinander unabhängig ($R = \varnothing$).

  (6) Eine Task höherer Priorität verdrängt eine Task niedrigerer Priorität sofort, Overhead wird vernachlässigt.

- **Taskbeschreibung**

  τ_i: (c_i, t_i, p_i) $\quad i = 1,...n.$

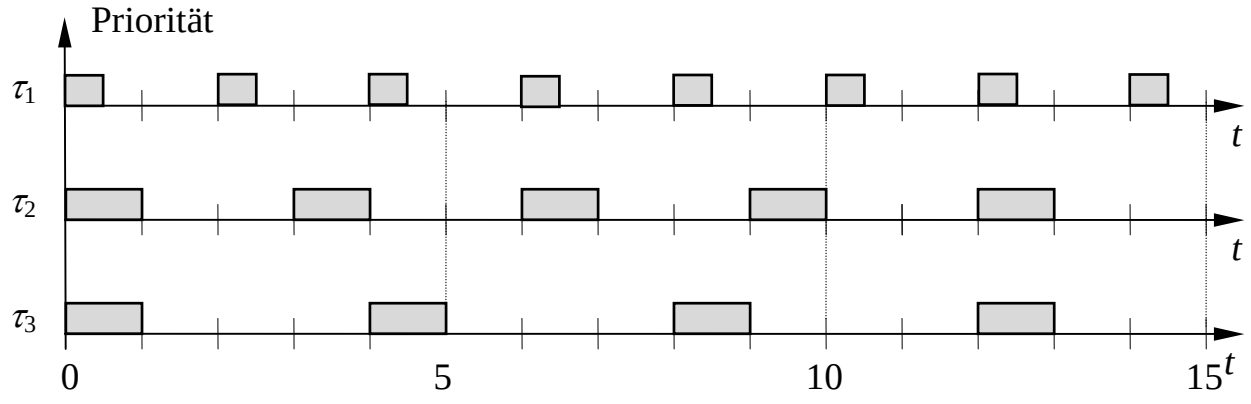
- **Kritischer Moment** einer Task:

  Zeitpunkt, zu dem die Anforderung einer Task zu deren größtmöglicher Antwortzeit führt.

– ***Kritische Zeit*** einer Task (kritisches Intervall):

  Zeit zwischen kritischem Moment und Beendigung der Task.

– ***Beispiel 4.1.*** $t_1 = 2,\ t_2 = 3,\ t_3 = 4,\ c_1 = \frac{1}{2}, c_2 = c_3 = 1;\ p_1 \succ p_2 \succ p_3.$



- **Satz 1.** Für eine Task tritt dann ein kritischer Moment ein, wenn die Task gleichzeitig mit allen Tasks höherer Priorität angefordert wird.

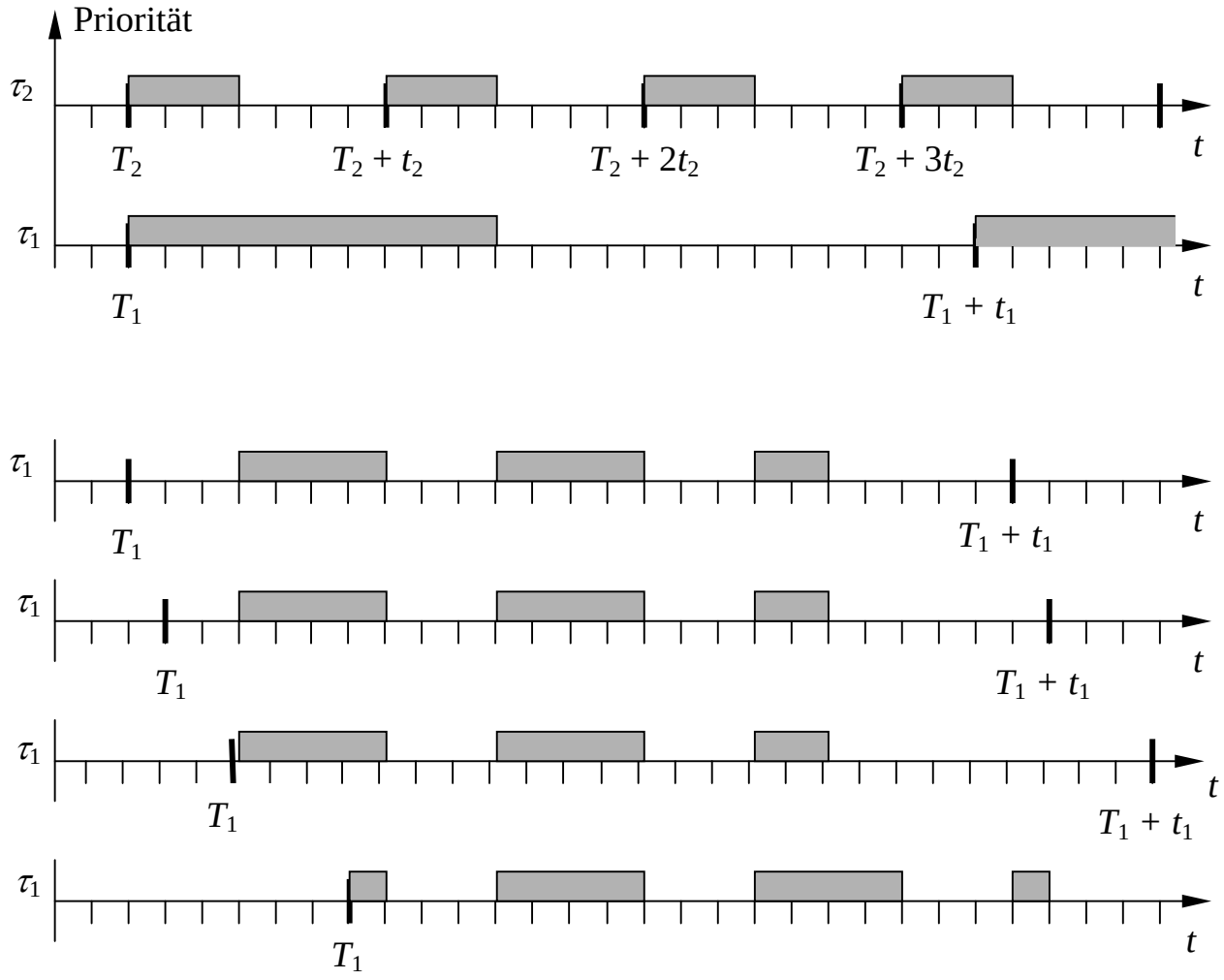
  Beweis (n = 2).

  Sei $p_2 \succ p_1$. Zur Zeit T_1 erfolge eine Anforderung an τ_1, und für τ_2 erfolge eine Anforderung zu den Zeiten

  $T_2 = T_1, \quad T_2 + t_2, \quad T_2 + 2t_2, \ldots$

  Dann erhöht sich die Antwortzeit von τ_1 um $\left\lceil \frac{c_1}{t_2 - c_2} \right\rceil \cdot c_2$.

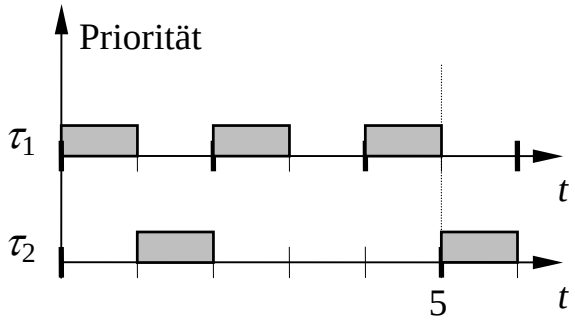
  Weiter: Ein Vergrößern oder Verkleinern von T_1 läßt die Antwortzeit von τ_1 konstant oder verringert sie.



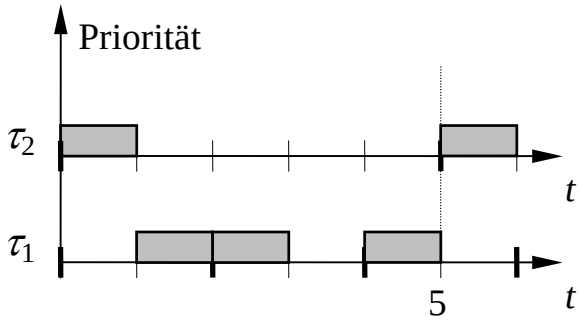
– ***Folgerung.*** Für die Existenz eines Ablaufplans ist es hinreichend (und notwendig), daß alle Tasks, die in kritischen Momenten angefordert werden, ihre Deadline einhalten. Damit genügt es bei der Untersuchung der Ausführbarkeit eines Ablaufplans, die gleichzeitige Anforderung von Tasks zu betrachten.

– ***Beispiel 4.2.*** τ_1: (1, 2, p_1), τ_2: (1, 5, p_2).

a) $p_1 \succ p_2$:



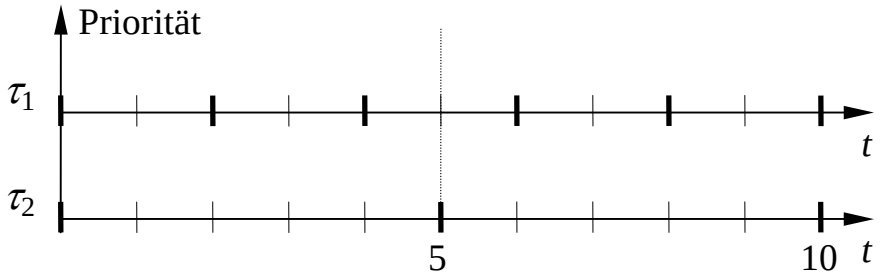
b) $p_2 \succ p_1$:



c) Modifikation:

c_1 =

(max.)



- ***Lemma 1.*** Sei $T = \{\tau_1, \tau_2\}$ mit $t_1 < t_2$. Gibt es einen ausführbaren Ablaufplan bei $p_2 \succ p_1$, so gibt es einen solchen Plan auch für $p_1 \succ p_2$.

Beweis.

Notwendig (aber nicht hinreichend) für die Existenz eines Ablaufplans bei $p_1 \succ p_2$ ist mit $k = \left\lfloor \frac{t_2}{t_1} \right\rfloor$ (s. Abb.; man beachte Satz 1) die Bedingung

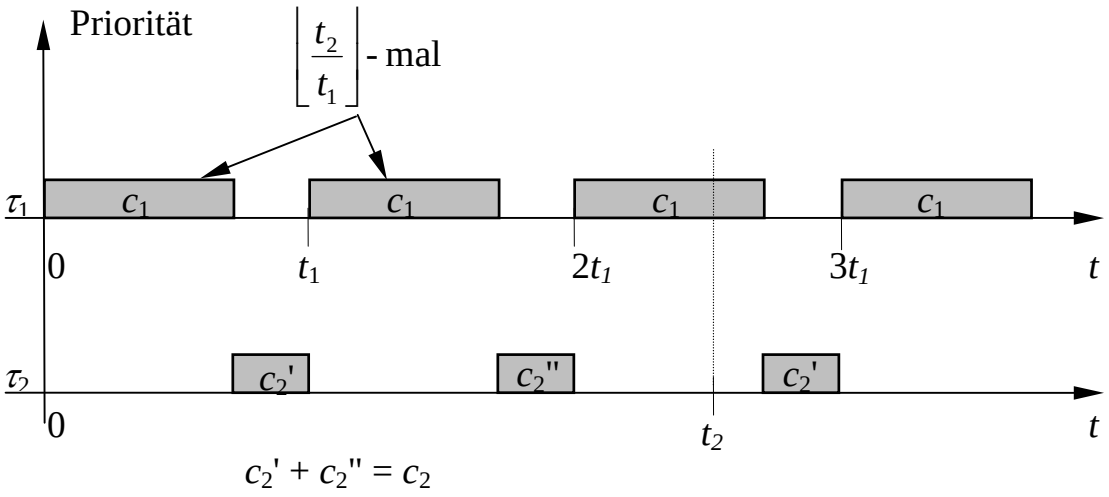
$$kc_1 + c_2 \leq t_2 . \qquad (4.1)$$

Sei nun $p_2 \succ p_1$. Dann muß offenbar gelten:

$$c_1 + c_2 \leq t_1. \qquad (4.2)$$

Diese Bedingung impliziert wegen $k \geq 1$ Bedingung (4.1.):

$$kc_1 + c_2 \leq kc_1 + kc_2 \leq kt_1 \leq t_2.$$



- ***Lemma 1.*** Sei $T = \{\tau_1, \tau_2\}$ mit $t_1 < t_2$. Gibt es einen ausführbaren Ablaufplan bei $p_2 \succ p_1$, so gibt es einen solchen Plan auch für $p_1 \succ p_2$.

Beweis.

Gibt es einen Ablaufplan bei $p_2 \succ p_1$, so muß offenbar gelten:

$$c_1 + c_2 \le t_1. \tag{4.1}$$

Sei nun $p_1 \succ p_2$. Hinreichend für die Ausführbarkeit eines Ablaufplans ist dann mit $k = \left\lfloor \frac{t_2}{t_1} \right\rfloor$ (s. Abb.; man beachte Satz 1)

$$kc_1 + (t_2 - kt_1) + c_2 \le t_2. \tag{4.2}$$

Dies ist gleichbedeutend mit

$$k(c_1 - t_1) + t_2 + c_2 \le t_2,$$

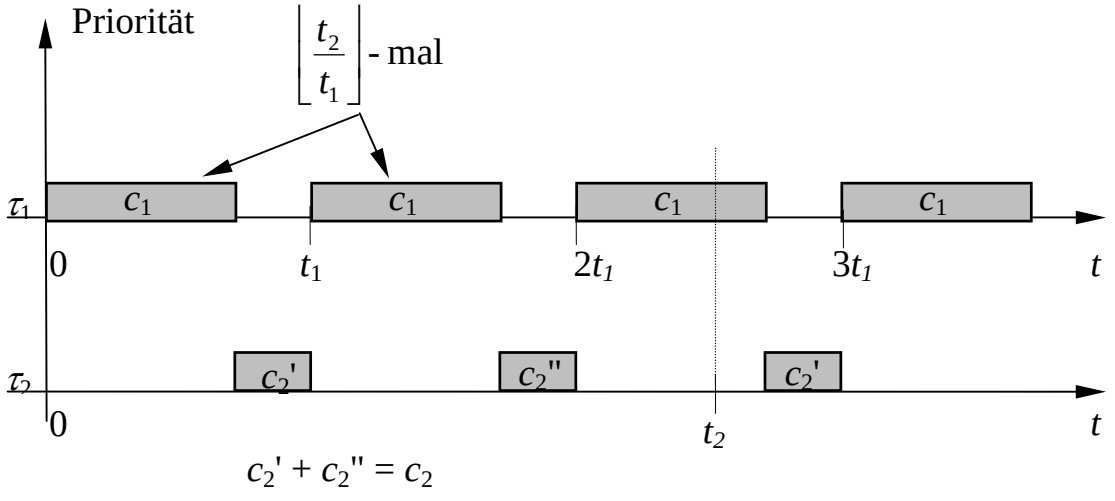
d.h.

$$c_2 \le k(t_1 - c_1). \tag{4.3}$$

Diese Bedingung folgt leicht aus (4.1.):

$$c_2 \le t_1 - c_1 \le k(t_1 - c_1),$$

letzteres, da $k \ge 1$ (wegen $t_2 > t_1$) und $t_1 \ge c_1$.



- ***Lemma 1.*** Für $T = \{\tau_1,...,\tau_n\}$ sei eine statische Prioritätszuordnung *pr* gegeben; o.B.d.A. sei $pr(\tau_i) = i$, $i = 1,...,n$ (1: höchste Priorität). Weiter gebe es ein $k \in \{1,...,n-1\}$ mit $t_k > t_{k+1}$, und T sei unter diesen Bedingungen einplanbar. Dann ist T auch einplanbar, wenn die Prioritäten von τ_k und τ_{k+1} vertauscht werden.

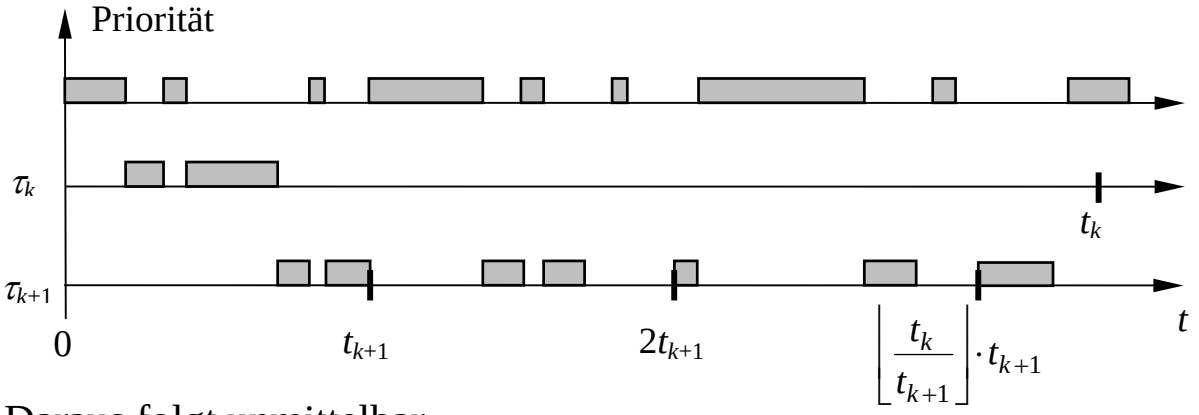
Beweis.

Sei $m = \left\lfloor \frac{t_k}{t_{k+1}} \right\rfloor$

und $c_0 = \sum_{i=1}^{k-1} c_i^*$, c_i^*: Gesamtausführungszeit von τ_i in $[0, m \cdot t_{k+1}]$.

Notwendig für die Einplanbarkeit von T bei *pr* ist

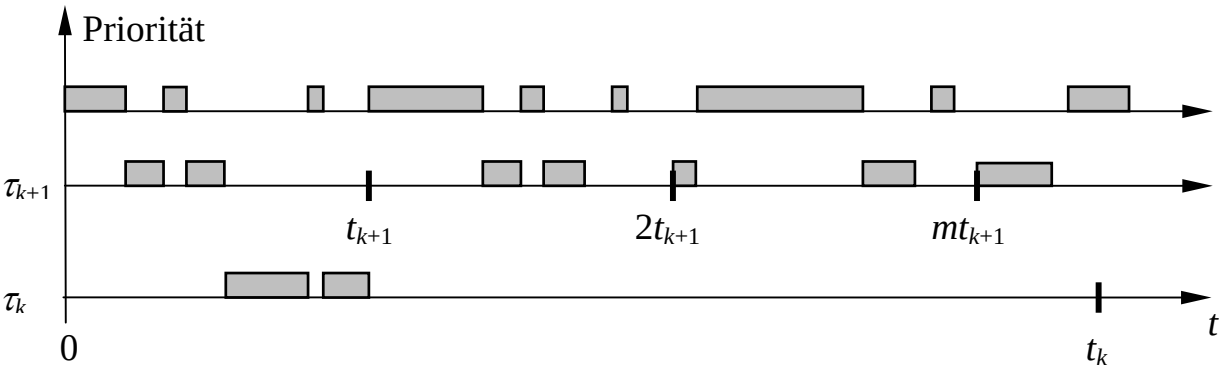
$$c_0 + c_k + mc_{k+1} \le mt_{k+1}. \qquad (4.1)$$



Daraus folgt unmittelbar

$$c_k \le m(t_{k+1} - c_{k+1}) - c_0, \qquad (4.2)$$

was (unter Beachtung von Satz 1) hinreichend für die Einplanbarkeit von T nach Vertauschen der Prioritäten von τ_k und τ_{k+1} ist.



- **Ratenmonotone Prioritätszuordnung RMS**

  Den Tasks einer Taskmenge T wird eineindeutig eine Priorität in der Reihenfolge ihrer Anforderungsraten zugeordnet (höchste Rate entspricht höchster Priorität; bei gleicher Rate kann Priorität beliebig festgelegt werden).

- **Satz 2 (Optimalitätseigenschaft von RMS).**

  Sei T eine Taskmenge, und sei $\mathfrak{A}(T)$ die Klasse aller statischen Prioritätszuordnungen. Dann gilt: Gibt es für T irgendeine Zuordnung aus $\mathfrak{A}(T)$, die zu einem ausführbaren Ablaufplan führt, so führt auch RMS zu einem solchen Plan.

  Beweis.

  Sei

  P: $T \to \{1,..., n\}$ eineindeutig,

  so daß es einen Ablaufplan gebe. Sind $\tau_i, \tau_{i+1} \in T$ mit $t_i < t_{i+1}$ und $P(\tau_{i+1}) \succ P(\tau_i)$, so ist nach Lemma 1 auch der Ablaufplan ausführbar, bei dem die Prioritäten von τ_i und τ_{i+1} vertauscht sind.

  RMS ist eine Permutation der ursprünglichen Prioritätszuordnung, jede Permutation läßt sich als Produkt von Transpositionen darstellen.

4.3. Admission-Kriterien für RMS

Gegeben sei eine Taskmenge $T = \{\tau_1,...,\tau_n\}$, $\tau_i := (c_i, t_i, p_i)$, $i = 1,...,n$.

- **Begriffe.**

– ***Prozessorauslastung von T***

$$U = \sum_{i=1}^{n} \frac{c_i}{t_i} \tag{3.4}$$

Eine Taskmenge T heißt ***den Prozessor voll auslastend,*** wenn es für T bei der geg. Prioritätszuordnung einen Ablaufplan gibt, der jedoch bei Vergrößerung der Bearbeitungszeit einer Task nicht mehr ausführbar ist.

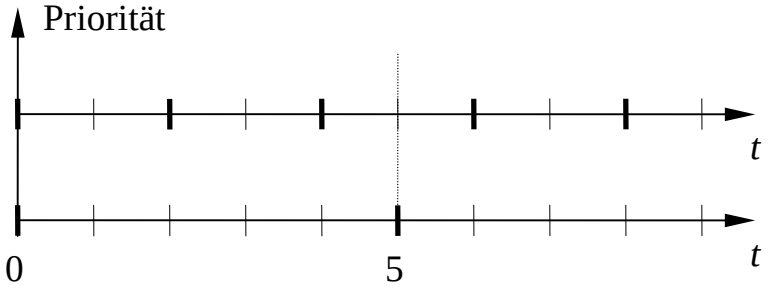
Aufgabe: Bestimmung eines Wertes U_g, so daß es für alle Taskmengen mit $U \le U_g$ stets einen Ablaufplan gibt. Da RMS optimal, genügt es, U_g bzgl. RMS zu bestimmen.

– ***Obere Grenze U_g der Prozessorauslastung (schedulable utilization)***

Minimum von U über alle Taskmengen (über alle möglichen Werte von c_i und t_i), die bei RMS den Prozessor voll auslasten.

– ***Beispiel.*** ***4.2. c)*** $U =$

4.2. d) $c_1 = 0{,}9$ $t_1 = 2$ $c_2 = 2{,}5$ $t_2 = 5$ $U =$



- **Lemma 2.** Für zwei Tasks ist die obere Grenze U_g der Prozessorauslastung unter RMS

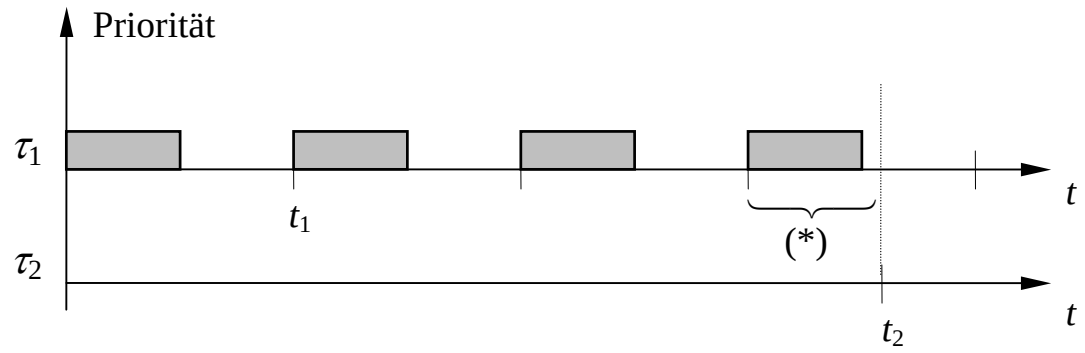
$$U_g = 2(\sqrt{2} - 1).$$

Beweis.

Sei $T = \{\tau_1, \tau_2\}$ eine Menge von zwei Tasks mit $t_2 > t_1$, mithin $p_1 > p_2$. Dann ist (Satz 1!) $[0, t_2) =: I$ das kritische Intervall für τ_2. In I treten $\left\lceil \frac{t_2}{t_1} \right\rceil =: l$ Anforderungen an τ_1 auf. Weiter sei $k := \left\lfloor \frac{t_2}{t_1} \right\rfloor$.

– ***Fall a):***

Alle in I auftretenden Anforderungen an τ_1 werden vollständig in I erfüllt. Das bedeutet (s. Abb.)



$$c_1 \le t_2 - kt_1. \qquad (*)$$

Dann lastet T den Prozessor genau dann voll aus, wenn

$$c_2 = t_2 - lc_1,$$

und damit ist

$$U = \frac{c_1}{t_1} + \left(\frac{t_2}{t_2} - l \cdot \frac{c_1}{t_2} \right) =$$

$$= 1 + c_1 \left(\frac{1}{t_1} - \frac{l}{t_2} \right), \qquad (a)$$

und wegen

$$l = \left\lceil \frac{t_2}{t_1} \right\rceil \geq \frac{t_2}{t_1}$$

ist

$$\frac{l}{t_2} \geq \frac{1}{t_1}$$

und damit ist (a) monoton fallend in c_1.

– ***Fall b):***

Bei

$$c_1 \geq t_2 - kt_1$$

lastet T den Prozessor genau dann voll aus, wenn

$$c_2 = k(t_1 - c_1),$$

so daß

$$U = \frac{c_1}{t_1} + k \cdot \frac{t_1 - c_1}{t_2} =$$

$$= k \cdot \frac{t_1}{t_2} + c_1 \left(\frac{1}{t_1} - \frac{k}{t_2} \right). \qquad \text{(b)}$$

Dabei ist wegen $\lfloor x \rfloor \leq x$ der Faktor von c_1 nichtnegativ und somit U monoton steigend in c_1.

– ***Also:***

U_g tritt bei $c_1 = t_2 - kt_1$ auf, und dann gilt

$$U_g = 1 - \frac{t_1}{t_2} \left(l - \frac{t_2}{t_1} \right) \left(\frac{t_2}{t_1} - k \right).$$

Sei

$$r = \frac{t_2}{t_1} - k \neq 0.$$

Dann ist

$$\frac{t_2}{t_1} = k + r\,, \qquad l = k + 1$$

und mithin

$$U_g = 1 - \frac{1}{k+r} \cdot (k + 1 - k - r) \cdot r =$$

$$= 1 - \frac{r(1-r)}{k+r} = \varphi(k,r) \rightarrow \text{ Min.!}$$

Da U_g monoton in k steigt und $k \in \mathbb{N}^+$ gilt, bedeutet dies

$$U_g = 1 - \frac{r(1-r)}{r+1} \rightarrow \text{Min.!}$$

Daraus resultiert

$$r = \sqrt{2} - 1$$

und schließlich

$$U_g = 2(\sqrt{2} - 1).$$ ■

- **Satz 3** (LIU/LAYLAND 1973). Für eine Menge von n Tasks gibt es einen Ablaufplan gemäß RMS, wenn $U \le U_g$ gilt mit

$$\boxed{U_g = n\left(\sqrt[n]{2} - 1\right)} \tag{3.5}$$

(„obere Grenze der Prozessorauslastung“). o.B.

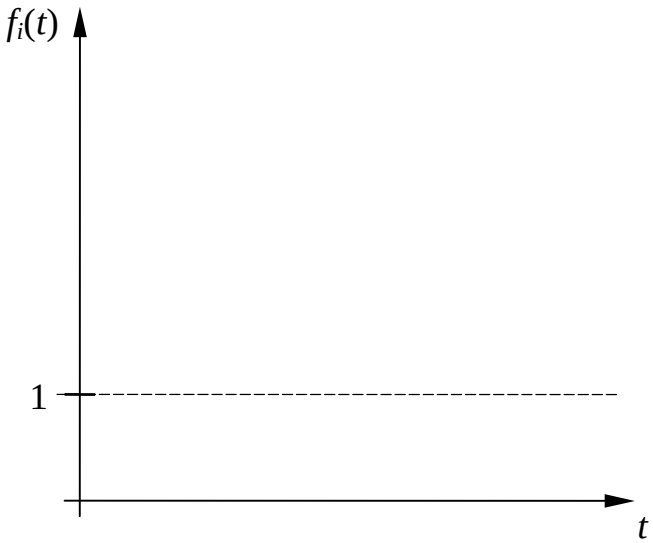
$n = 2$: $U_g \approx 0{,}828$

$n = 3$: $U_g \approx 0{,}780$

$n \to \infty$: $U_g \to \quad \approx 0{,}693$.

- **Satz 4 (LEHOCZKY 1989).** Sei $T = \{\tau_1,\ldots,\tau_n\}$ geordnet nach aufsteigender Periode. Dann gilt: Für T gibt es genau dann einen Ablaufplan gemäß RMS, wenn für alle $i = 1,\ldots,n$ gilt:

$$\min_{t\in(0,t_i]} f_i(t) \le 1 \qquad \text{mit} \qquad f_i(t) = \frac{1}{t}\cdot\sum_{j=1}^{i} c_j \left\lceil \frac{t}{t_j} \right\rceil .$$

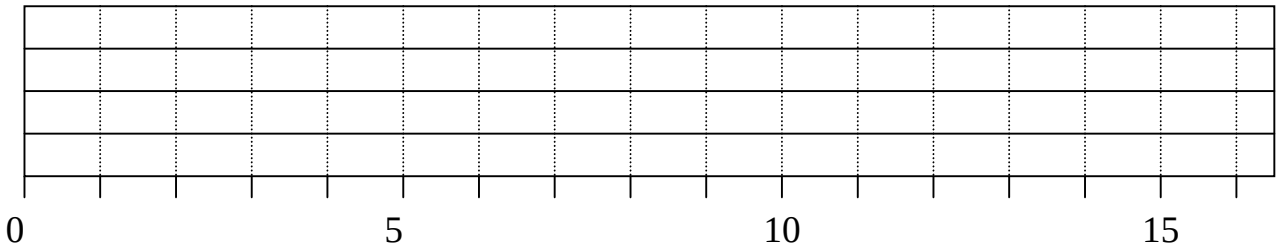


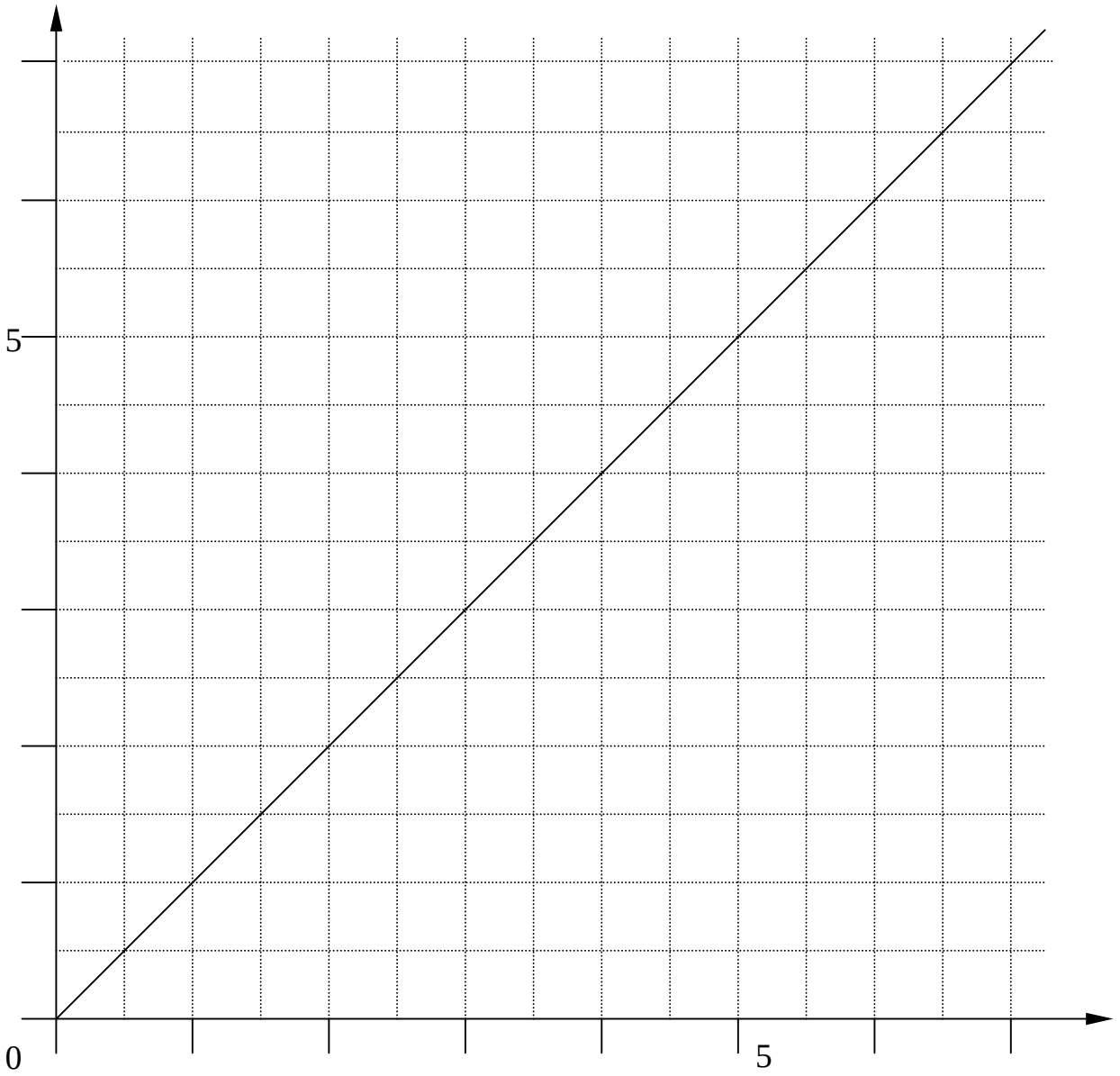
- **Analyse des Zeitbedarfs (time demand analysis)**

$$\overset{?}{\underset{i=1,\ldots,n}{\forall i}} \quad \underset{0<t\le t_i}{\exists t} \; : \quad w_i(t) = c_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{t_j} \right\rceil c_j \; < \; t$$

Beispiel 4.3.

T = {(2; 1), (4; 0,5), (5; 0,5), (6; 1,5)}





- **Struktur des Beweises für $n \geq 2$**

0. Es werden ausschließlich Taskmengen T betrachtet, die „gerade noch einplanbar“ sind (die den Prozessor maximal auslasten, schwer einplanbar sind). Für diese Taskmengen wird die kleinstmögliche Auslastung bestimmt.

   o.B.d.A. sei T nach Periodenlängen geordnet, d.h. $t_1 < t_2 < ... < t_n$.

1. Sei $t_n \leq 2t_1$.

– Bei gegebenen t_i wird eine Taskmenge konstruiert, die gerade noch einplanbar ist.

– Bei gleichen t_i hat jede andere Taskmenge eine höhere Auslastung.

2. Die Auslastung der in 1. konstruierten Taskmenge ist abhängig von den Periodenlängen, genauer von den Periodenverhältnissen $\frac{t_{i+1}}{t_i}$. Für diese Taskmengen ist der kleinstmögliche Auslastungswert $U_n = n(\sqrt[n]{2} - 1)$.

3. T sei eine Taskmenge mit $t_n > 2t_1$.

– Konstruktion einer Taskmenge T': (c_i, t_i') für $i < n$, (c_n', t_n) mit $t_1' < t_2' < t_n < 2t_1'$. Deren Auslastung ist nach 2. mindestens U_n.

– Die Auslastung der ursprünglichen Taskmenge T ist (echt) größer als die Auslastung von T'.

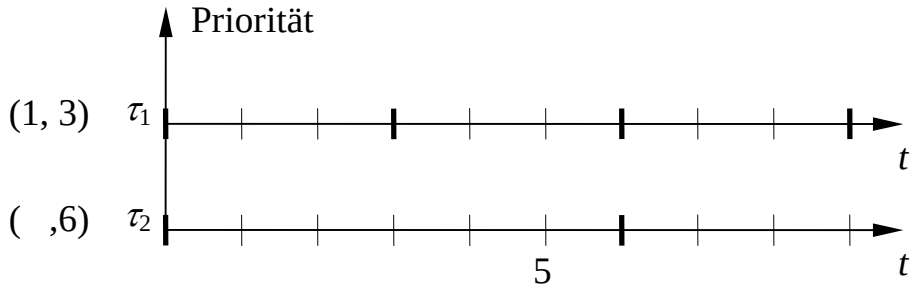
- **Beispiel 4.4.**

  $n = 2$

- **Folgerung.**

  Ist $\frac{t_j}{t_i} \in \mathbb{N}$ für alle i, j mit $1 \le i \le j$, $i, j = 1,\ldots,n$, so gibt es genau dann einen Ablaufplan, wenn

$$\sum_{i=1}^{n} \frac{c_i}{t_i} \le 1.$$



- **Verallgemeinerungen**

  $d_i < t_i$: RMS ist nicht mehr optimal. Aber: Zuordnung

$$\text{Priorität} \sim d_i^{-1}$$

  ist optimal („deadline-monotones Scheduling“).

  $d_i > t_i$: Weder raten- noch deadline-monotones Scheduling sind optimal.

4.4. Dynamische Scheduling-Verfahren

- **EDF – Earliest Deadline First**

– ***Voraussetzungen:*** wie in 3.2.

– ***Algorithmus***

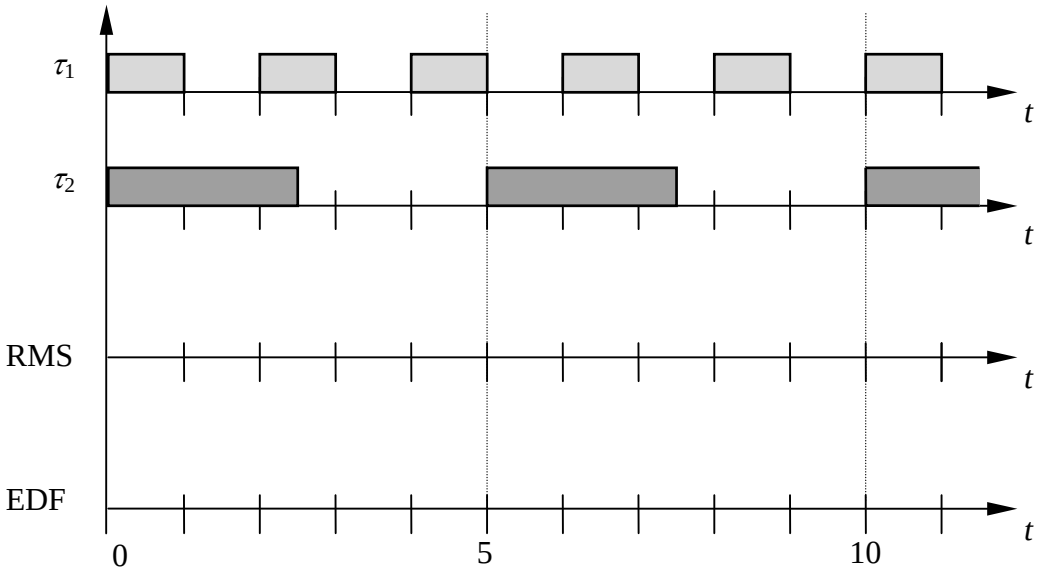
Unter den Tasks, die bereit und noch nicht vollständig beendet sind, wird diejenige ausgewählt, deren Zeitschranke am nächsten liegt.

– ***Es gilt:***

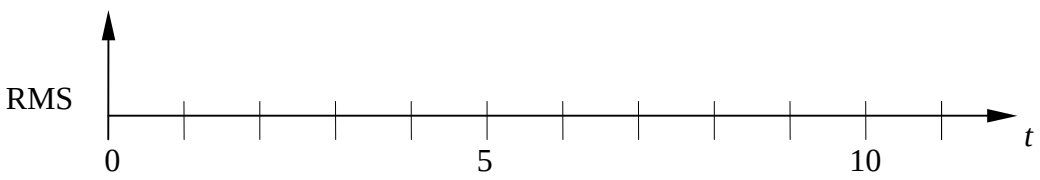
(1) EDF ist optimal bzgl. der Klasse $\mathfrak{A}(T)$ aller dynamischen Scheduling-Verfahren.

(2) Genau dann gibt es für eine Taskmenge T einen Ablaufplan nach EDF, wenn $U \leq 1$.

– ***Beispiel 4.4.*** τ_1: $c_1 = 1$ $t_1 = 2$ τ_2: $c_2 = 2{,}5$ $t_2 = 5$. $U =$



Modifikation: $c_2' = 2$ damit $U' =$ $\approx 0{,}83$

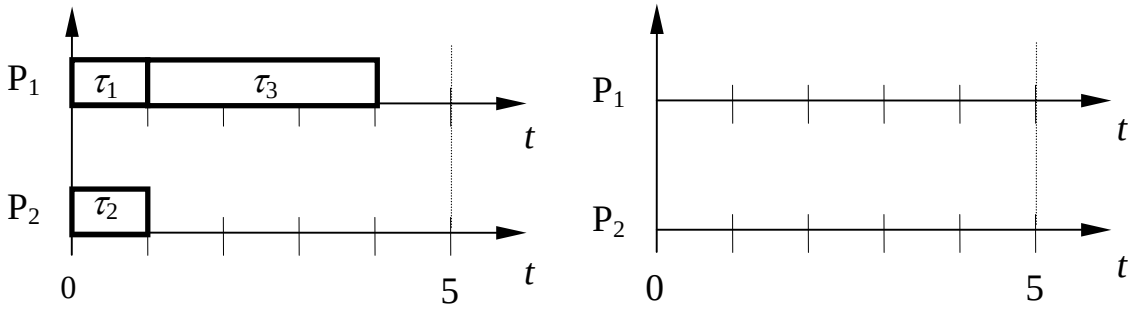


– ***Probleme***

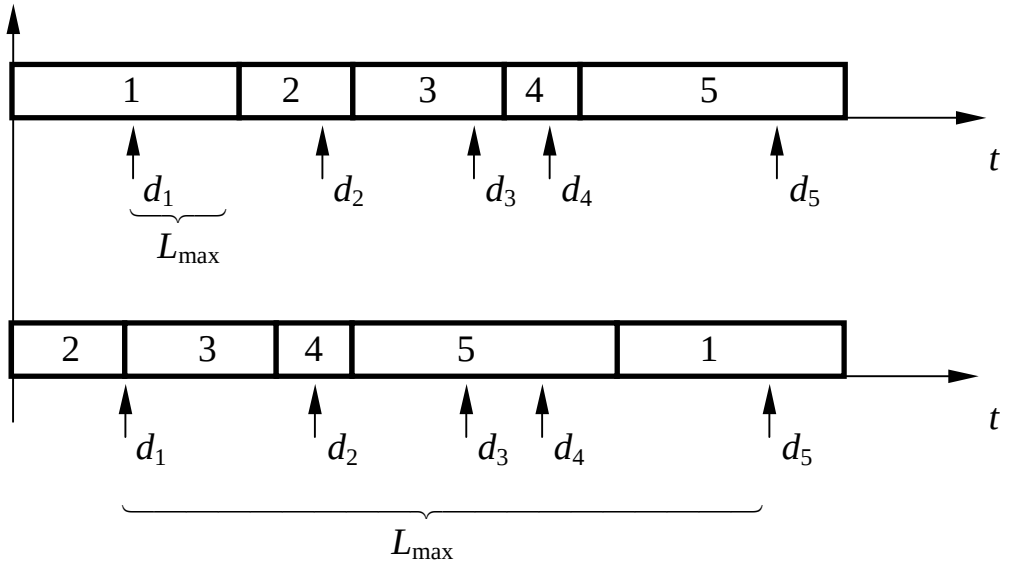
EDF ist nicht optimal bei $R \neq \varnothing$, gemeinsam genutzten Betriebsmitteln und in Mehrprozessorsystemen.

Beispiel 3.4. 2 Prozessoren P_1, P_2;

3 Tasks: (c_i, d_i): (1; 1) (1; 2) (3; 3,5)



„Domino-Effekt“:



- **Weitere Strategien**

LLF Least Laxity First

LRF Least Release Time First

MUF Maximum Urgency First (Gefährlichk. – Laxity – Nutzerprior.)

Gemischte Strategien: RMS/EDF

SJF Shortest Job First