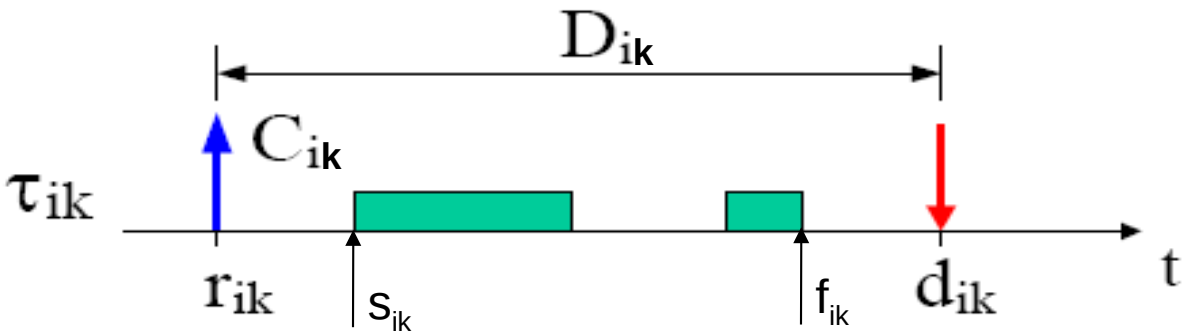
Cenni di schedulazione in tempo reale

E.Mumolo

mumolo@units.it

Task in tempo reale

- Un task t_i è una sequenza di processi in tempo reale τ_{ik} ciascuno caratterizzato da
  - un tempo d'arrivo r_{ik}
  - un tempo di inizio esecuzione s_{ik}
  - un tempo di fine esecuzione f_{ik}
  - una deadline assoluta d_{ik},
  - una deadline relativa D_{ik},
  - da un tempo di esecuzione C_{ik}

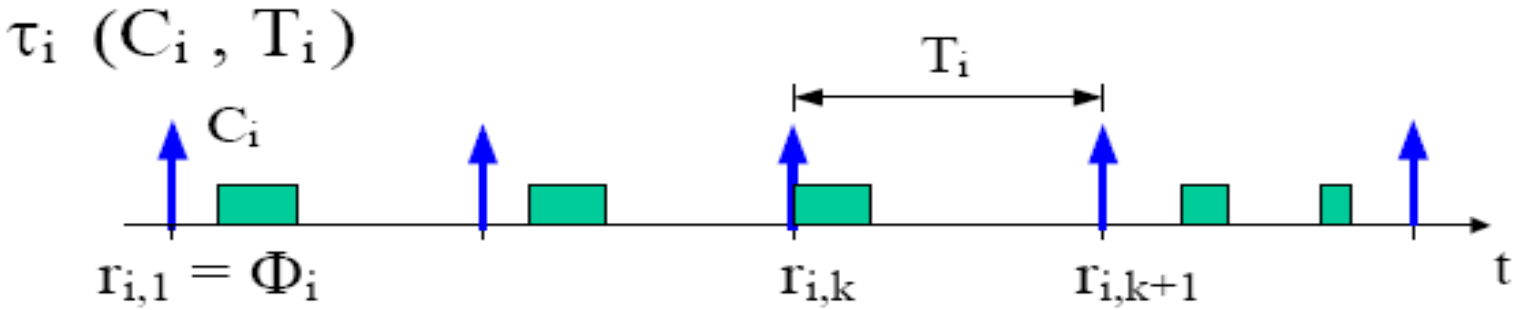


Task periodici

- Triggerati a periodi fissi da un timer
- Consistono in una sequenza infinita di attività identiche, chiamate istanze.
- Ciascuna istanza è caratterizzata da un periodo T e da un tempo di calcolo C

Task periodico τ_i

$$\begin{cases} r_{i1} = \Phi_i \\ r_{i,k+1} = r_{i,k} + T_i \end{cases}$$

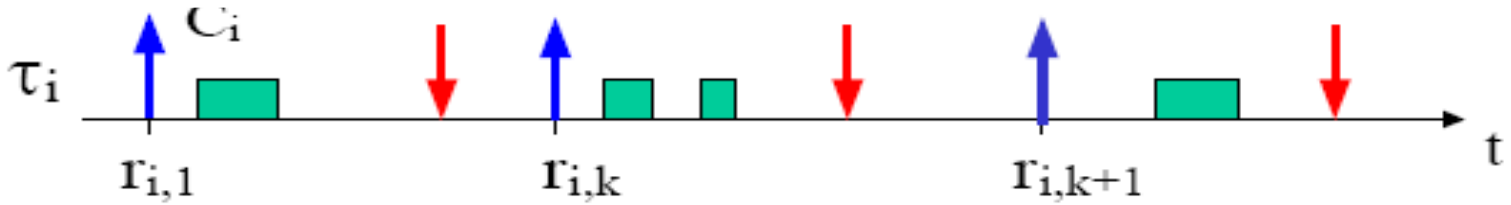


Task aperiodici

- Triggerati da interrupt esterni
- I task sporadici sono triggerati da interrupt esterni con un minimo tempo di interarrivo tra gli interrupt

Task aperiodici: $r_{i,k+1} > r_{i,k}$

Task sporadici: $r_{i,k+1} \geq r_{i,k} + T_i$



Metriche

- Lateness: L=f-d
- Exceeding time: E=max(0,L) → tempo in cui un processo e' rimasto attivo oltre la propria deadline
- Slack time (o LAXITY): LX=d-a-C → ritardo di attivazione max consentita
- Metriche di valutazione: basate sulla funzione di costo che dipende dal tempo di terminazione del task. La funzione di costo rappresenta l'importanza relativa del task.

Scheduling Real Time per Processi Aperiodici

- Ottimizzare una funzione di costo definita sui parametri temporali

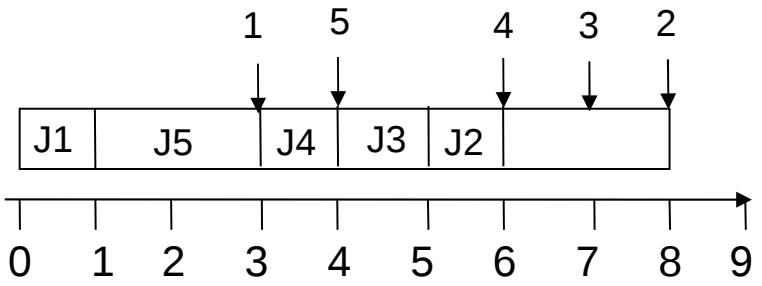
- Notazione di Graham: $(\alpha|\beta|\gamma)$
  dove:
  - α : macchina fisica (monoprocessore, multiprocessore etc)
  - β : tipo di vincoli ai processi (precedenza, preemption etc.)
  - γ : funzione di costo minimizzata

- Esempio:
  $(1|prec|L_{MAX})$, $(3|nopreempt.|\Sigma f_i)$, $(2| |\Sigma f_i)$

Algoritmo di Jackson

- Algoritmo $(1|a_0|L_{max})$ per un sistema di n tasks
- Consideriamo un insieme di task $J=\{J_i(a_i, C_i, d_i), i=1\ldots n\}$, dove $a_i=a_0$ per ogni $i=1\ldots n$
- Algoritmo: la massima lateness L_{max} e' minimizzata se i processi sono schedulati in ordine di deadline crescenti
- La complessita' di calcolo dipende principalmente dalla procedura di ordinamento dell'insieme di task → O(nlogn)

	J_1	J_2	J_3	J_4	J_5
C_i	1	1	1	1	2
d_i	3	8	7	6	4



$L_{max} = -1$

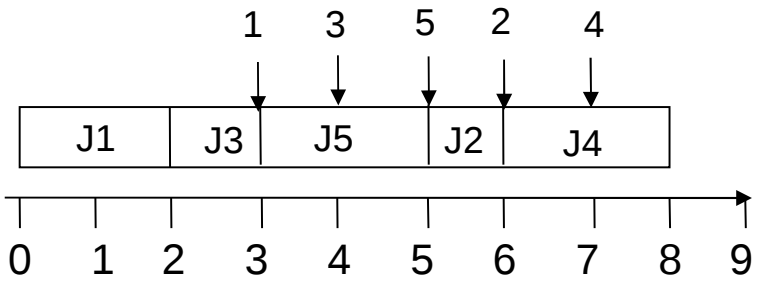
Algoritmo di Jackson

- Test di schedulabilità:

$$\forall\ i=1..n;\ \sum_{k=1}^{i} C_k \leq d_i$$

- Esempio di schedulazione Non Fattibile

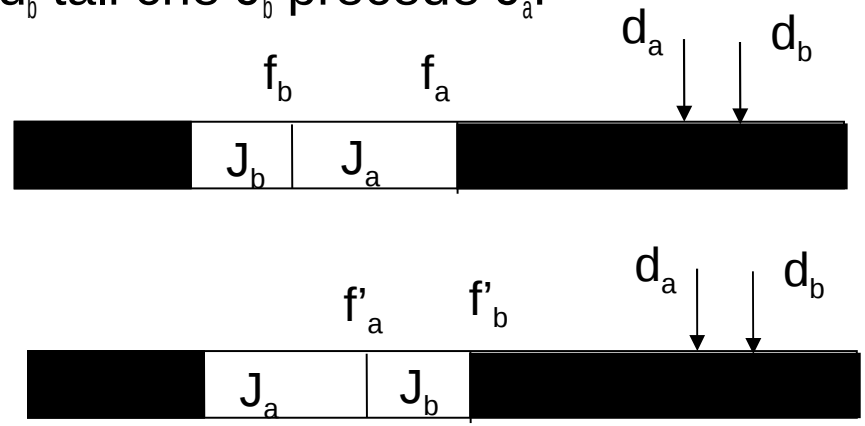
	J_1	J_2	J_3	J_4	J_5
C_i	2	1	1	2	2
d_i	3	6	4	7	5



$L_{max} = 1$

Algoritmo di Jackson

- Ottimalità dell'algoritmo di Jackson
- Per una schedulazione generica, esisteranno almeno due task J_a e J_b con $d_a \leq d_b$ tali che J_b precede J_a:



$L_a = f_a - d_a$
$L_b = f_b - d_b$
$L_{max} = f_a - d_a$

$L'_a = f'_a - d_a$
$L'_b = f'_b - d_b$

- Se si invertono i due task, la lateness massima diminuisce:

Se $(L'_a > L'_b)$ $L'_{max} = f'_a - d_a < f_a - d_a$ → $L'_{max} < L_{max}$

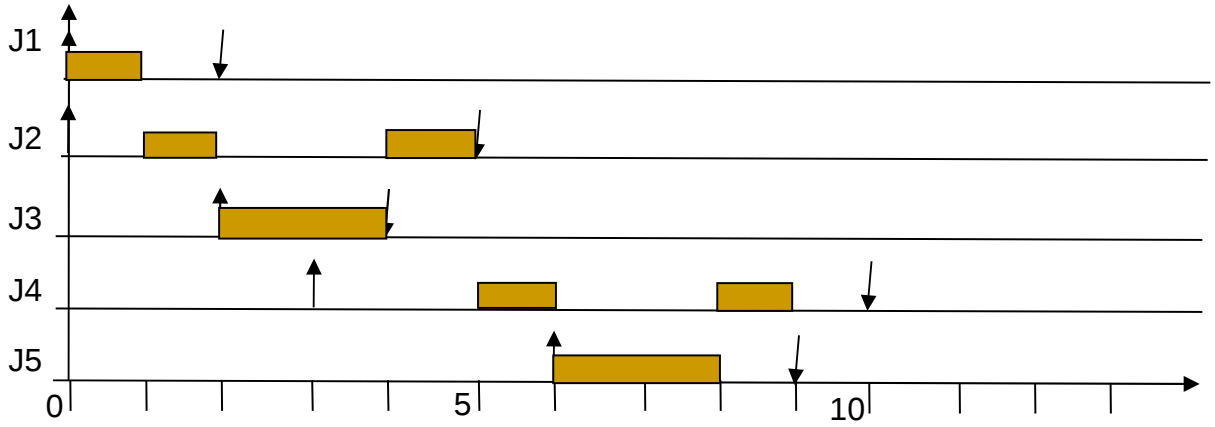
Se $(L'_b > L'_a)$ $L'_{max} = f'_b - d_b = f_a - d_b < f_a - d_a$ → $L'_{max} < L_{max}$

Eseguendo un numero finito di scambi di questo tipo si ottiene la schedulazione ottima

Algoritmo di Horn

- Algoritmo (1|preemp|L_{max})
- Rimuove l'ipotesi di attivazioni simultanee: attivazione dinamica e pre-emption
- Estensione dell'algoritmo di Jackson
- Algoritmo: La massima lateness L_{max} di un insieme di n task con attivazione dinamica e' minimizzata se, ogni volta che un nuovo task entra nel sistema la coda dei processi pronti viene riordinata per deadline crescente e la CPU viene assegnata al processo con deadline piu' imminente.
- Chiamata anche Earliest Deadline First (EDF)
- Ottimalita' nel senso che minimizza L_{max} e nel senso della schedulazione.

	a_i	C_i	d_i
J1	0	1	2
J2	0	2	5
J3	2	2	4
J4	3	2	10
J5	6	2	9



Algoritmo di Horn

- Complessita' $O(n^2)$, dove n è il numero di processi che possono essere attivati dinamicamente.
- Test di garanzia di schedulabilità: derivato dal test di Jackson:

$$\forall\ i=1..n;\ \sum_{k=1}^{i} c_k(t) \le \bar{d}_i$$

dove $c_k(t)$ sono i tempi residui istantanei di esecuzione e $\bar{d}_i$ sono le deadline riscalate rispetto ai tempi di arrivo.

- Minimizzazione di Lmax: deriva da Jackson
- Teorema:

Se un insieme di task aperiodici non è schedulabile con l'algoritmo di Horn, allora non è schedulabile con nessun altro algoritmo

Dim.

In altre parole, l'enunciato del teorema afferma che:

se un insieme di task è schedulabile con un qualche algoritmo A, allora sicuramente è schedulabile con l'algoritmo di Horn.

Algoritmo di Horn (cont.)

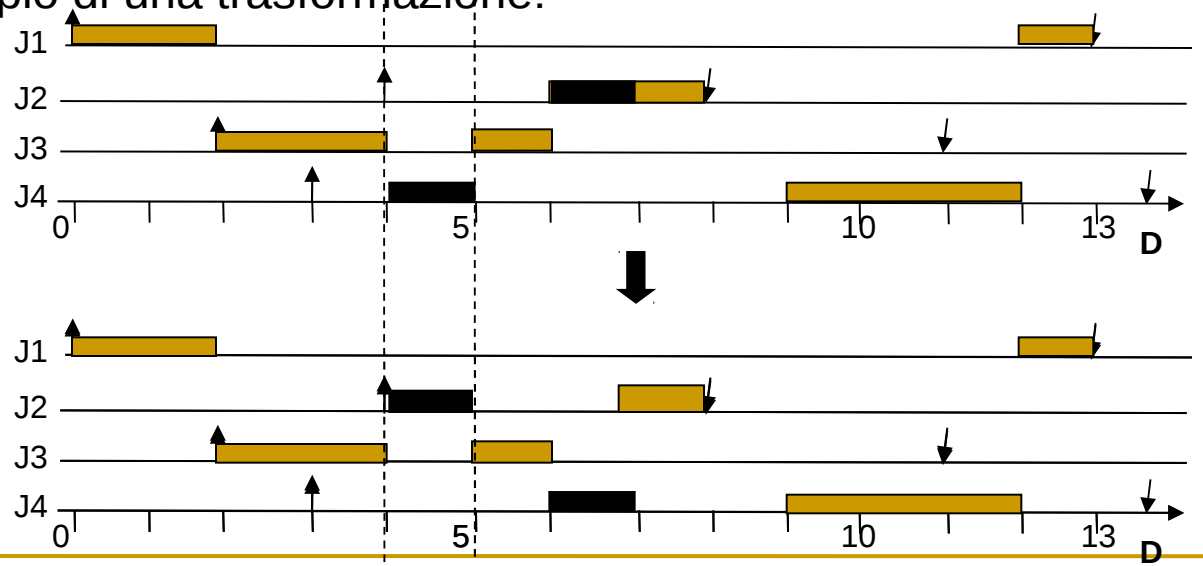
- Si divida la scala temporale in quanti pari all'unità di tempo del sistema
- Sia t=0 il primo istante di attivazione dei processi
- Sia D=max(d_i) la deadline più lontana
- Sia σ_A una qualsiasi schedulazione fattibile
- Sia σ (t) il task in esecuzione al tempo t nella schedulazione corrente
- Sia E(t) il task con deadline più imminente
- Sia t_E l'istante di tempo in cui inizia E(t) nella schedulazione corrente

Allora: la schedulazione può essere trasformata in una schedulazione di Horn con il seguente algoritmo:

```
Trasforma(){
   σ = σ_A;
   for (t=0; t<D; t++)
    if(σ (t) ≠ E(t)){
          σ(t_E) = σ(t);
          σ(t) = E(t);
    }
}
```

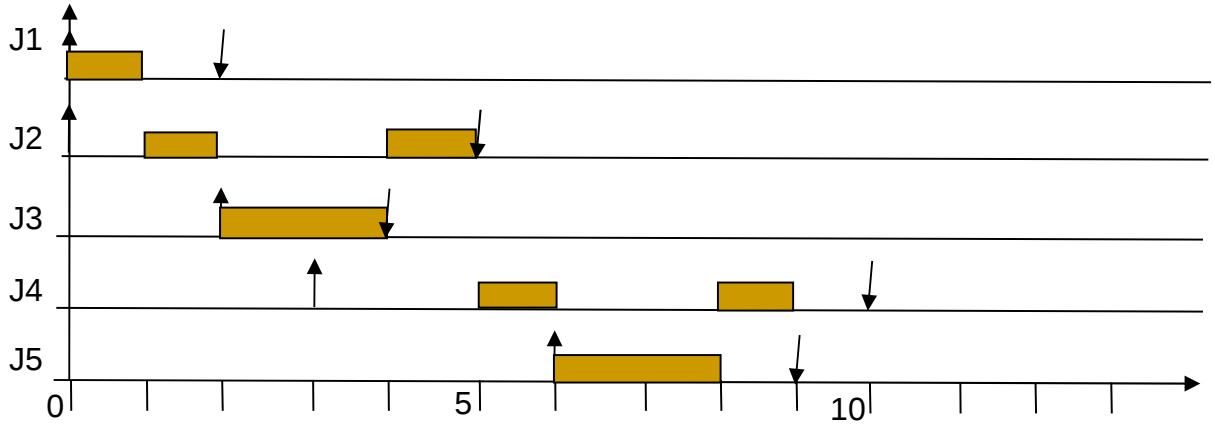
Algoritmo di Horn (cont.)

- Ciascuna trasformazione preserva il tempo di calcolo dei task (i quanti possono essere solo traslati, non accorciati o allungati)
- Tutti i tempi possono al più essere ritardati di t_E
- Se la schedulazione σ_A è fattibile, allora prima della trasformazione $(t_E+1) \leq d_E$, ma $d_E \leq d_i$ per ogni i, quindi dopo la trasformazione $(t_E+1) \leq d_i$ quindi tutti i task terminano entro le deadline → Horn è fattibile
- Esempio di una trasformazione:



Algoritmo di Horn (cont.)

	a_i	C_i	d_i
J1	0	1	2
J2	0	2	5
J3	2	2	4
J4	3	2	10
J5	6	2	9



- Analisi della schedulabilità: deve essere fatta ad ogni arrivo

  ➔ le deadline devono essere riscalate ad ogni arrivo del tempo dell'arrivo

  - Istante 0: sono presenti in coda J1 e J2 (nell'ordine). Tempo residuo per J1: 1; per J2: 2.

  1 <= d1= 2 1+2 <= d2=5

  - Istante 2: sono presenti in coda J3 e J2 (nell'ordine). Tempo residuo per J3: 2; per J2: 1

  2 <= d3=2 2+1 <= d2= 3

  - Istante 3: sono presenti in coda J3, J2, J4 (nell'ordine). Tempo residuo per J3: 1; per J1: 1; per J4: 2

  1 <= d3=1 1+1 <= d1= 2 1+1+2 <= d4=7

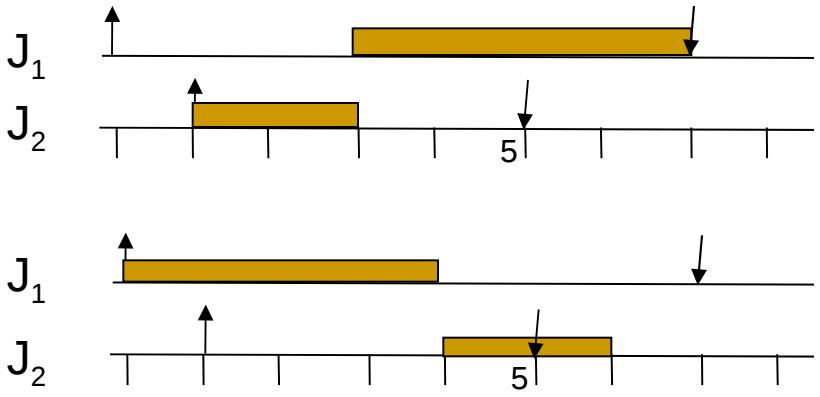
  - Istante 6: sono presenti in coda J5, J4 (nell'ordine). Tempo residuo per J5: 2; per J4: 1

  2 <= d5=3 2+1 <= d4=4

Schedulazione non pre-emptive. Algoritmo di Horn

- Se si esclude l'ipotesi di preemption, con attivazione dinamica l'algoritmo EDF non e' piu' ottimo
- Esempio:

	a_i	C_i	D_i
J_1	0	4	7
J_2	1	2	5



Schedulazione ottima

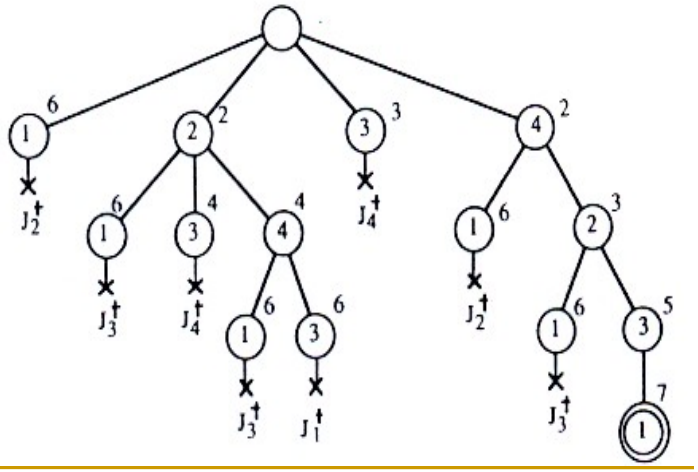
Schedulazione EDF

Schedulazione non pre-emptive. Algoritmo di Bratley

- Schedulazione senza pre-emption di un insieme di task attivati dinamicamente
- Ricerca su un albero con pruning
- Complessita' O(nn!)
- Algoritmo off-line. Esempio:

	a_i	C_i	d_i
J_1	4	2	7
J_2	1	1	5
J_3	1	2	6
J_4	0	2	4

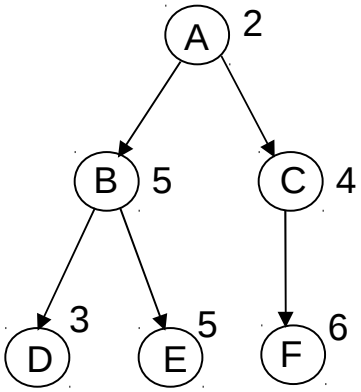
Numero nel nodo
→ task che viene schedulato
Numero accanto al nodo
→ tempo in cui il task termina
J^+ → task che supera la deadline
→ schedulazione fattibile

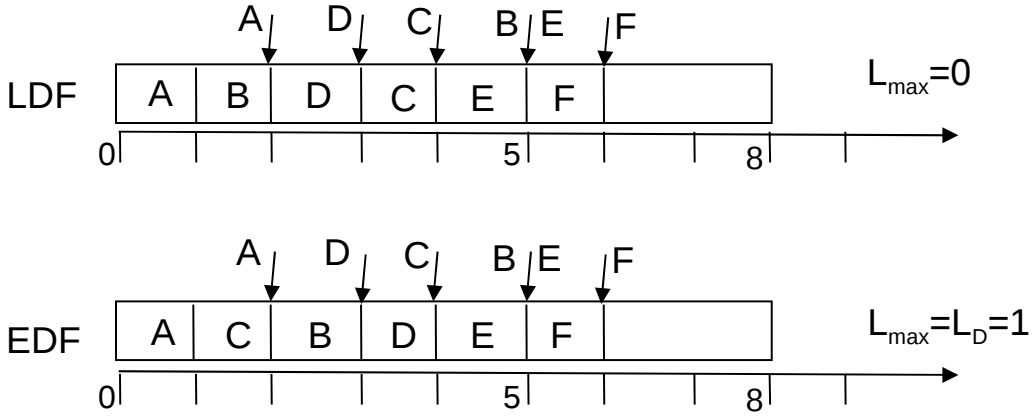


Algoritmi di Scheduling con Vincoli di Precedenza

- Puo' essere risolta con algoritmi polinomiali solo se si impongono ipotesi semplificative
- Algoritmo Latest Deadline First (LDF). Algoritmo ($1|prec, a_0| L_{max}$)
- Algoritmo: Dato un insieme J di n task con grafo di precedenza, si costruisce la lista di scheduling a partire dal fondo. Fra tutti i task che non hanno successori nel grafo, si seleziona il processo con la deadline piu' lunga.
- Schedulato l'ultimo task, la lista viene eseguita in ordine inverso.
- Esempio:

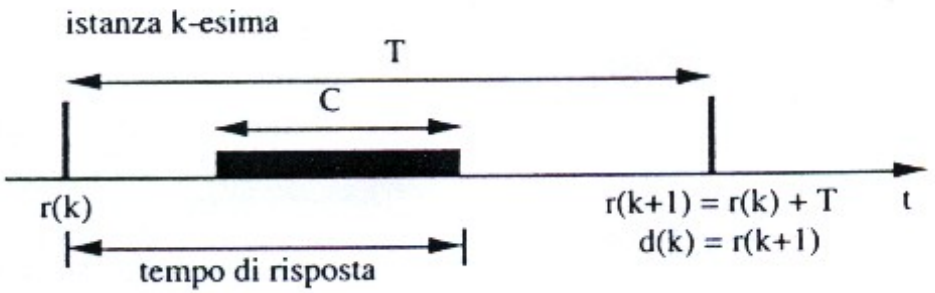
	C_i	d_{ii}
J_A	1	2
J_B	1	5
J_C	1	4
J_D	1	3
J_E	1	5
J_F	1	6





Task periodici

- Sono la maggioranza delle attivita' di elaborazione. Es. regolazione, acquisizione, filtraggio, monitoraggio, comando di attuatori etc.
- Ipotesi:
  - Tutte le richieste di esecuzione sono inoltrate ad intervalli regolari (periodo)
  - Il tempo di eseuzione di un task e' costante
  - La deadline coincide con la fine del periodo corrente
  - Tutti i task sono indipendenti
- Quindi, un processo periodico e' caratterizzato da due parametri:
  - Periodo T_i
  - Tempo di esecuzione C



Task periodici

- Fattore di utilizzazione del processore U
  - E' la frazione di tempo utilizzata dalla CPU per eseguire l'insieme di task (e' una misura della occupazione del tempo di CPU per eseguire un insieme di task periodici)
  - In un insieme di n task: $U = \sum_{1}^{n} \frac{C_i}{T_i}$
  - Il processore e' "completamente utilizzato" dall'insieme di task se un piccolo aumento di un C_i rende la schedulazione non fattibile
  - "Limite superiore minimo" U_{lsm} del fattore di utilizzazione: minimo tra i fattori di utilizzazione calcolati su tutti gli insiemi di task che utilizzano completamente il processore. Parametro caratteristico di scheduling. E' il carico massimo gestibile da un algoritmo di schedulazione.

Task periodici
Earliest Deadline First (EDF)

- Si seleziona dalla lista dei processi pronti quello la cui deadline e' piu' imminente
- Pre-emptive: se arriva un task con deadline minore → sospensione
- Utilizzabile nei SO a base prioritaria (priorita' alta=deadline vicina)
- La coda dei processi pronti ordinata (velocizzare)
- *Teorema di schedulabilità: Condizione necessaria e sufficiente per la schedulabilità e'* $\Sigma C_i/T_i \leq 1 \rightarrow U_{lsm}=1$
- Esempio:

	C_i	T_i
τ_1	1	5
τ_2	2	7
τ_3	4	9

U=1/5+2/7+4/9=0.93