# Grundlagen der Echtzeitplanung

1. Grundlegende Begriffe und Konzepte

2. Planungsverfahren ( Scheduling )

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

   2.2 Planen periodischer Tasks
   - Planen nach monotonen Raten
   - EDF

## Planen nach Spielräumen: Least Laxity First (LLF)

- **Def.: Spielraum (Laxity, relative Deadline)**

  $$\Delta lax_i = ( d_i - r_i ) - \Delta e_i$$

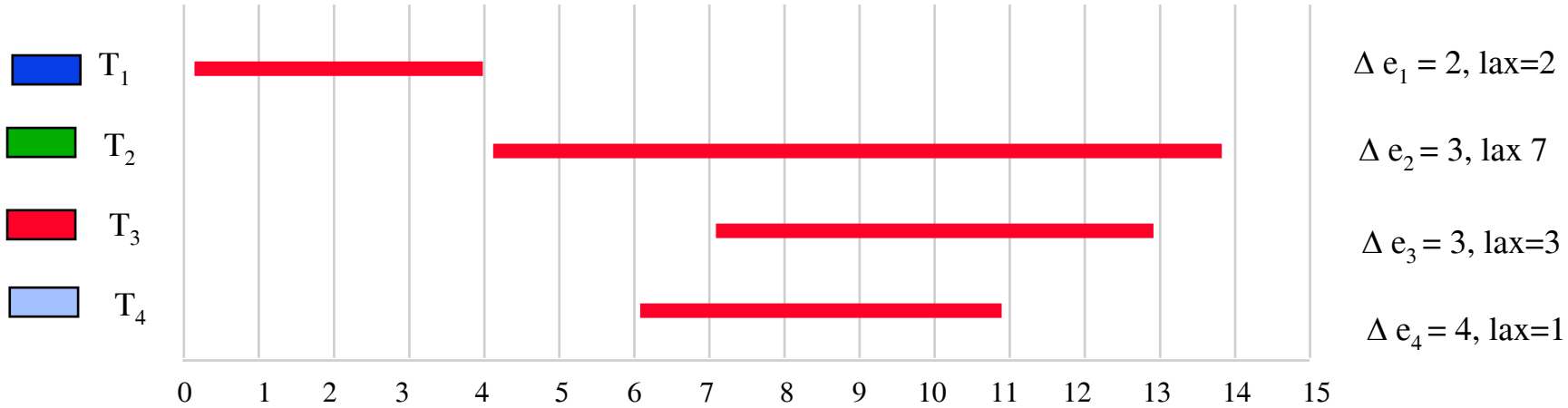
- **Bedingung für die Einplanbarkeit:**

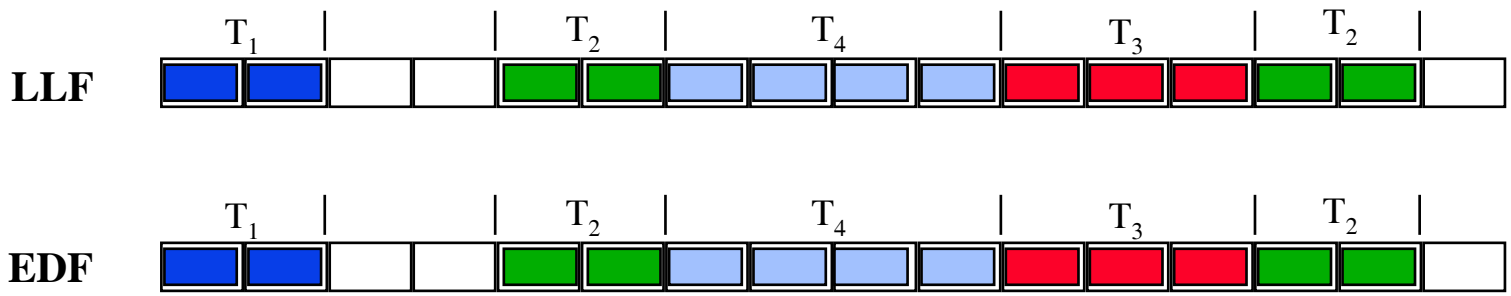
  $$r_i \leq s_i \leq r_i + \Delta lax_i = d_i - \Delta e_i$$

- **Vorgehensweise:**

  **Bei LLF wird unter den rechenbereiten Tasks diejenige ausgewählt, deren Spielraum am kürzesten ist.**

## LLF für unterbrechbare Tasks (1 | preemt | $L_{max}$)

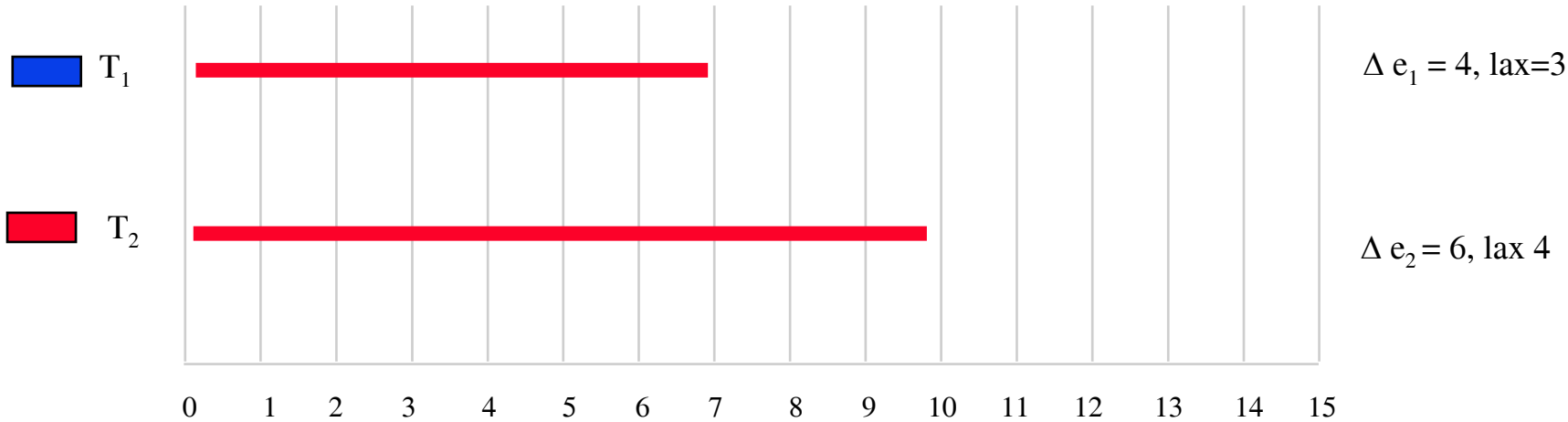


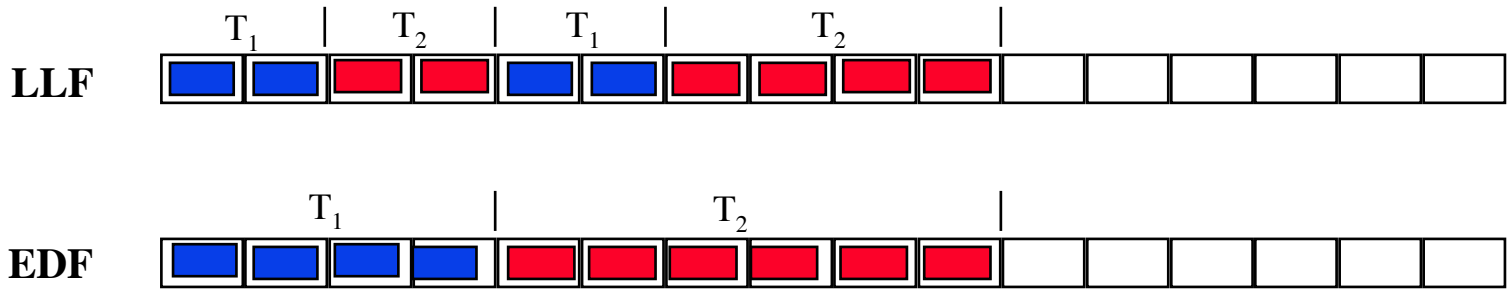




**LLF ist optimal für unterbrechbare Tasks**

# LLF für unterbrechbare Tasks

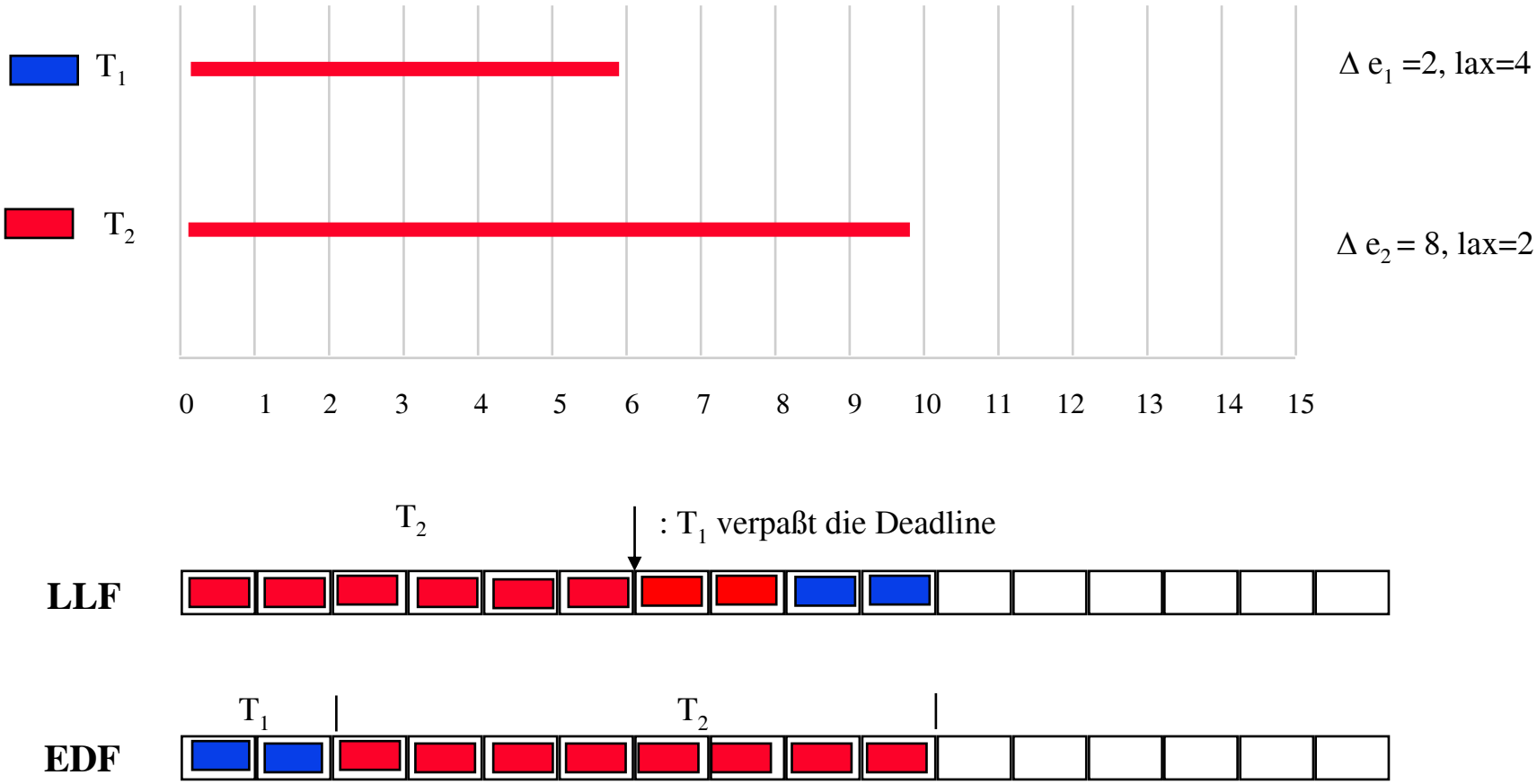
## LLF führt zu häufigeren Unterbrechungen als eine Planung durch EDF



## LLF für nicht unterbrechbare Tasks (1 | non-preemt, sync | $L_{max}$)

## LLF ist nicht optimal für nicht unterbrechbare Tasks



# Zusammenfassung

- **LLF ist nicht optimal für nicht unterbrechbare Prozesse**
- **LLF ist optimal für unterbrechbare Prozesse führt aber zu häufigeren Unterbrechungen als EDF, da sich die Laxity dynamisch ändert.**

# Grundlagen der Echtzeitplanung

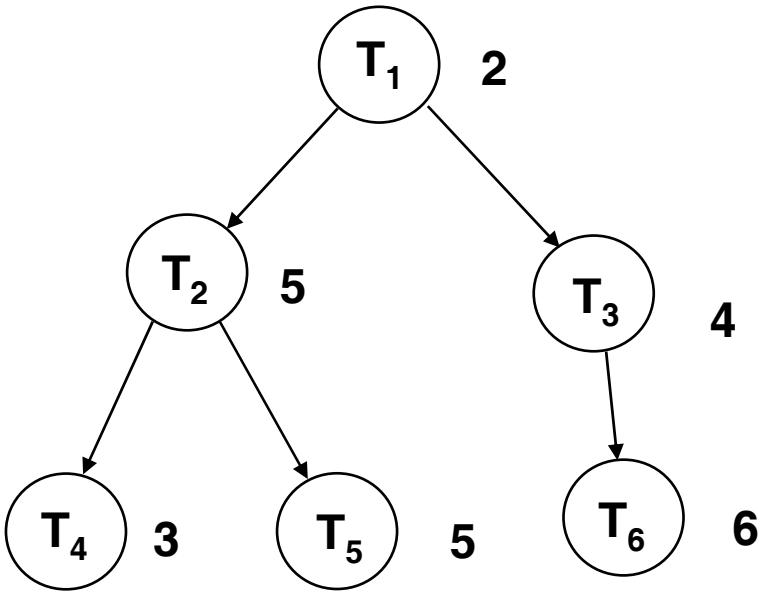
1. Grundlegende Begriffe und Konzepte

2. Planungsverfahren ( Scheduling )

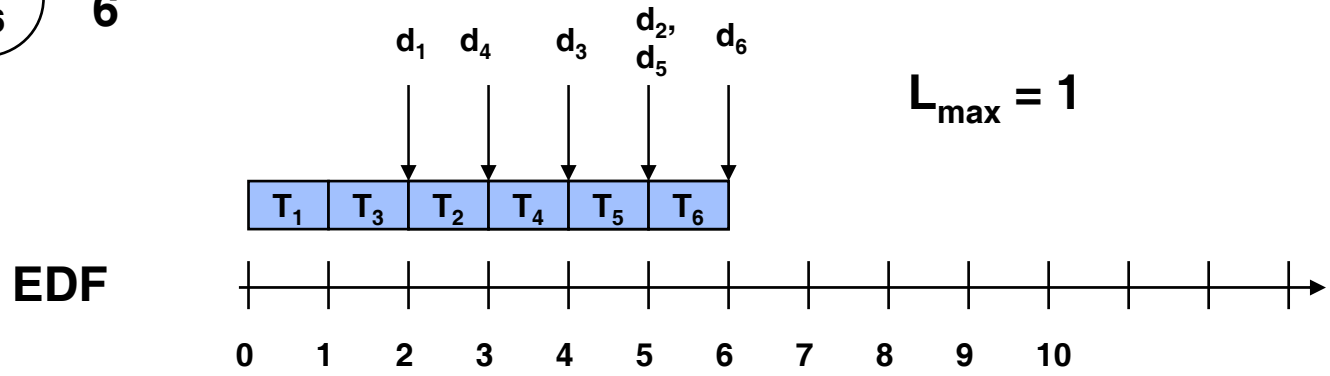
   2.1 Planen aperiodischer Tasks
   - Planen durch Suchen
   - Planen nach Fristen
   - Planen nach Spielräumen
   - **Planen abhängiger Tasks**

   2.2 Planen periodischer Tasks
   - Planen nach monotonen Raten
   - EDF

# Beispiel einer Menge abhängiger Tasks




| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $\Delta e_i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $d_i$ | 2 | 5 | 4 | 3 | 5 | 6 |




$L_{max} = 1$

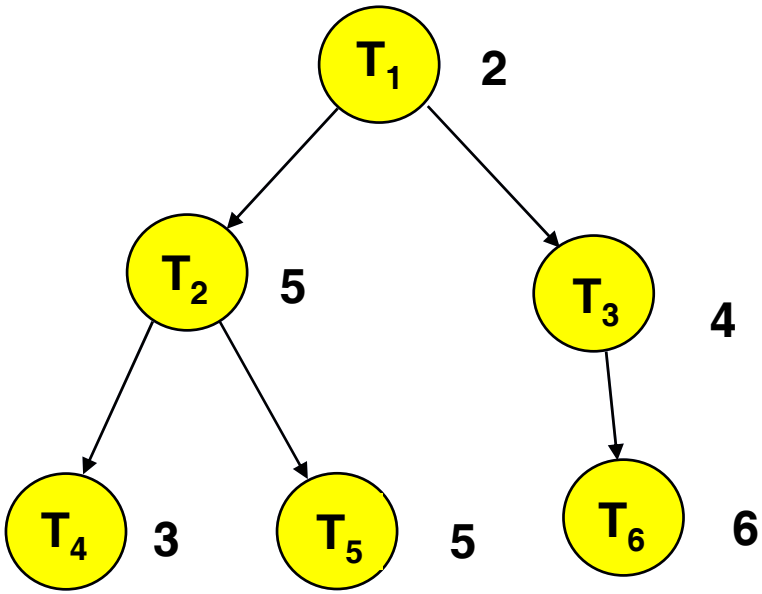
# LDF: Latest Deadline First (Lawler 1973)

## ( 1 | prec, sync | $L_{max}$)

**Gegeben:** Taskmenge abhängiger Tasks $T = \{T_1, .., T_n\}$,
Azyklischer gerichteter Graph, der die Vorrangrelation beschreibt.

**Aus der Menge der Tasks deren Nachfolger bereits alle ausgewählt wurden oder die keinen Nachfolger besitzen wählt LDF die Task mit der spätesten Deadline aus. Die Warteschlange der Tasks wird also in der Reihenfolge der zuletzt auszuführenden Tasks aufgebaut. Zur Laufzeit werden die Tasks vom Kopf der aufgebauten Warteschlange ausgeführt, so dass die Task, die zuletzt in die Warteschlange eingefügt wurde zuerst ausgeführt wird.**

# Planen nach LDF



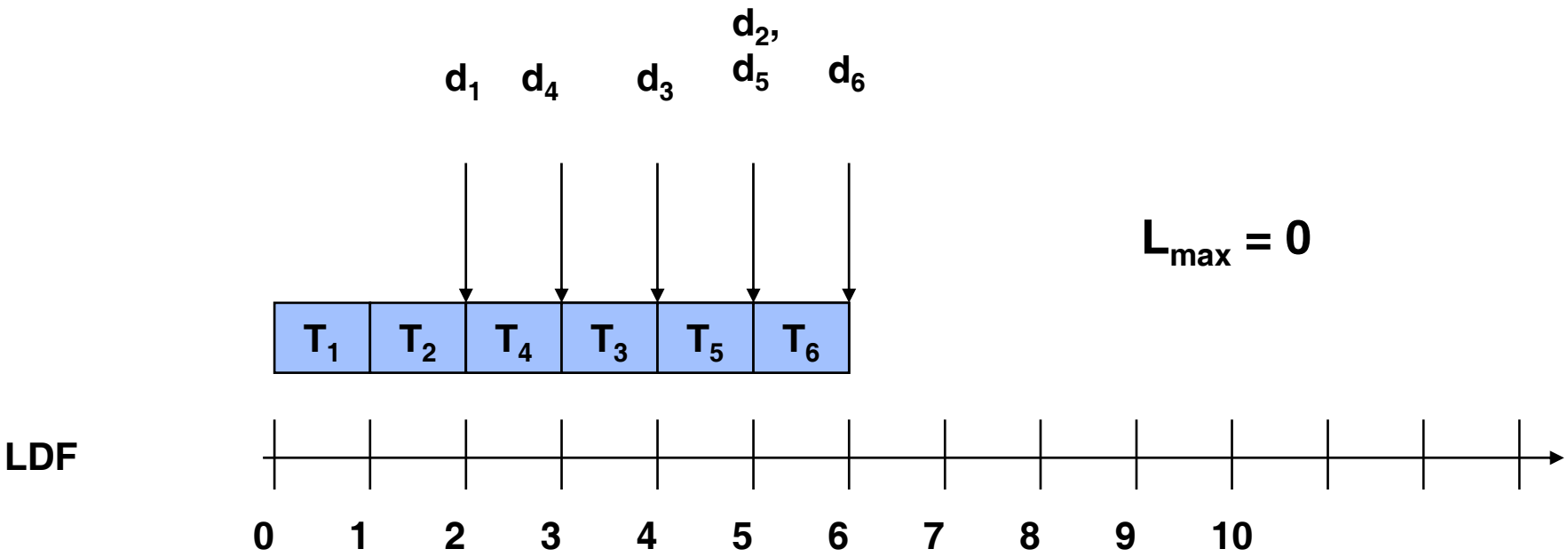

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $\Delta e_i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $d_i$ | 2 | 5 | 4 | 3 | 5 | 6 |

| Schritt | Plan | Noch nicht eingeplante Tasks |
|---|---|---|
| 1. | $T_6$ | $T_5$ $T_3$ $T_4$ $T_2$ $T_1$ |
| 2. | $T_5$ | $T_3$ $T_4$ $T_2$ $T_1$ |
| 3. | $T_3$ | $T_4$ $T_1$ $T_2$ |
| 4. | $T_4$ | $T_2$ $T_1$ |
| 5. | $T_2$ | $T_1$ |
| 6. | $T_1$ | |

Einplanbar nach LDF

Nicht einplanbar nach LDF

## Planen nach LDF

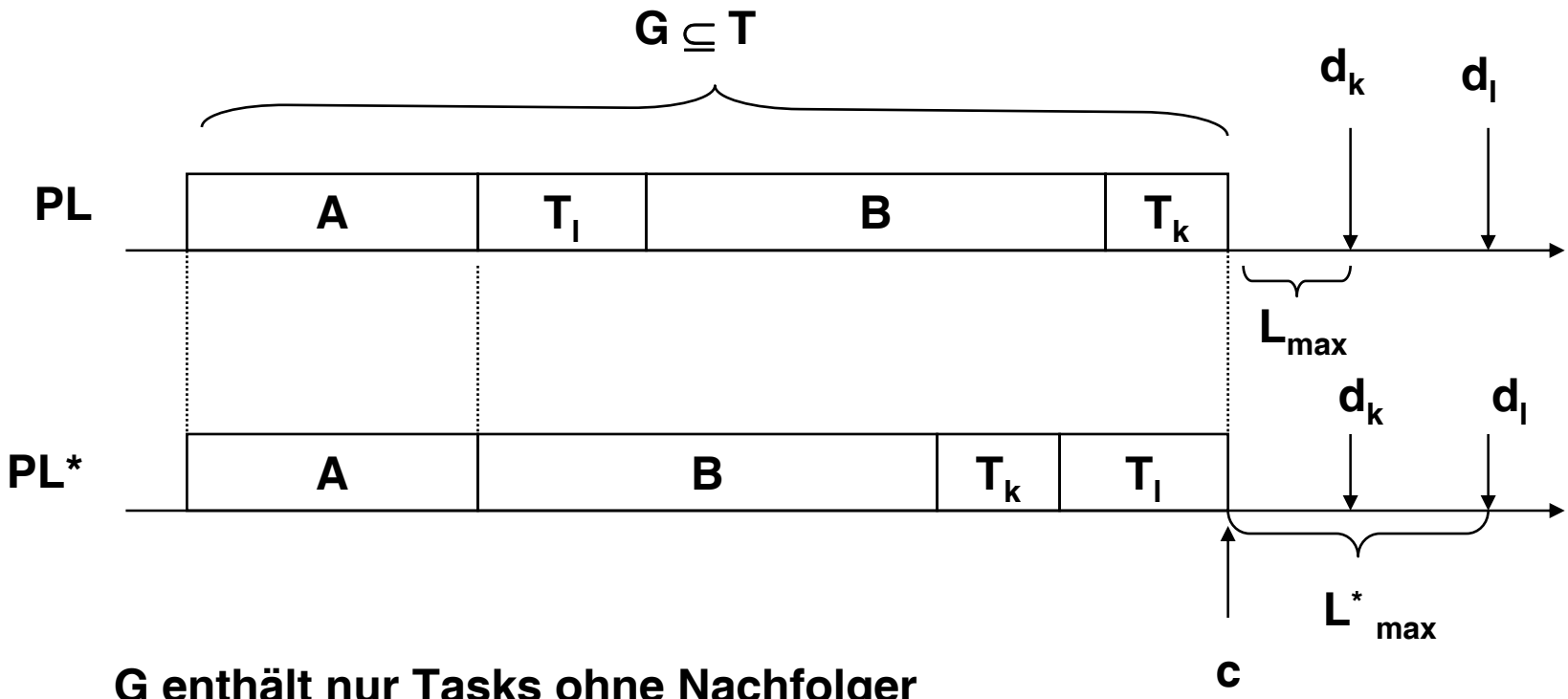



**Komplexität: $O(n^2)$: Für jeden Schritt muss im Abhängigkeitsgraphen nach der Untermenge ohne Nachfolger gesucht werden.**

**LDF ist optimal. (Lawler 1973)**

## Beweis der Optimalität von LDF

**Idee:**




**G enthält nur Tasks ohne Nachfolger**

**$G = A \cup \{T_l\} \cup B \cup \{T_k\}$**

# EDF unter Berücksichtigung der Vorrangrelation

**$(1 \mid prec, preemt \mid L_{max})$**

**Idee: Umwandlung einer Menge abhängiger Tasks in eine Menge unabhängiger Tasks durch Modifikation der Bereitzeiten und der Deadlines.**

**Beobachtung:**

1. **Eine Task kann nicht vor ihrer Bereitzeit ausgeführt werden.**
2. **Eine abhängige Task kann keine Bereitzeit besitzen die kleiner ist als die Bereitzeit der Task von der sie abhängt.**
3. **Eine Task $T_b$, die von einer anderen Task $T_a$ abhängt, kann keine Deadline $d_b \leq d_a$ besitzen.**

**Algorithmus:**

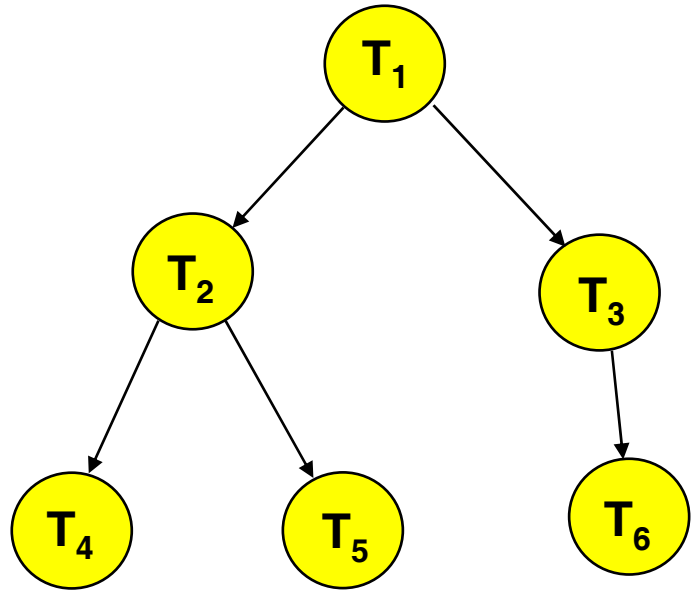
1. **Modifikation der Bereitzeiten**
2. **Modifikation der Deadlines**
3. **Schedule nach EDF erstellen**

**Modifikation der:**

**Bereitzeiten**

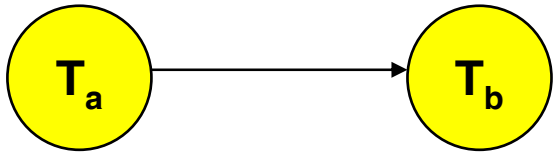
**Deadlines**

**Anfangsknoten**

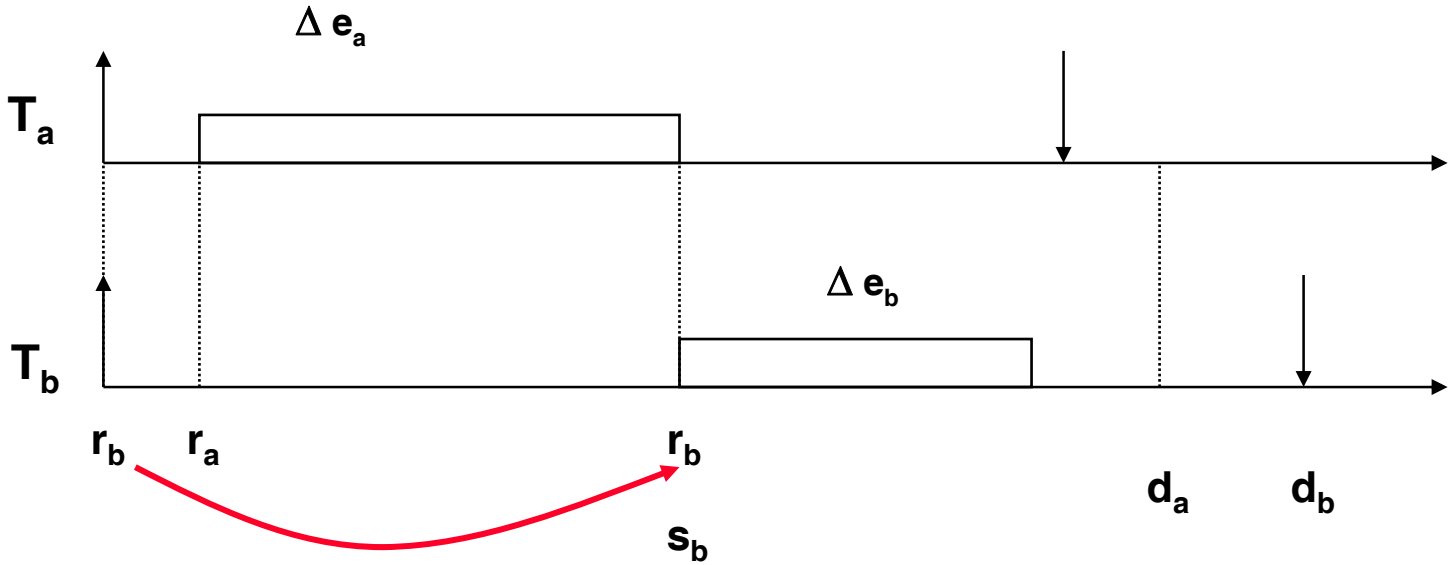



**Endknoten**

## Modifikation der Bereitzeiten




$r_b$ ersetzen durch: $\max(r_b, r_a + \Delta e_a)$




$$\begin{cases} s_b \geq r_b \\ s_b \geq r_a + \Delta e_a \end{cases}$$

# Algorithmus zur Modifikation der Bereitzeiten:

1. Für einen beliebige Anfangsknoten des Vorrang-Graphen setze $r^*_i = r_i$ .

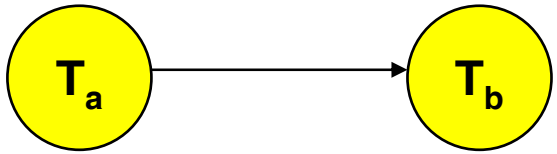
2. Wähle eine Task $T_i$ , deren Bereitzeit (noch) nicht modifiziert wurde, aber deren Vorgänger alle modifizierte Bereitzeiten besitzen.

   Wenn es keine solche Task gibt: EXIT.

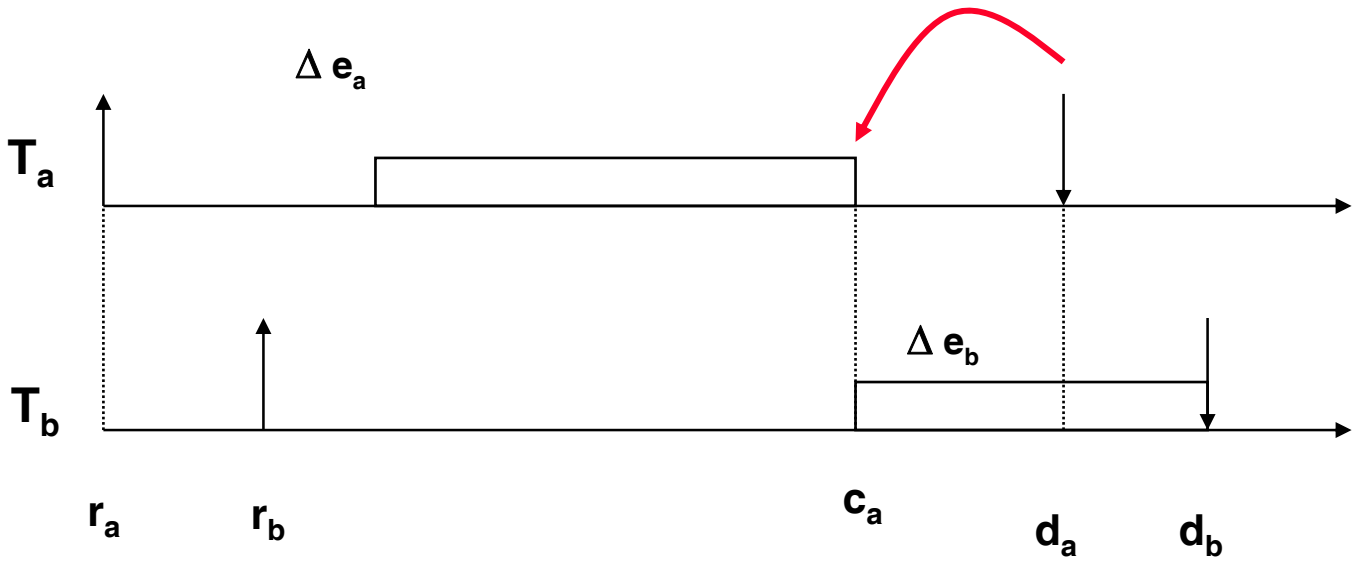
3. Setze $r^*_i = \max [r_i , \max(r^*_v + \Delta e_v : T_v \rightarrow T_i )]$.

4. Gehe nach Schritt 2.

## Modifikation der Deadlines




$d_a$ ersetzen durch: $\min(d_a, d_b - \Delta e_b)$




$$\begin{cases} c_a \leq d_a \\ c_a \leq d_b + \Delta e_b \end{cases}$$

# Algorithmus zur Modifikation der Deadlines:

1. Für einen beliebige Endknoten des Vorrang-Graphen setze $d^*_i = d_i$ .

2. Wähle eine Task $T_i$ , deren Deadline (noch) nicht modifiziert wurde, aber deren unmittelbare Nachfolger alle modifizierte Deadlines besitzen.

   Wenn es keine solche Task gibt: EXIT.

3. Setze $d^*_i = \min [d_i , \min(d^*_N - \Delta e_N : T_i \rightarrow T_N )]$.

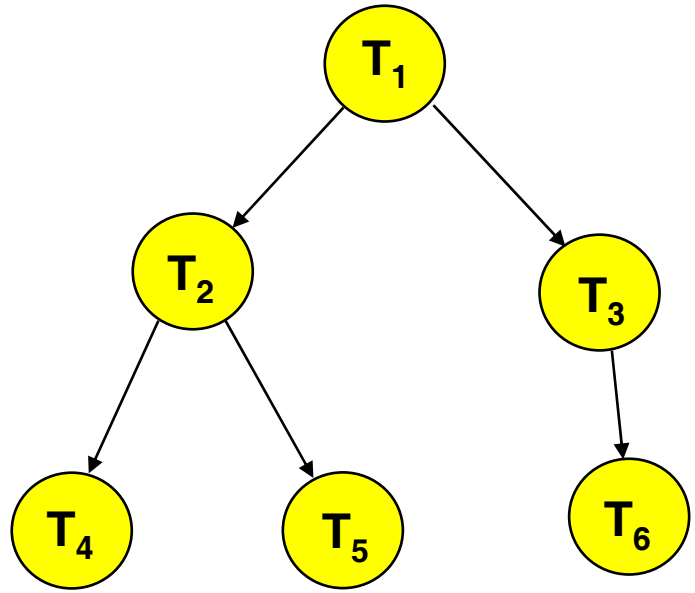
4. Gehe nach Schritt 2.

**Modifikation der:**

**Bereitzeiten**

**Deadlines**

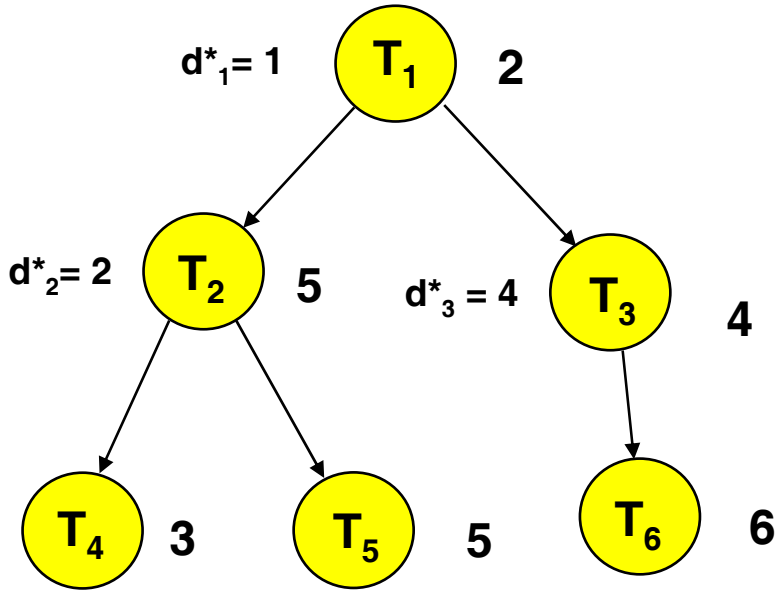
**Anfangsknoten**




**Endknoten**

**ursprüngliche Taskparameter**

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $\Delta e_i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $r_i$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $d_i$ | 2 | 5 | 4 | 3 | 5 | 6 |




**modifizierte Taskparameter**

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $\Delta e_i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $r_i$ | 0 | 1 | 1 | 2 | 2 | 2 |
| $d_i$ | 1 | 2 | 4 | 3 | 5 | 6 |

$$r^*_i = \max [r_i \ , \max(r^*_v + \Delta e_v : T_v \rightarrow T_i )]$$

$$d^*_i = \min [d_i \ , \min(d^*_N - \Delta e_N : T_i \rightarrow T_N )]$$

**modifizierte Taskparameter**

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ |
|---|---|---|---|---|---|---|
| $\Delta e_i$ | 1 | 1 | 1 | 1 | 1 | 1 |
| $r_i$ | 0 | 1 | 1 | 2 | 2 | 2 |
| $d_i$ | 1 | 2 | 4 | 3 | 5 | 6 |

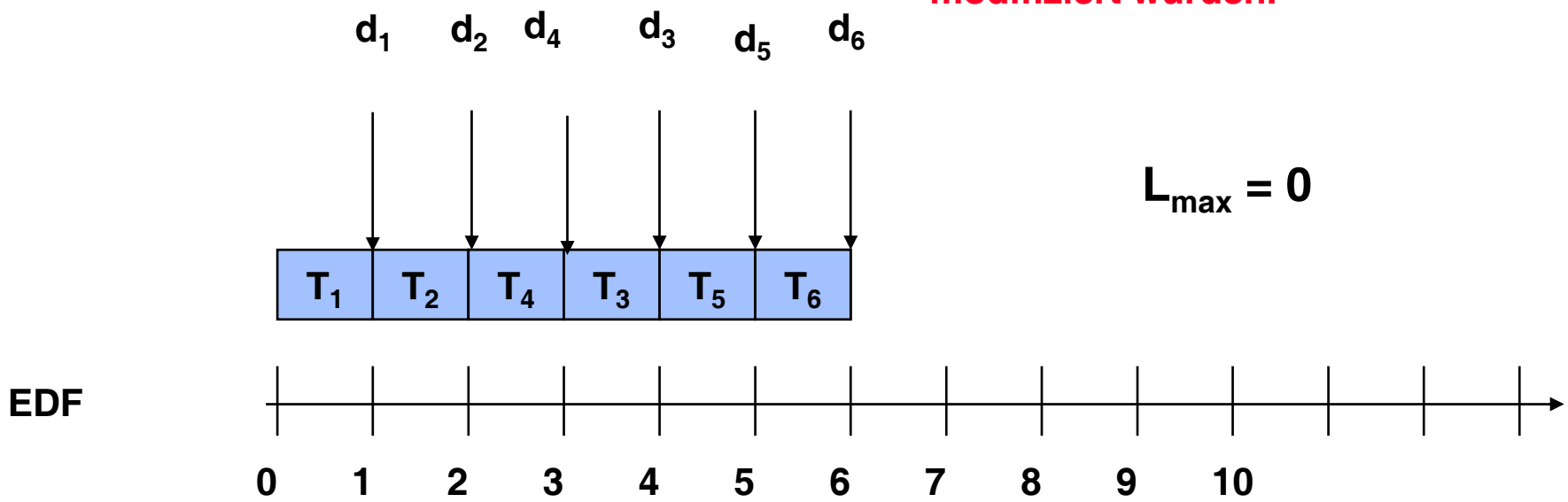
**Beweisidee:**

**1.)**

**eine abhängige Task kann nie vor der Task starten, von der sie abhängig ist, da die Bereitzeiten entsprechend modifiziert wurden**

**2.)**

**eine abhängige Task kann nie die Task unterbrechen, von der sie abhängig ist, da die Deadlines entsprechend modifiziert wurden.**




$L_{max} = 0$

| | sync. activation | preemptive async. activation | non-preemptive async. activation |
|---|---|---|---|
| independent | **EDD** (Jackson '55)<br>$O(n \log n)$<br>Optimal | **EDF** (Horn '74)<br>$O(n^2)$<br>Optimal | ***Tree search***<br>(Bratley '71)<br>$O(n\, n!)$<br>Optimal |
| precedence constraints | **LDF** (Lawler '73)<br>$O(n^2)$<br>Optimal | **EDF** *<br>(Chetto et al. '90)<br>$O(n^2)$<br>Optimal | **Spring** (Stankovic & Ramamritham '87)<br>$O(n^2)$<br>Heuristic |