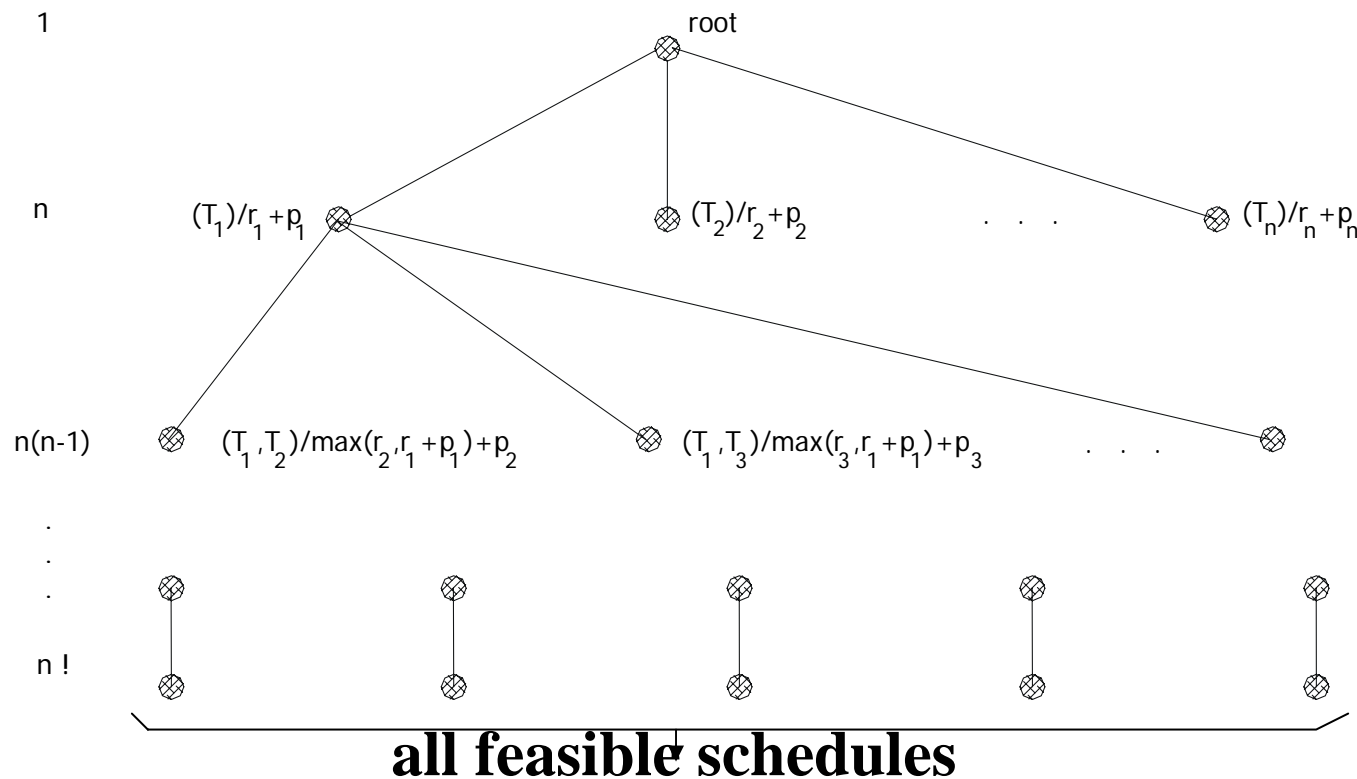


# Single processor scheduling

$$C_{\max}$$

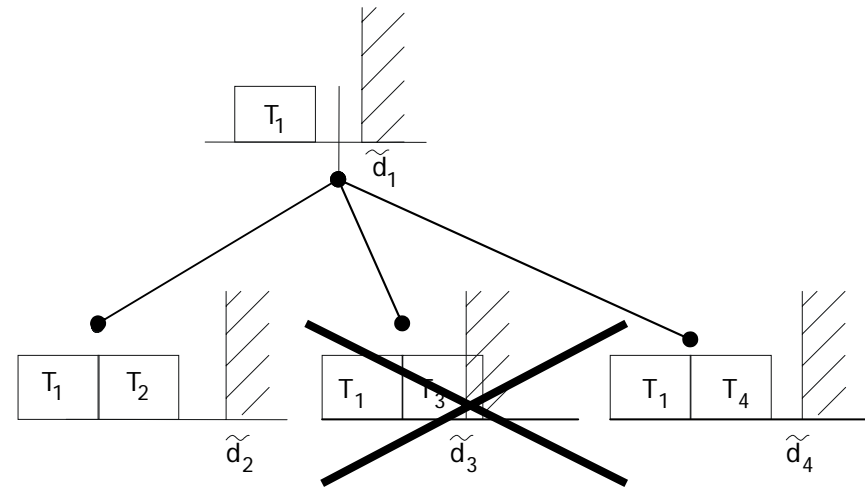
- $1|\text{prec}|C_{\max}$  – simple
  - if tasks are assigned in whatever order in accordance with precedence relation, then  $C_{\max} = \sum p_j$
- $1||C_{\max}$  – simple
- $1|r_j|C_{\max}$  – simple
  - tasks are scheduled in order of nondecreasing release times
- $1|d_j|C_{\max}$  – simple
  - tasks are scheduled in order of nondecreasing deadlines (**EDF** – earliest deadline first)
  - EDF provides optimal solution iff there exists a schedule that meets all the deadlines

- $1 \mid r_j, d_j \sim \mid C_{\max}$  – NP hard problem
  - transformation from the 3-PARTITION problem
  - polynomial algorithm can be found if  $p_j=1$
  - general problem can be solved by applying **Branch&Bound** algorithm by **Bratley**



**(i) exceeding deadlines**

– if completion time associated with at least one of the nodes under node  $v$  level  $k-1$  then all nodes under  $v$  can be eliminated

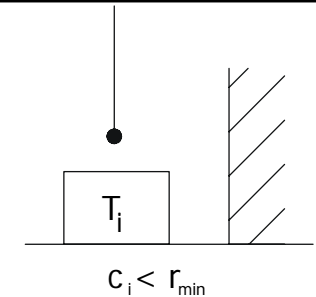


due to this vertex it is needed to eliminate both “brother” vertices

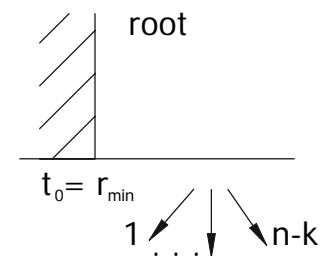
**(ii) problem decomposition –**

if the completion time  $C_i$  of all scheduled tasks is less than or equal to smallest release time of all unscheduled tasks

situation at level  $k$



it remains to schedule  $(n-k)$  tasks



# Optimality test of Bratley's algorithm

**block** is a group of tasks such that the first task starts at its release time and all the following tasks to the end of the schedule are processed without idle time

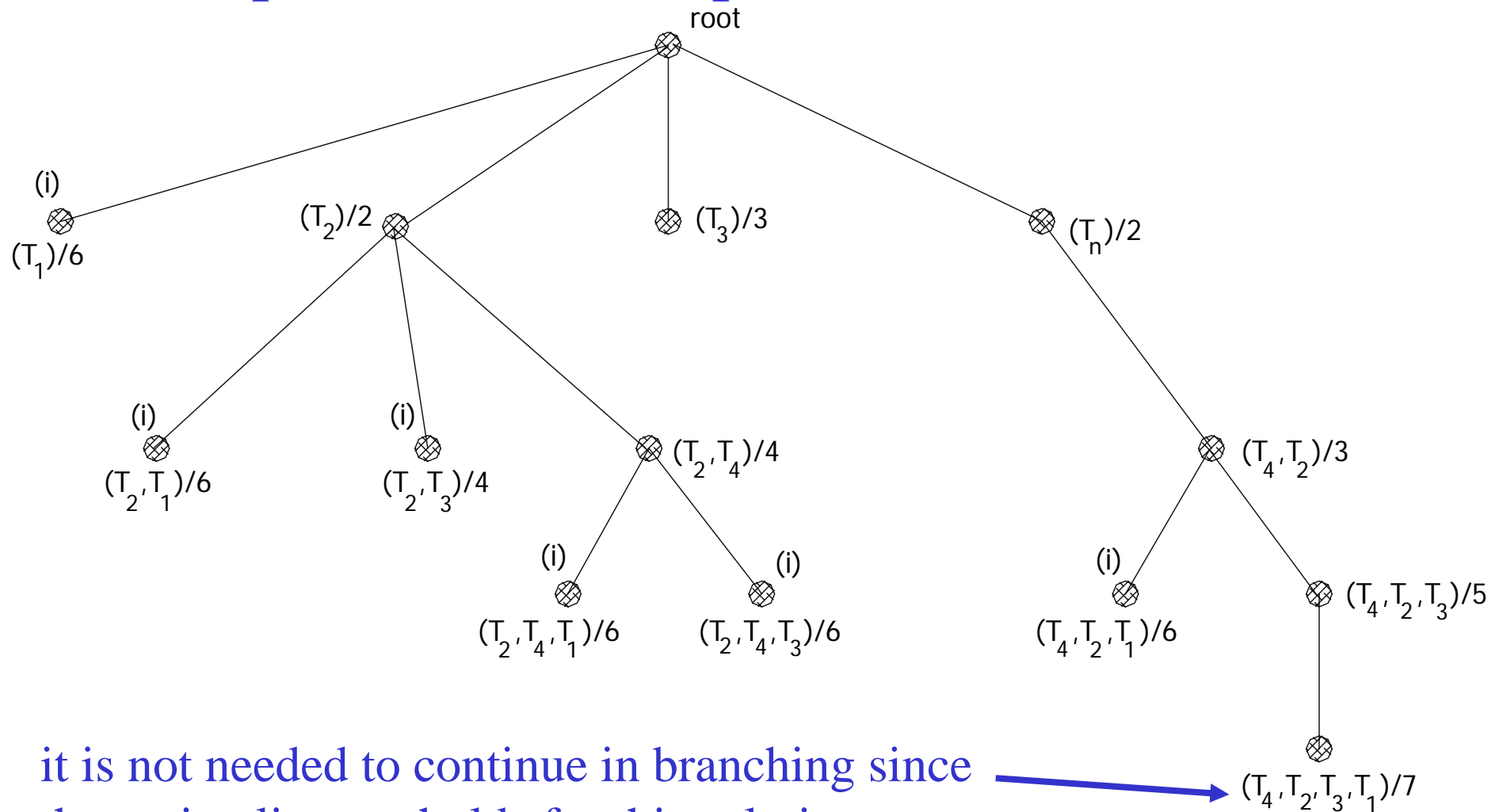
block satisfies **release time property** if release time of all tasks in the block are greater or equal to the release time of the first task in the block

Lemma: A schedule is optimal iff it contains a block that satisfies the release time property

Proof:

- if part (each schedule with block satisfying RTP is optimal) - follows from the definition of RTP
- only if part (each optimal schedule has block satisfying RTP) – by contradiction – suppose schedules that do not have block with RTP, none of them is optimal

Example:  $r = [4, 1, 1, 0]$ ,  $p = [2, 1, 2, 2]$ ,  $d^{\sim} = [7, 5, 6, 4]$



it is not needed to continue in branching since the optimality test holds for this solution

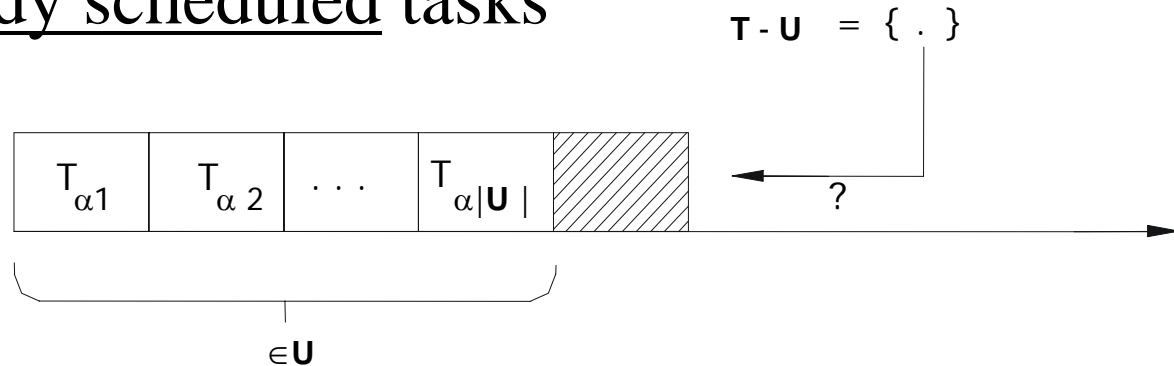
$$\sum w_j C_j$$

- $1 \parallel \sum C_j$  – Shortest Processing Time (SPT) rule  
- tasks are scheduled in order of nondecreasing  $p_j$
- $1 \parallel \sum w_j C_j$  – Weighted SPT rule  
– tasks are scheduled in order of nondecreasing  $p_j/w_j$
- $1|r_j| \sum C_j$ ,  $1|r_j| \sum w_j C_j$  – NP hard
- $1|pmtn, r_j| \sum C_j$  – solvable by modified SPT
- $1|pmtn, r_j| \sum w_j C_j$  – NP hard
- $1|d_j \sim| \sum C_j$ , – solvable by modified SPT
- $1|d_j \sim| \sum w_j C_j$  – NP hard
- $1|prec| \sum w_j C_j$  – NP hard

- $1|r_j| \Sigma C_j$  – NP hard

two heuristic algorithms based on two criteria for adding a task to an existing partial schedule

**U** – set of already scheduled tasks



Earliest Completion Time (ECT) rule

- 1) select task  $T_j$  with  $\min\{C_j | T_j \in \mathbf{T-U}\}$
- 2) assign  $T_j$  to **U**
- 3) calculate  $s_j = \max\{r_j, C_{\alpha|U|}\}$  and  $C_j = s_j + p_j$

Earliest Starting Time (EST) rule

- 1) select task  $T_j$  with  $\min\{s_j | T_j \in \mathbf{T-U}\}$
- 2,3) ....

- $1|d_j \sim| \sum w_j C_j$  – NP hard

heuristic algorithm: Smith's Backward rule

begin

$p := \sum p_j$

while  $\mathbf{T} \neq \emptyset$

$\mathbf{T}_p := \{T_j | T_j \in \mathbf{T}, d_j \sim \geq p\}$  //set of tasks with  
less urgent deadlines

select  $T_j \in \mathbf{T}_p$  with maximal  $p_j/w_j$  //WSTP in  $\mathbf{T}_p$   
schedule  $T_j$  at n-th position //backward rule

$n := n - 1;$

$\mathbf{T} := \mathbf{T} - \{T_j\};$

$p := p - p_j;$

endwhile

end

## Smith's Backward rule - cont

algorithm complexity  $O(n \log n)$

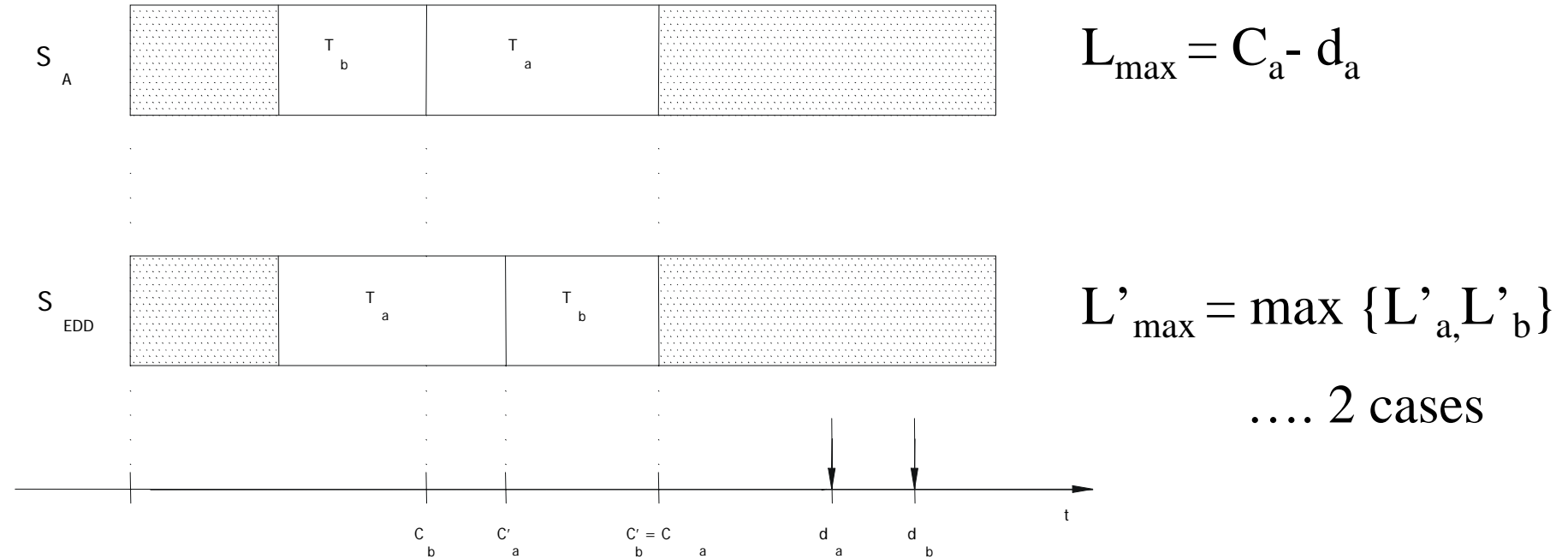
this heuristic is exact if:

- (i) unit processing time  $1 \mid p_j=1, d_j \sim \mid \Sigma w_j C_j$
- (ii) unit weights  $1 \mid d_j \sim \mid \Sigma C_j$
- (iii) agreeable weights, i.e. problems where  $p_i \leq p_j$  implies  $w_i \geq w_j$  for  $i, j = 1, \dots, n$

- $1 \mid \text{prec} \mid \Sigma w_j C_j$  – NP hard – **formulation of 0/1 programming**

# $L_{\max}$

- 1 ||  $L_{\max}$  – Earliest Due Date (EDD) first - Jackson
  - tasks are scheduled in order of nondecreasing due dates
  - optimality can be proven by a simple interchange:
    - Let  $S_A$  be a schedule produced by algorithm  $A$
    - If  $A$  is different than EDD, then there exist two tasks  $T_a$  and  $T_b$  with  $d_a \leq d_b$ , such that  $T_b$  immediately precedes  $T_a$  in  $S_A$
    - Interchanging the position of  $T_a$  and  $T_b$  cannot increase  $L_{\max}$ .
    - By finite number of transpositions  $S_A$  is changed to  $S_{EDD}$ .



1. if  $L'_a \geq L'_b$  then  $L'_{\max} = C'_a - d_a < C_a - d_a$
2. if  $L'_a \leq L'_b$  then  $L'_{\max} = C'_b - d_b < C_a - d_a$

in both cases:  $L'_{\max} < L_{\max}$

# $L_{\max}$

- $1|r_j| L_{\max}$  – NP hard
- $1|r_j, p_j=1| L_{\max}$  – can be solved by modified EDD
- $1|pmtn, r_j| L_{\max}$  – modification of EDD by Horn
- $1|pmtn, r_j, d_j=d_j^{\sim}| L_{\max}$  – the same Horn's algorithm using EDF
- $1|pmtn, prec, r_j, d_j=d_j^{\sim}| L_{\max}$  – transformation to independent task set and then EDF

- $1|r_j, p_j=1| L_{\max}$  – can be solved by **modified EDD**

begin

$t:=0$ ;

  while  $\mathbf{T} \neq \emptyset$

$t:=\max\{t, \min_{T_j \in \mathbf{T}}\{r_j\}\}$

    //shift time if no  
    task is executable

$\mathbf{T}_r := \{T_j | T_j \in \mathbf{T}, r_j \leq t\}$

    //set of execut. tasks

    select  $T_j \in \mathbf{T}_r$  with minimal  $d_j$

    //EDD in  $\mathbf{T}_r$

    schedule  $T_j$  at instant  $t$

$\mathbf{T} := \mathbf{T} - \{T_j\}$ ;

$t := t + 1$ ;

  endwhile

end

- $1|pmtn, r_j| L_{\max}$  – modification of EDD by Horn

Theorem: given a set of  $n$  independent tasks with arbitrary release times, any algorithm that at any instant executes the task with earliest absolute due date among all the ready tasks is optimal with respect to minimizing  $L_{\max}$

When we assume  $d_j \sim = d_j$  then the same applies for EDF (Earliest Deadline First) since it optimizes both:

- **schedulability** – if there exists a feasible schedule ( $L_{\max} \leq 0$ ) for given instance, then EDF is able to find it
- $L_{\max}$  - EDF minimizes  $L_{\max}$

# EDF optimality

- Let  $S_A$  be a schedule produced by algorithm  $A$  and  $S_{EDF}$  by EDF.
- Without loss of generality  $S_A$  can be divided into unit time slices.
- Let  $i(t)$  is id of task executing slice  $t$
- Let  $j(t)$  is id of ready task with earliest deadline at time  $t$
- If  $S_A \neq S_{EDF}$  then there is slice  $t$  such that  $i(t) \neq j(t)$
- Interchanging position of  $i(t)$  and  $j(t)$  cannot increase maximum lateness.
- If schedule  $S_A$  starts at time  $t=0$  and  $D$  is the latest deadline then  $S_{EDF}$  is obtained from  $S_A$  by at most  $D$  transpositions.

Figure Buttazzo page 58

# EDF preserves schedulability

- .....pravdepodobne lze nahradit uvahou, ze schedulability odpovida splneni podminky  $L_{\max} \leq 0$ , (neboli kdyz nase optimalizace nalezne reseni s  $L_{\max} > 0$ , tak je zrejme ze to neni “schedulable” jelikoz neexistuje zadny rozvrh s mensim  $L_{\max}$  ...obracene je to trivialni – pokud nalezneme rozvrh s  $L_{\max} \leq 0$  pak je to rozvrhnutelne)
- Let  $\tau(t)$  is the time ( $\tau(t) > t$ ) at which next slice of task  $j(t)$  begins its execution in current schedule
- At any instant each slice of  $S_A$  can be:  
either brought forward – schedulability is obviously preserved  
or postponed - if slice of  $T_i$  is postponed at  $\tau(t)$  and  $S_A$  is schedulable then it must be  $(\tau(t)+1) \leq \tilde{d}(t)$  being  $\tilde{d}(t)$  the earliest deadline. Since  $\tilde{d}(t) \leq \tilde{d}_i$  for any unexecuted  $T_i$  then we have  $(\tau(t)+1) \leq \tilde{d}_i$ , which guarantees schedulability of the slice postponed at  $\tau(t)$ .

- $1 | \text{pmtn, prec, } r_j, d_j \sim | L_{\max}$  – Chetto, Silly, Bouchentouf

Basic idea is to transform a set  $T$  of dependent tasks into  $T^*$  of independent tasks by modification of timing parameters:

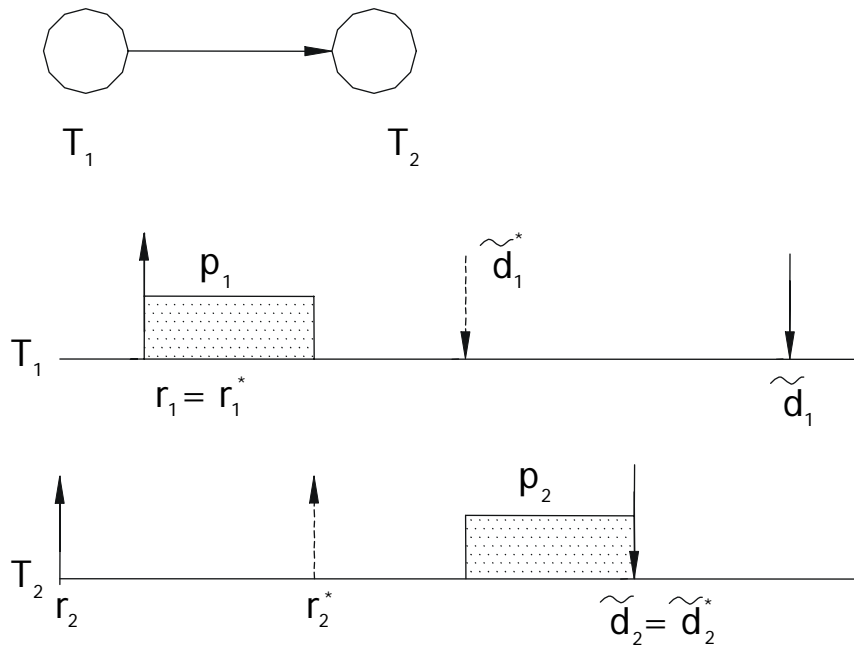
– modification of the release times

- 1) For any task without predecessors set  $r_j^* = r_j$
- 2) Select a task  $T_j$  such that its release time has not been modified but the release times of all immediate predecessors  $T_h$  have been modified. If no such task exists, exit.
- 3) set  $r_j^* = \max \{ r_j, \max \{ r_h^* + p_h \mid T_h \text{ is immediate predec. of } T_j \} \}$  and skip to step 2.

- modification of deadlines
  - 1) For any task without successors set  $d_j^{\sim*} = d_j^{\sim}$
  - 2) Select a task  $T_j$  such that its deadline has not been modified but the deadlines of all immediate successors  $T_k$  have been modified. If no such task exists, exit.
  - 3) set  $d_j^{\sim*} = \min\{d_j^{\sim}, \min\{d_k^{\sim*} - p_k \mid T_j \text{ is immediate suc. of } T_k\}\}$  and skip to step 2.
- EDF is executed on  $T^*$

### Proof of optimality

- since  $r_j^* \geq r_j$  and  $d_j^{\sim*} \leq d_j^{\sim}$  the schedulability of  $T^*$  implies also schedulability of  $T$
- precedence constraints are not violated since due to EDF and modification of timing parameters the scheduled tasks are ordered in the same way as given by the precedence relations



- $r_1^* = r_1$
- $r_2^* = r_1 + p_1$
- $d_1^{\sim*} = d_2^{\sim} - p_2$
- $d_2^{\sim*} = d_2^{\sim}$