Grundlagen der Echtzeitplanung

1. Grundlegende Begriffe und Konzepte

2. Planungsverfahren (Scheduling)

2.1 Planen aperiodischer Tasks
- Planen durch Suchen
- Planen nach Fristen
- Planen abhängiger Tasks

2.2 Planen periodischer Tasks
- Planen nach monotonen Raten

Klassifizierung der Schedulingalgorithmen

Annahmen über die Systemumgebung:
- Einprozessor-System,
- Mehrprozessor-System,
- verteiltes System, ...

Annahmen über die Task-Menge:
- Unterbrechbarkeit,
- Unabhängigkeit,
- Aktivierung, ...

Planungs(Optimalitäts)kriterium bezogen auf eine Kostenfunktion

Notation:

n | async | L_{max}

3 | non-preemt | N_{late}

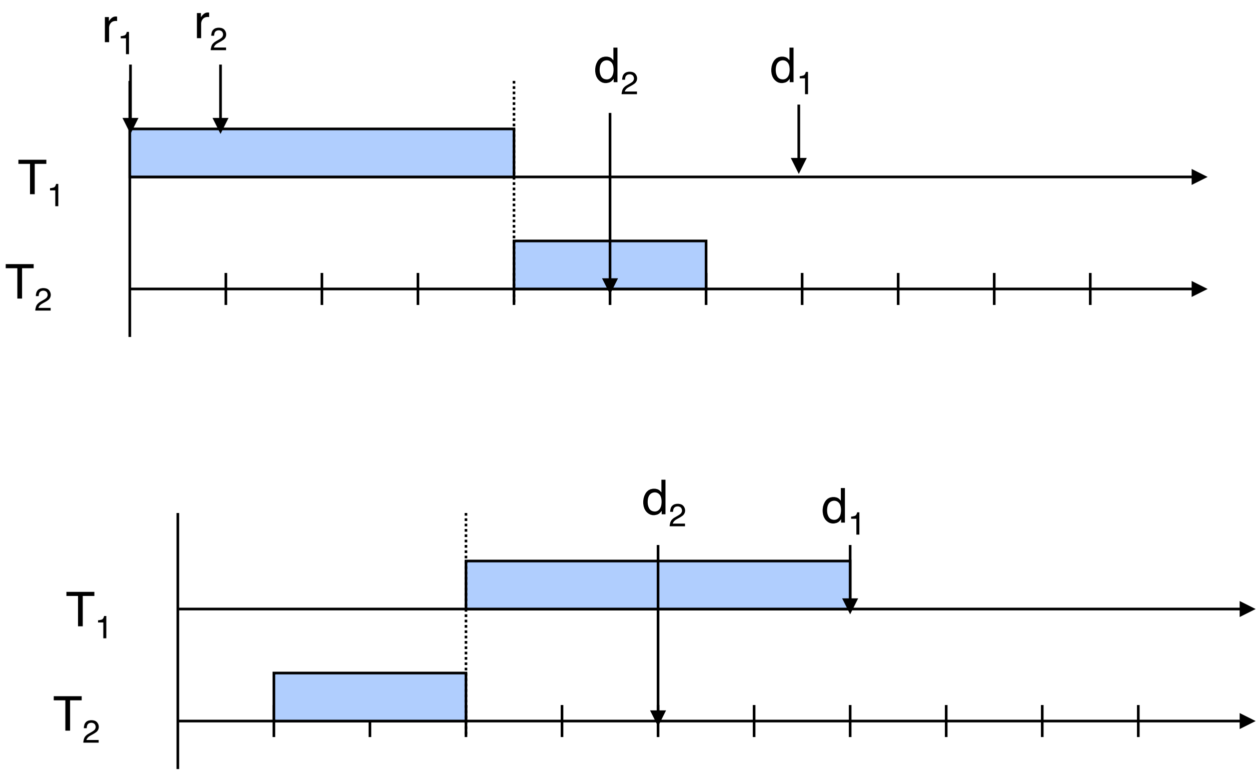
1 | sync | L_{max}

1 | async, preemt | L_{max}

Annahmen: System Taskmenge Kriterium

Planung nicht unterbrechbarer Tasks

Problem:



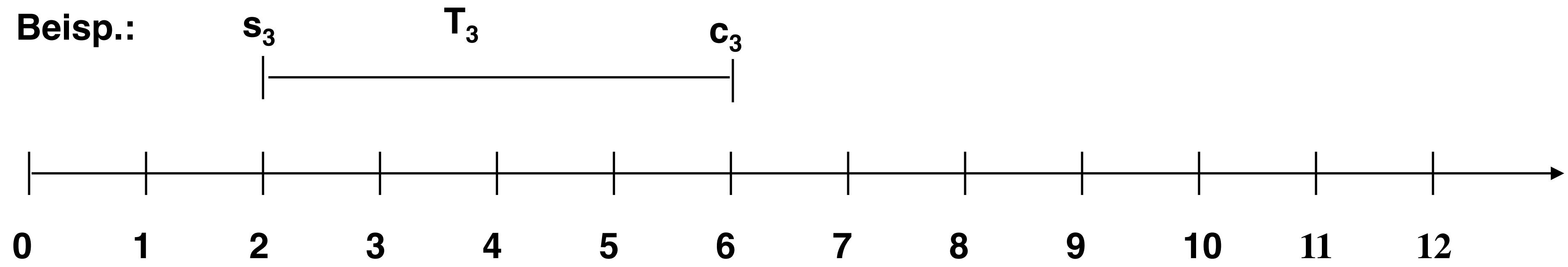
Planen durch Durchsuchen des Lösungsraums

1 | async | feasible

Gegeben : Menge T nicht unterbrechbarer Tasks mit |T| = n, r_i , Δ e_i , c_i .

Ein statisches Planungsverfahren soll untersuchen, ob ein Plan existiert. Das Verfahren soll einen Plan in Form einer Folge von Tupeln (i, s_i) mit i : Tasknr. und s_i : Startzeit von T_i .

Beisp.:



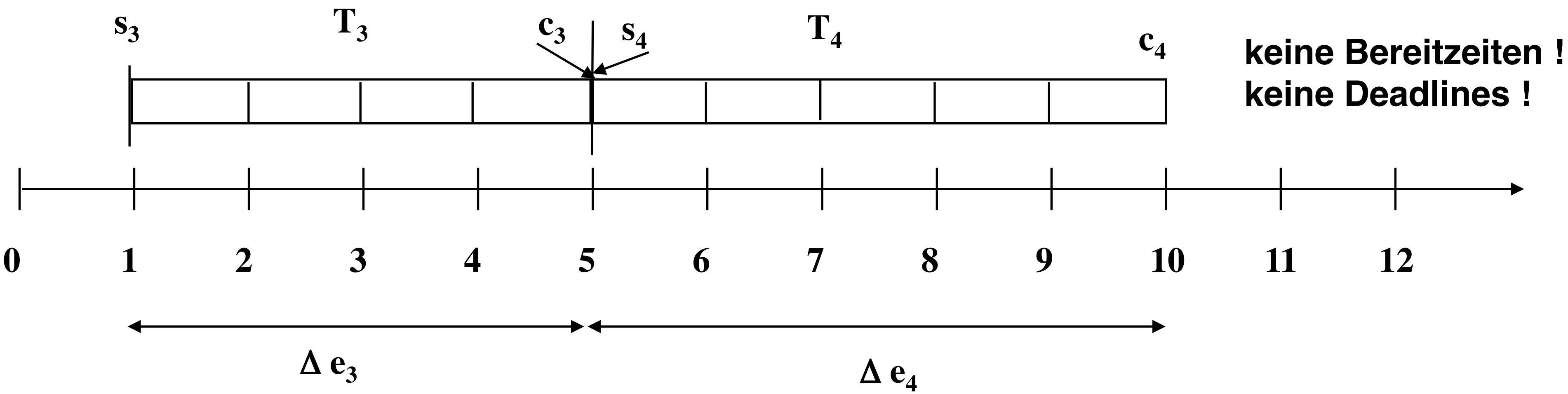
Der Eintrag für T_3 im Plan ist (i, s_i) = (3, 2)

Δ e_3 = 4
c_3 = 6

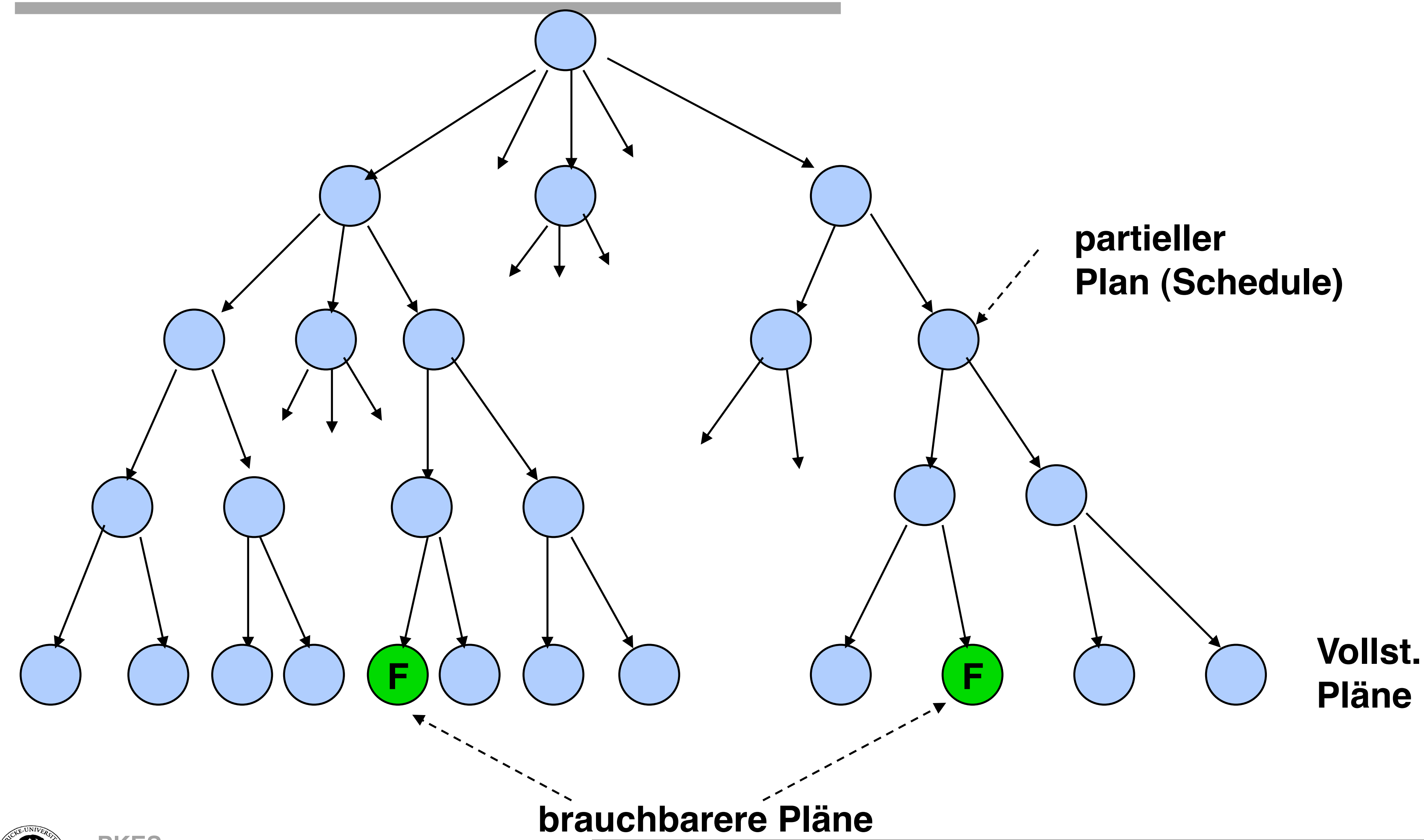
Generierung des Lösungsraums

Es werden alle möglichen Kombinationen von Tasks als Lösungen generiert ohne Beachtung von Bereitzeiten und Fristen.

Für n Tasks gibt es n! Lösungsmöglichkeiten. Die Lösungen einschließlich aller Lösungsschritte können in einem Baum der Tiefe n dargestellt werden. Jeder Knoten des Baums markiert einen bestimmten Stand des Planungsverfahrens.



Suchbaum zur Erstellung eines Plans



Planung unter Einbeziehung von Bereitzeiten und Fristen

Nicht alle kombinatorisch erzeugten Pläne sind gültig, wenn Bereitzeiten und Fristen berücksichtigt werden!

⇨ **Bewertungsfunktion wird benötigt !**

Heuristik: Einplanen einer Task so früh wie möglich, d.h. $s - r = min$

Vorgehensweise:

Trifft man im Baum auf einen ungültigen Plan, d.h.läßt sich eine Task mit entsprechender Deadline nicht mehr einplanen, wird die Suche in dem entsprechenden Ast nicht weiter fortgesetzt.

Planung unter Einbeziehung von Bereitzeiten und Fristen

Bratley`s Algorithmus (P. Bratley; M. Florian, P. Robillard: “Scheduling with earliest start and due date contstraints”, Naval Research Quarterly, 18 (4), 1971

Klasse: 1lnon-preemtlL_{max}

earliest PL (PL_k , i) **: Funktion, die bezogen auf den Plan PL_k die Task T_i in die früheste Lücke einplant, die**

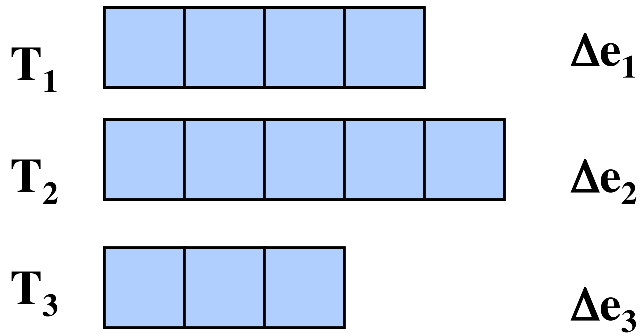
1. aufgrund der Bereitzeit r_i von T_i
2. aufgrund des bereits bisher erstellten Plans PL_k

möglich ist.

feasible PL (PL_k , i) **: Funktion, die testet, ob Task T_i in den bisherigen Plan PL_k integriert werden kann, d.h. ob überhaupt noch eine einplanbare Lücke für Task T_i existiert.**

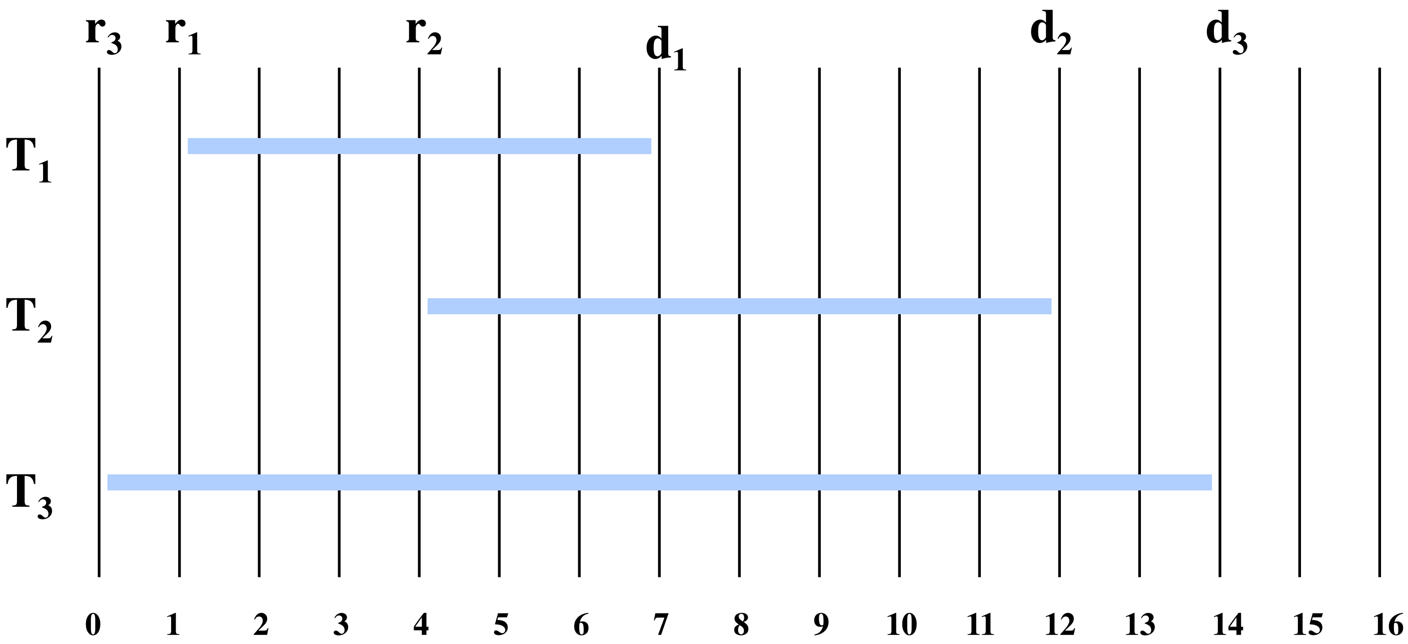
Planungsverfahren 2:

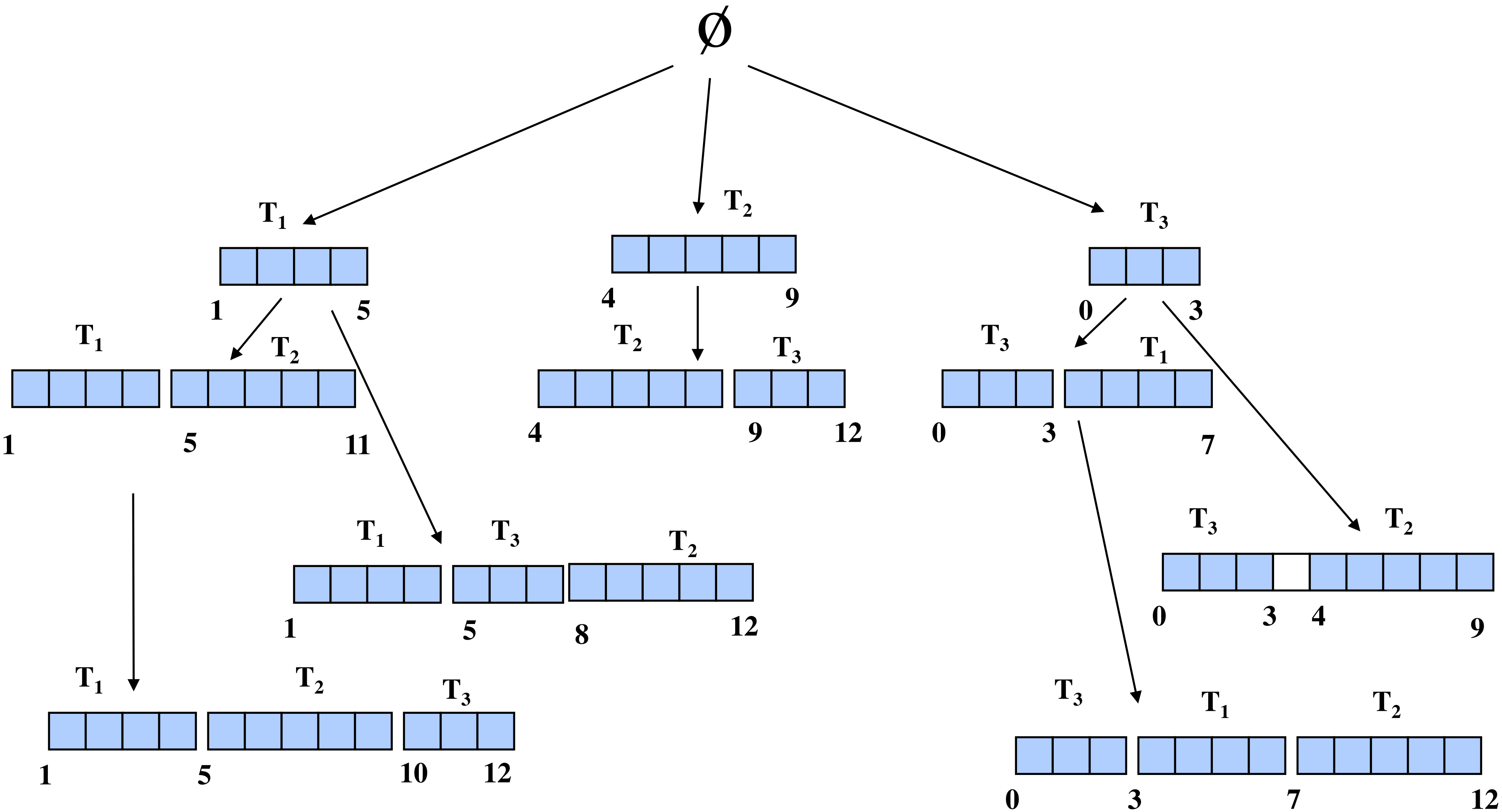
```
schedule (PL_k , X_k ) :
    FORALL ( i   IN  T \ X_k ) AND feasible PL (PL_k , i )
        schedule  (earliest PL (PL_k , i ) , X_k ∪ { i })
```



Taskliste

$T_1, r_1, d_1, \Delta e_1$	⇨	1, 1, 7, 4
$T_2, r_2, d_2, \Delta e_2$		2, 4, 12, 5
$T_3, r_3, d_3, \Delta e_3$		3, 0, 14, 3





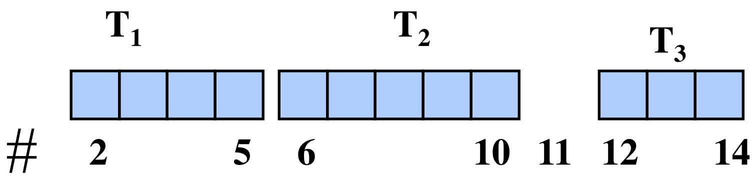
$<(T_1, s_1), (T_2, s_2)(T_3, s_3)>$
$= <(1, 1), (2, 5), (3, 10)>$

$<(T_3, s_3), (T_1, s_1)(T_2, s_2)>$
$= <(3, 0), (1, 3), (2, 7)>$

Gütekriterium

Ist das vorgestellte Planungsverfahren optimal ?

Es gibt Pläne, die gültig sind, aber nicht mit Planungsverfahren 2 erzeugt werden, z.B solche, die Lücken aufweisen.



Frage: Wenn schon nicht alle möglichen Pläne gefunden werden - gibt es Fälle, in denen ein gültiger Plan existiert, aber nicht erzeugt wird ?

**Satz: Gegeben sei eine Menge nicht unterbrechbarer Tasks T = {1, ... , n}.
Wenn es gültige Pläne gibt, werden diese durch Planungsverfahren 2 erzeugt.**

Probleme:

- Die kombinatorische Erzeugung aller Pläne benötigt n! Planungsschritte bei n Tasks.
- Die Konstruktion eines Plans mit diesem Verfahren ist nicht mit polynomiellen Aufwand zu realisieren.
- Tatsächlich ist das Suchverfahren NP-vollständig

Schlechte Nachricht:

Falls die Bereitzeiten nicht alle gleich sind und nicht-unterbrechbare Tasks vorliegen, gibt es kein anderes Verfahren das optimal ist.

Grundlagen der Echtzeitplanung

1. Grundlegende Begriffe und Konzepte

2. Planungsverfahren (Scheduling)

2.1 Planen aperiodischer Tasks

- Planen durch Suchen
- Planen nach Fristen

2.2 Planen periodischer Tasks

- Planen nach monotonen Raten

EDF : Earliest Deadline First

Der Prozessor wird derjenigen Task zugeteilt, welche die kürzeste Frist besitzt, d.h. deren Deadline auf der Zeitlinie den kleinsten Wert hat. Wenn es keine rechenbereiten Tasks gibt, bleibt der Prozessor untätig (idle).

- **Eine der verbreitetesten zeitbasierten Planungsstrategien**
- **Anwendbar auf unterbrechbare und nicht unterbrechbare Tasks**
- **Anwendbar in statischen und dynamischen Planungsverfahren**

EDF bei nicht unterbrechbaren Tasks

Voraussetzung: Tasks haben gleiche Bereitzeiten. Klasse: 1| sync, non-preemt| L_{max}

Die Tasks T = (1, ... , n) seien nach ihren Fristen geordnet mit: $1 \leq i \leq j \leq n \Rightarrow d_i < d_j$.
Die Funktion *deadline* (PL_k , i) erzeugt aus dem Plan PL_k durch Einfügen von Task i den Plan PL_{k+1} .

Planungsverfahren EDF_n :

***schedule* (PL , T) :**

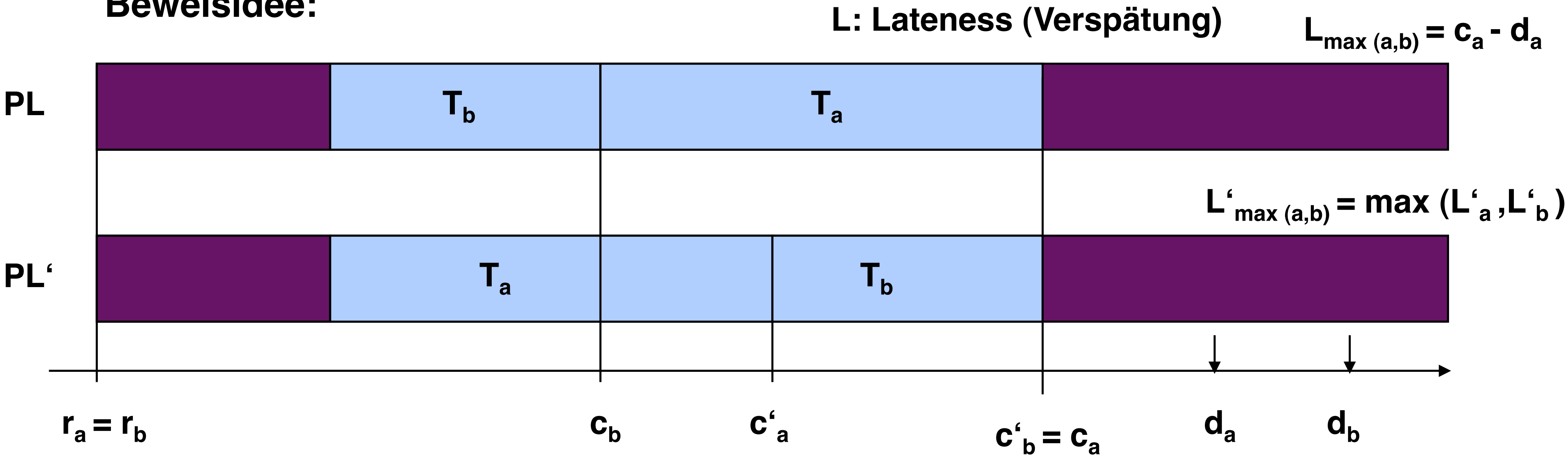
```
        PL  = < > ;
        i = 1;
        WHILE  ( i ≤ n )
             BEGIN
                 PL =  deadline (PL_k , i );
                 i = i + 1
        END
```

Satz (Jackson 1955) EDD: Earliest Due Date:

Gegeben sei eine Menge unabhängiger Tasks. Alle Tasks werden synchron aktiviert. Jeder Algorithmus, der die Tasks in der Reihenfolge nicht absteigender Deadlines ausführt ist optimal im Hinblick auf die Minimisierung der maximalen Verspätung.

Klasse: 1 | non-preemt, sync | L_{max}

Beweisidee:



if $L'_a \geq L'_b$ then $L'_{max\,(a,b)} = c'_a - d_a < c_a - d_a$ (da gilt: $c'_a < c_a$)

if $L'_a \leq L'_b$ then $L'_{max\,(a,b)} = c'_b - d_b < c_a - d_a$ (da gilt: $d_a < d_b$)

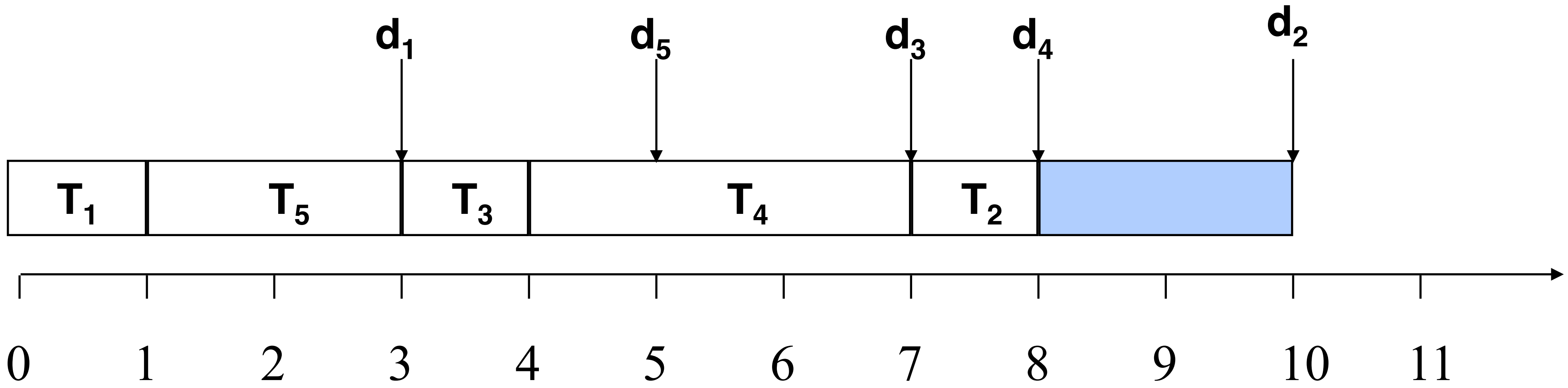
In beiden Fällen ist $L'_{max\,(a,b)} < L_{max\,(a,b)}$

EDD ist bei <u>nicht unterbrechbaren</u> Tasks optimal im Hinblick auf maximale Verspätung, wenn alle Bereitzeiten gleich sind !

Für die Brauchbarkeit eines Plans gilt:
Falls EDD keinen gültigen Plan liefert, gibt es keinen !

Beispiel 1:

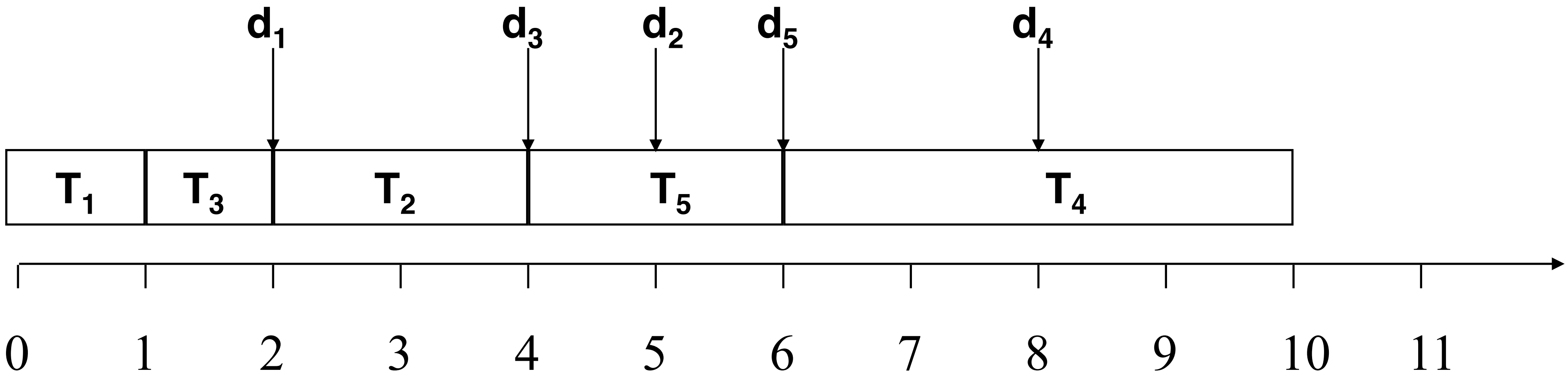
	T_1	T_2	T_3	T_4	T_5
Δe_i	1	1	1	3	2
d_i	3	10	7	8	5



$$\boxed{L_{max} = L_4 = -1}$$

Beispiel 2:

	T_1	T_2	T_3	T_4	T_5
Δe_i	1	2	1	4	2
d_i	2	5	4	8	6

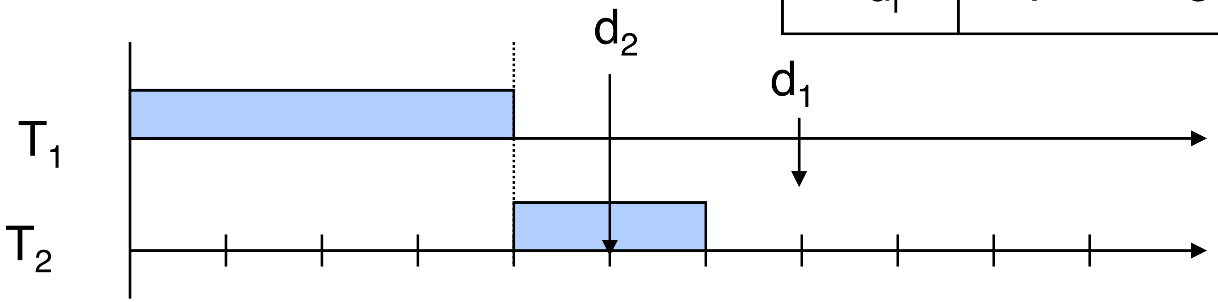


$$\boxed{L_{max} = L_4 = 2}$$

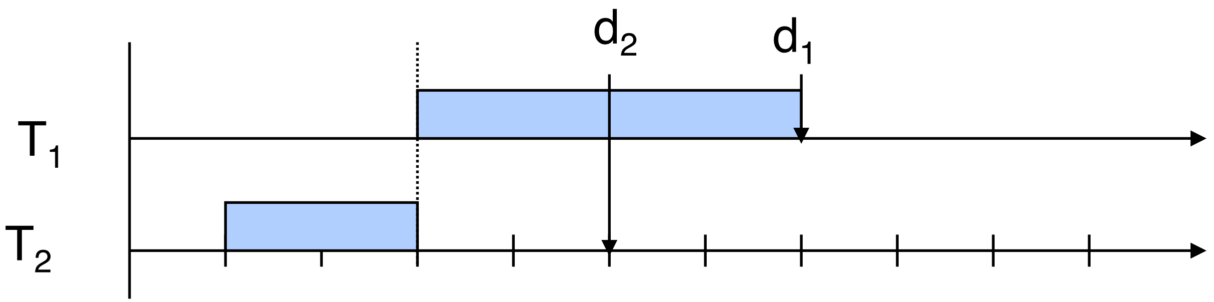
EDF ist im allgemeinen Fall nicht optimal bei einem nicht-unterbrechbaren Task-Modell.

Beispiel:

	T_1	T_2
r_i	0	1
Δe_i	4	2
d_i	7	5



Nach EDF



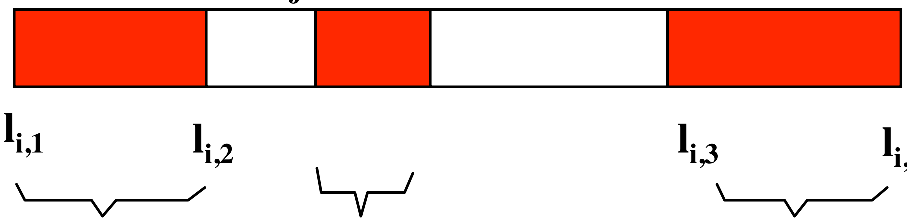
brauchbar

EDF für unterbrechbaren Tasks

Motivation: Planung wenn nicht alle Bereitzeiten gleich sind und Tasks dynamisch bereit werden.

- **Zerlegung der Ausführungszeit einer Task**

$$\sum_{j=1}^{m_i} \Delta l_{i,j} = \Delta e_i$$



m_i = 3 Zeitintervalle

Zeitpunkte $l_{i,1}$ $l_{i,2}$ $l_{i,3}$ $l_{i,4}$

Zeitintervalle $\Delta l_{i,1}$ $\Delta l_{i,2}$ $\Delta l_{i,3}$

Randbedingungen:

1.) $\sum_{j=1}^{m_i} \Delta l_{i,j} = \Delta e_i$

2.) $r_i \leq s_i = l_{i,1} \leq l_{i,mi} + \Delta l_{i,mi} = c_i \leq d_i$.

Plan wird dargestellt als Folge von Tupeln:
$(i, l_{i,j} , \Delta l_{i,j})$

Notwendiges und hinreichendes Kriterium für unterbrechbare Tasks

- Neue Tasks können hinzukommen, deshalb muss der Test bei jeder neuen Task dynamisch ausgeführt werden.
- Tasks sind (einschließlich einer neu hinzukommenden Task) nach aufsteigenden ***Deadlines*** geordnet
- Da Tasks unterbrochen werden können wenn eine neue Task zum Zeitpunkt t bereit wird, kann es sein, dass Tasks noch nicht vollständig ausgeführt sind, d.h. noch ein Rest ΔR_i ausgeführt werden muss.

Die maximale (worst-case) Zeit bis zum Abschluss c zum Zeitpunkt t kann dann Berechnet werden durch:

$$c_i = \Sigma_{k=1 \ldots i} \Delta R_k$$

Deshalb kann ein Plan unter der folgenden Bedingung garantiert werden:

$$\boxed{\forall\, i = 1, \ldots, n \quad \text{gilt:} \quad \Sigma_{k=1 \ldots i} \Delta R_k \leq d_i}$$

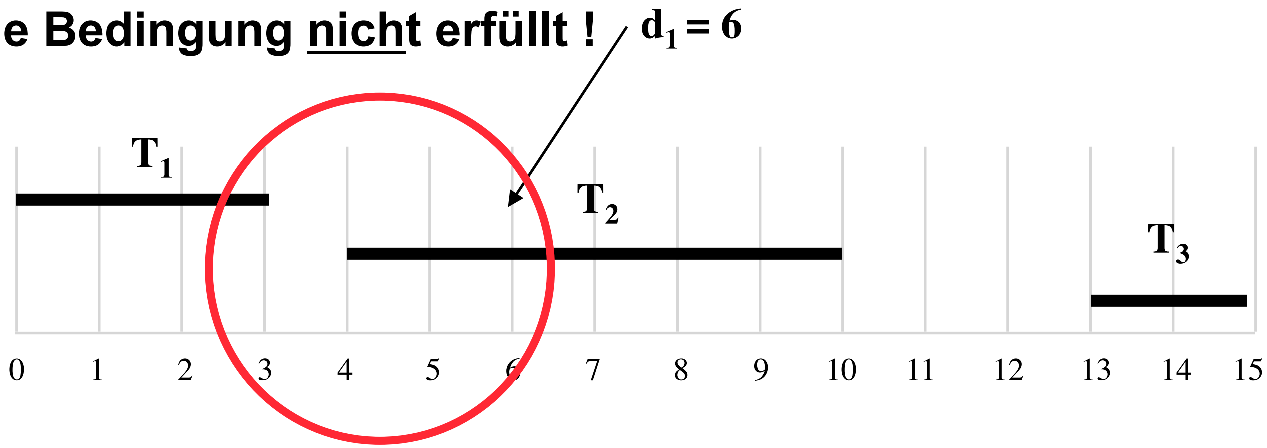
Beispiel: hinreichende aber nicht exakte Bedingung für Einplanbarkeit

Bedingung: $\forall\, i,j\,,\; d_i < d_j :\quad d_i \leq d_j - \Delta e_j$

In beiden Fällen wird die Bedingung <u>nicht</u> erfüllt !

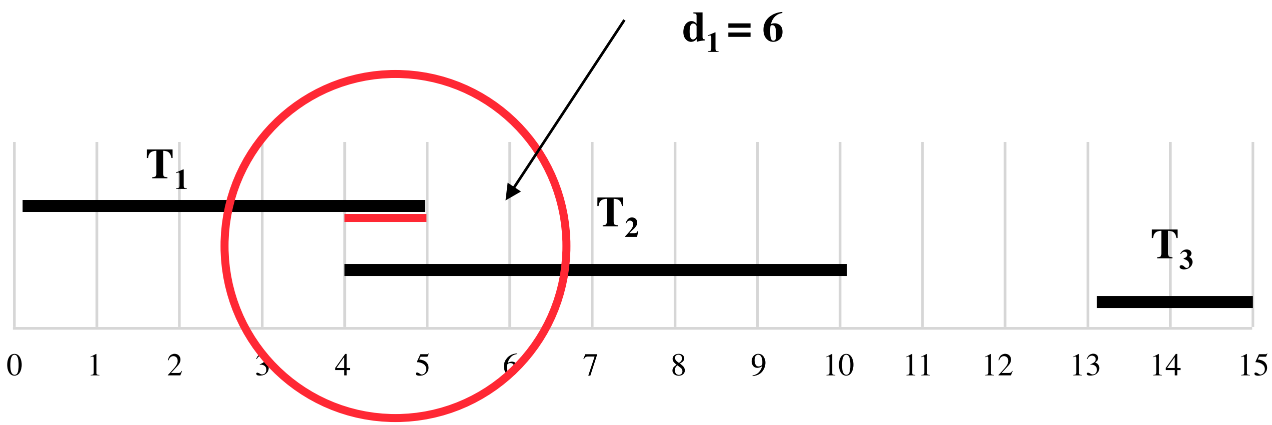
$T_1 : \Delta e_1 = 3, r_1 = 0, d_1 = 6$
$T_2 : \Delta e_2 = 6, r_2 = 4, d_2 = 10$
$T_3 : \Delta e_3 = 2, r_3 = 13, d_3 = 15$

$\Rightarrow$ **planbar !!**



$T_1 : \Delta e_1 = 5, r_1 = 0, d_1 = 6$
$T_2 : \Delta e_2 = 6, r_2 = 4, d_2 = 10$
$T_3 : \Delta e_3 = 2, r_3 = 13, d_3 = 15$

$\Rightarrow$ **nicht planbar !!**



Beispiel 1:

Fall 1:

$$\left.\begin{array}{ll} \Delta e_1 = 3 & d_1 = 4 \\ \Delta e_2 = 4 & d_2 = 7 \\ \Delta e_3 = 3 & d_3 = 9 \\ \Delta e_4 = 5 & d_4 = 15 \end{array}\right\} \begin{array}{l} \textbf{Bed. 1 und 2 erfüllt} \\ \textbf{Bed. 3 nicht erfüllt} \end{array}$$

Fall 2:

$$\left.\begin{array}{ll} \Delta e_1 = 2 & d_1 = 4 \\ \Delta e_2 = 4 & d_2 = 7 \\ \Delta e_3 = 3 & d_3 = 9 \\ \Delta e_4 = 5 & d_4 = 15 \end{array}\right\} \begin{array}{l} \textbf{Bed. 1 und 2 erfüllt} \\ \textbf{Bed. 3 nicht erfüllt} \end{array}$$

Da die Tasks synchron aktiviert werden, kann t=0 angenommen werden.

Fall 1:

T_1	$c_1 = \Delta R_1$	$(\Delta R_1 = 3)$	$= 3$	$\leq d_1$
T_2	$c_2 = c_1 + \Delta R_2$	$(\Delta R_2 = 4)$	$= 7$	$\leq d_2$
T_3	$c_3 = c_2 + \Delta R_3$	$(\Delta R_3 = 3)$	$= 10$	$\geq d_3$ **!!**

Test erkennt: Nicht planbar

Fall 2:

T_1	$c_1 = \Delta R_1$	$(\Delta R_1 = 2)$	$= 2$	$\leq d_1$
T_2	$c_2 = c_1 + \Delta R_2$	$(\Delta R_2 = 4)$	$= 6$	$\leq d_2$
T_3	$c_3 = c_2 + \Delta R_3$	$(\Delta R_3 = 3)$	$= 9$	$\leq d_3$
T_4	$c_4 = c_3 + \Delta R_4$	$(\Delta R_4 = 5)$	$= 14$	$\leq d_4$

Test erkennt: Planbar

EDF Planung für unterbrechbare Tasks

EDF kann zu einem Zeitpunkt t nur auf rechenbereite Tasks angewandt werden, d.h. wenn t zwischen Bereitzeit und Deadline liegt und die Task noch nicht vollständig ausgeführt ist.

Ready (t) ist die nach Fristen geordnete Liste.

$$\text{Ready}(t) = \{T_i \mid r_i \leq t < d_i \ \text{und rest}(i, t) > 0\}$$

$$\text{mit rest}(i, t) = \Delta e_i - \sum_{j \in PL} \Delta l_{i,j}$$

d.h. rest (i, t) umfaßt alle Ausführungszeiten von Task i, die in den, bis zum Zeitpunkt t aufgestellten Plan noch nicht aufgenommen wurden.

Die Funktion *edf (Ready (t))* bestimmt zum Zeitpunkt t die Task aus Ready (t), welche die nächste Deadline besitzt. Da aber jederzeit neue Tasks hinzukommen können, die eingeplant werden müssen, ergibt sich für den nächsten Zeitpunkt nach t, an dem geplant werden muß:

$$\textit{nextavail}(t) = \begin{cases} \min \{r_i \mid r_i > t\} & \text{falls } \{r_i \mid r_i > t\} \neq \emptyset \\ \max \{d_i\} & \text{sonst} \end{cases}$$

$\min \{r_i \mid r_i > t\}$ bezeichnet dabei die nächste neu hinzugekommene Bereitzeit r_i nach dem Zeitpunkt t. Es muß also neu geplant werden, wenn eine neue Task mit der Bereitzeit $r_i > t$ hinzukommt, oder die Deadline d_i der gerade ausgeführten Task erreicht wird.

Das Planungsverfahren endet, wenn:

1.) $\sum_{i \in P} \text{rest}(i, t) = 0$ **, d.h. alle Tasks zu Ende geführt sind**

2.) $\neg\text{feasible PL}(i, t) \Leftrightarrow t + \text{rest}(i, t) > d_i$ **, d.h. wenn eine rechenbereite Task ihre Frist verletzen wird.**

Die Boolsche Funktion: ***allin PL (t)*** **liefert den Wert "wahr", wenn 1.) oder. 2.) gilt (d.h. das Planungsverfahren beendet ist) und den Wert"falsch" sonst (d.h. noch planbare Reste vorhanden sind).**

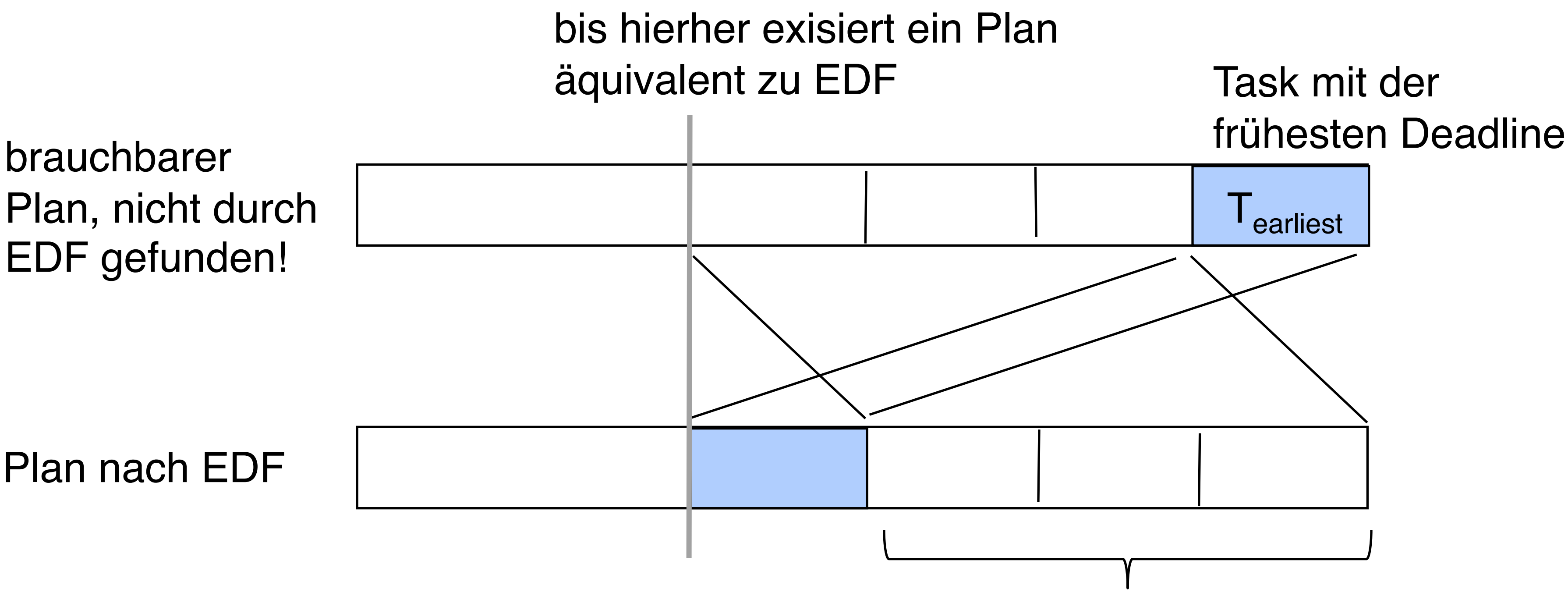
Planungsverfahren für unterbrechbareTasks nach EDF_u :

```
schedule (PL, T):
        PL = < > ;
                t = min { r_i | r_i ∈ T};
                WHILE ¬ allin PL(t)  DO
                        IF Ready (t) = Ø
                          THEN  t = nextavail (t);  ELSE
                              BEGIN
                               i = edf (Ready (t));
                               IF ¬ feasible (i, t) THEN BREAK;
                               Δl = min (rest (i,t), nextavail (t) - t ) ;
                               PL = PL (i, t, Δt);
                               t = t + Δt ;
                        END
```

Satz, Optimalität von EDF (Horn 1974):

Gegeben seien die unterbrechbaren Tasks T {1, . . . ,n}. Wenn es brauchbare Pläne gibt, dann werden sie durch den Planungsalgorithmus EDF_u gefunden

Beweisidee:



Tasks werden verschoben. Da sie alle spätere Deadlines als $T_{earliest}$ besitzen, ist der neue Plan auch brauchbar. EDF hätte das genauso gemacht.