Praktikum iz operativnih sistema

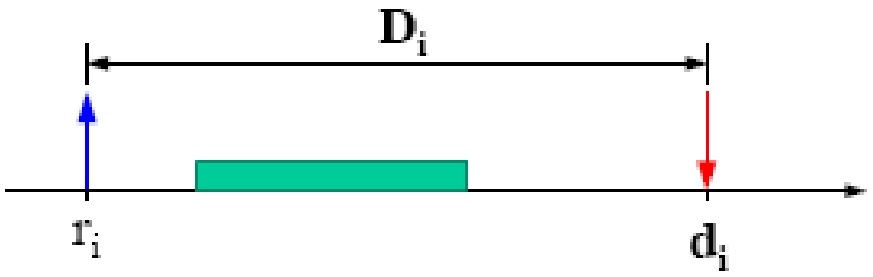
Lekcija 4: Raspoređivanje aperiodičnih poslova

zima 2019/2020

Prof. dr Branimir Trenkić

Real time(RT) algoritmi

- Poslovi mogu biti ***raspoređeni na osnovu***
- ***Relativnog*** deadline-a $\mathbf{D_i}$ (***statički***)
- ***Apsolutnog*** deadline-a $\mathbf{d_i}$ (***dinamički***)



Algoritam najranijeg dospeća

- ***Earliest Due Date*, EDD** (**1 | sync | L_{max}**)
- ***Pravilo***:
- Svakom poslu u skupu su dodeljeni ***vremenski zahtevi (C_i, D_i)***:
  - Vreme izvršavanja posla (wcet) – ***C_i***
  - Relativni *deadline* - ***D_i***

1. Svi poslovi nailaze simultano (**istovremeno**) (t = 0)
2. ***Primena*** algoritma – ***off line***
3. Serivisira se posao sa ***najkraćim relativnim deadline-om***

Algoritam najranijeg dospeća

- ***E**arliest **D**ue **D**ate*, **EDD** (**1 | sync | L_{max}**)
- Skup Γ možemo predastaviti na sledeći način:

$$\Gamma = \{J_i(C_i, D_i), i = 1, \ldots, n\}$$

- Dakle, ***prioriteti su fiksni*** (D_i je poznat unapred)
- ***Prekidanje nije uslov*** (s obzirom da se svi aktiviraju u istom trenutku)
- ***Ciljna mera:- Maksimalno prekoračenje*** – **L_{max}**
- ***Optimalan u tom smislu - Minimalizuje maksimalno prekoračenje*** – **L_{max}**

Algoritam najranijeg dospeća

- Teorema (***Jackson***-ovo pravilo):

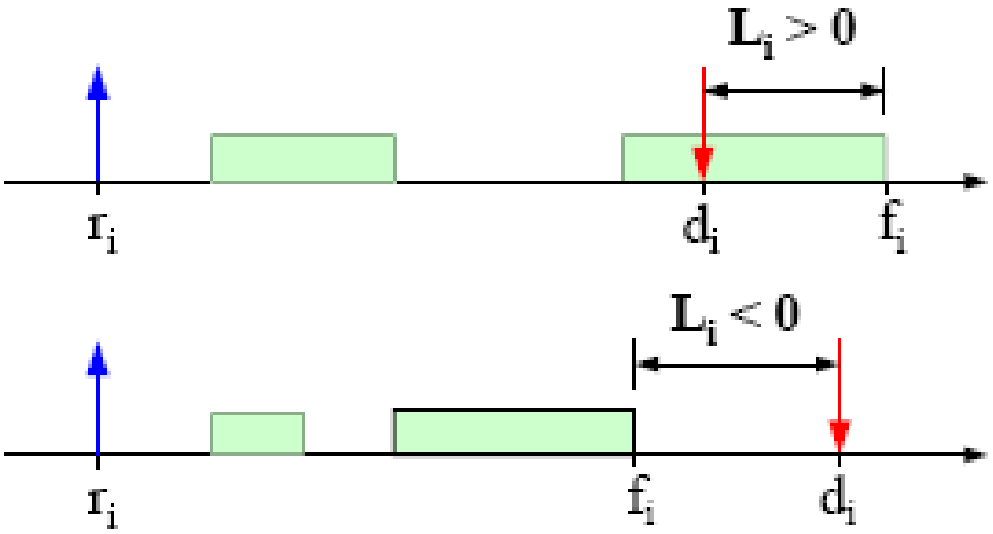
  Za dati skup od ***n nezavisnih poslova***, svaki algoritam kojim se izvršavaju poslovi po ***redosledu ne-opadajućih deadline-ova***

  je ***optimalan*** u odnosu na

  minimiziranje maksimalnog prekoračenja (L_{max})

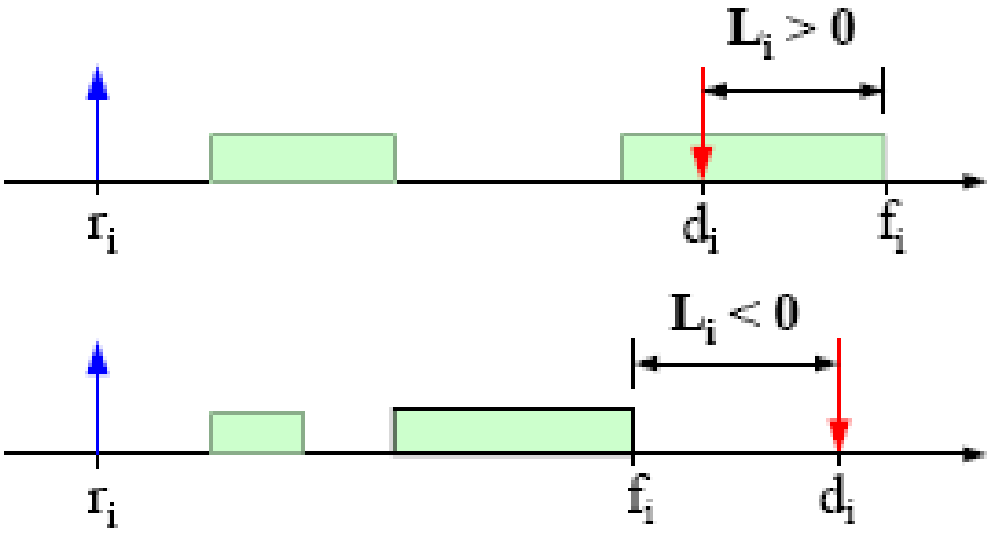
Prekoračenje (kašnjenje)

$$L_i = f_i - d_i$$



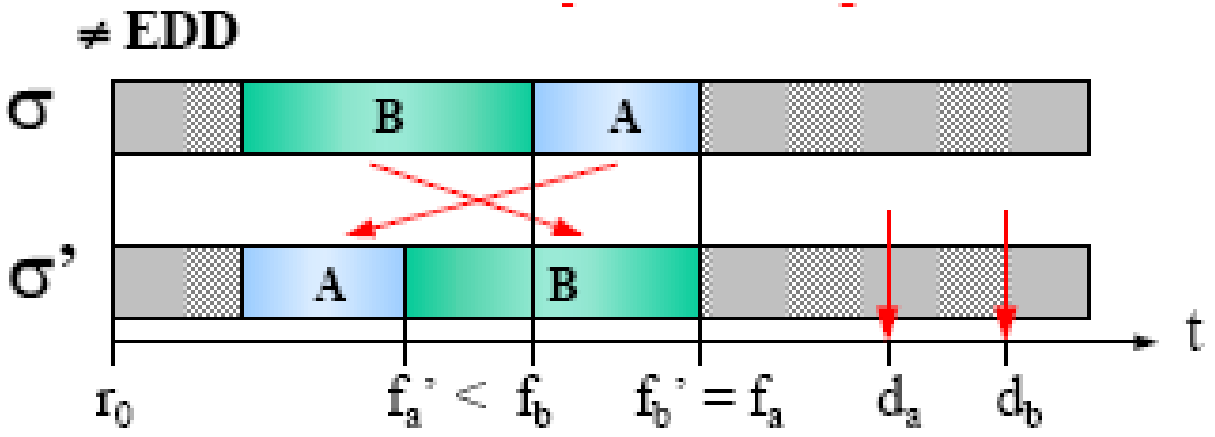
Maksimalno prekoračenje

$$L_{max} = max_i(L_i)$$



if ($L_{max} < 0$) **then**
ni jedan posao nije prekoračio svoj deadline

Dokaz optimalnosti EDD-a



$$L_{\max} = L_a = f_a - d_a$$

$$L'_{\max}(A,B) = \begin{cases} L'_a = f'_a - d_a < f_a - d_a \\ L'_b = f'_b - d_b < f_a - d_a \end{cases} \Rightarrow L'_{\max} < L_{\max}$$

Optimalnost EDD-a

$$\sigma \to \sigma' \to \sigma'' \to \cdots \to \sigma^*$$

$$L_{\max}(\sigma) \geq L_{\max}(\sigma') \geq \cdots \geq L_{\max}(\sigma^*)$$

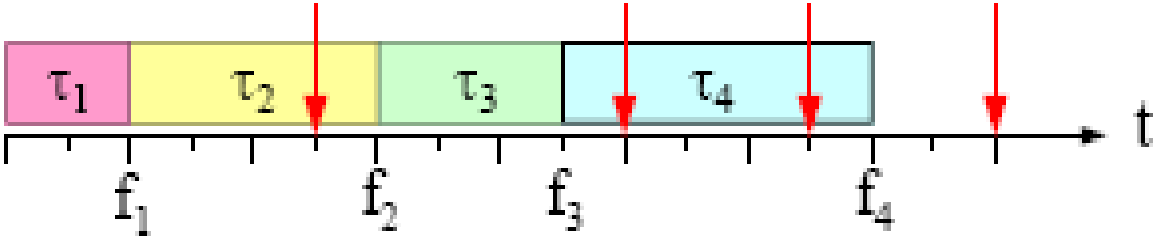
$$\sigma^* = \sigma_{EDD}$$

$$L_{\max}(\sigma_{EDD}) -$$

Minimalna vrednost koja može biti dostignuta ***u odnosu na bilo koji algoritam***

Test garancije EDD-a (off-line)

- Test rasporedljivosti ili ***test garancije izvodljivosti*** (***off line***)



- ***Raspored*** nad skup poslova Γ je **izvodljiv** ***ako i samo ako*** za svako **i**, važi da je $\mathbf{f_i \leq d_i}$

$$f_i = \sum_{k=1}^{i} C_k$$

$$\forall i \quad \sum_{k=1}^{i} C_k \leq D_i$$

Algoritam najranijeg *deadline*-a

- ***Earliest Deadline First, EDF*** (***1 | preem | L_{max}***)

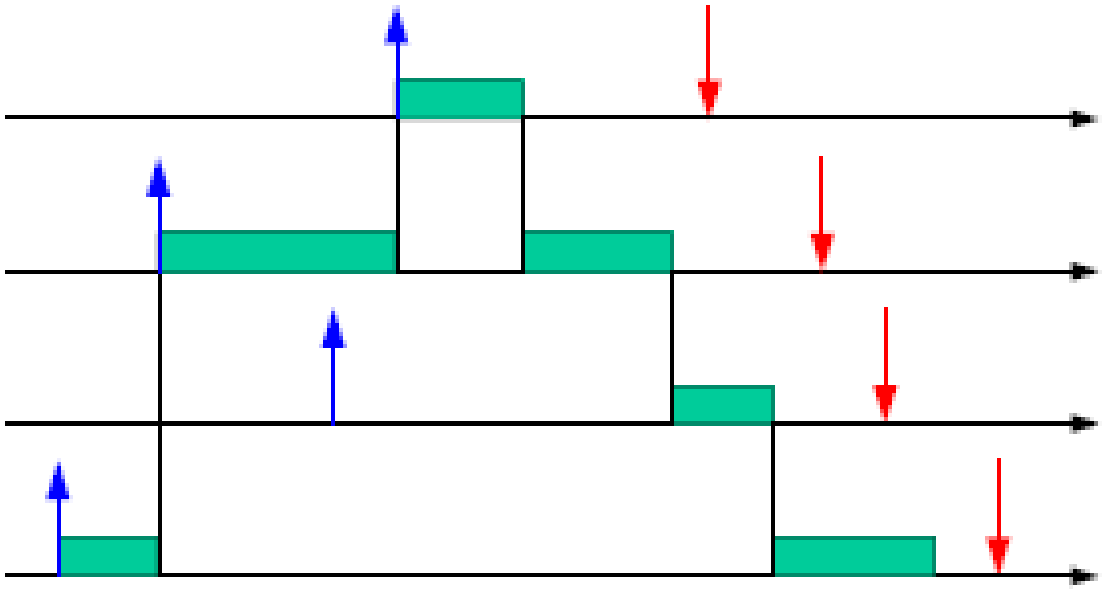
1. Nailazak poslova ***nije sinhron*** nego slučajan
   - Poslovi mogu naići u bilo kom trenutku
2. Ovaj algoritam ***selektuje posao*** sa ***najranijim apsolutnim deadline-om*** [Horn 74]
3. Omogućeno je potpuno ***prekidanje poslova***
4. ***Primena*** algoritma – ***on line***

- ***Dinamički prioritet*** ($\mathbf{d_i}$ zavisi od trenutka nailaska)

Algoritam najranijeg deadline-a

- ***Teorema (Horn)***:
- Naka je dat skup od **n** nezavisnih poslova sa ***slučajnim*** vremenima nailaska (*arrival*), algoritam koji ***u bilo kom trenutku*** izvršava posao sa ***najranijim apsolutnim*** deadline-om, među svim spremnim poslovima, je ***optimalan*** u smislu ***minimiziranja maksimalnog prekoračenja***

EDF - Primer



EDF optimalnost

- Algoritam ***EDF je optimalan*** *u smislu* ***izvodljivosti***
- Pored toga, može se pokazati da EDF, ***takođe***, ***minimizira maksimalno prekoračenje*** - L_{max}

EDF optimalnost

- Algoritam A ***je optimalan u smislu izvodljivosti***
- To znači:

  ako uopšte ***postoji izvodljiv raspored*** za skup poslova Γ, tada je ***algoritam A*** u stanju da ga ***nađe***

Način dokazivanja optimalnosti (Demonstracioni metod):

- Dovoljno je pokazati da je za dati ***proizvoljni izvodljivi raspored***, raspored generisan sa EDF je takođe izvodljiv

Osobine optimalnih algoritama

- ***Obrnuta tvrdnja***:
- Ako optimalan algoritam (u smislu izvodljivosti) proizvodi neizvodljiv raspored, onda ***ne postoji*** algoritam koji može proizvesti izvodljiv raspored!

- Ako algoritam **A** ***minimizira*** L_{max} tada je **A** takođe ***optimalan u smislu izvodljivosti***

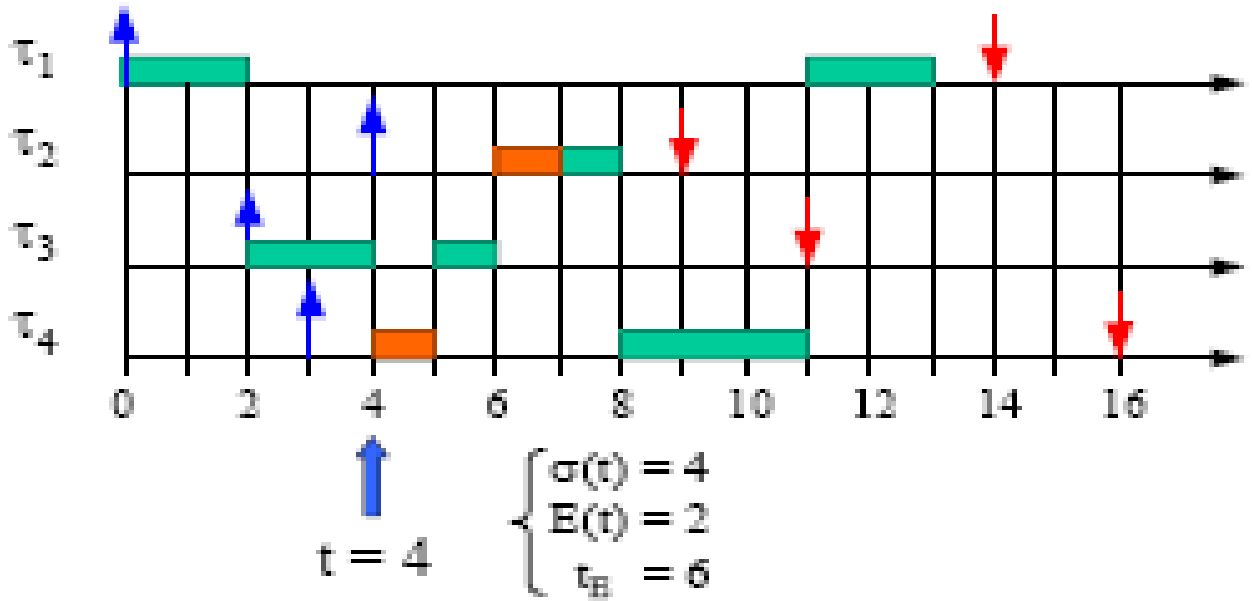
  Obrnuto ne važi!

EDF optimalnost

$$\begin{cases} \sigma(t) = \text{posao u intervalu } [t, t+1) \\ E(t) = \text{posao sa min } d \text{ u trenutku } t \\ t_E = \text{vreme izvrsenja posla } E \end{cases}$$

$$\sigma \neq \sigma_{EDF} \Rightarrow (\exists t)\,\sigma(t) \neq E(t)$$

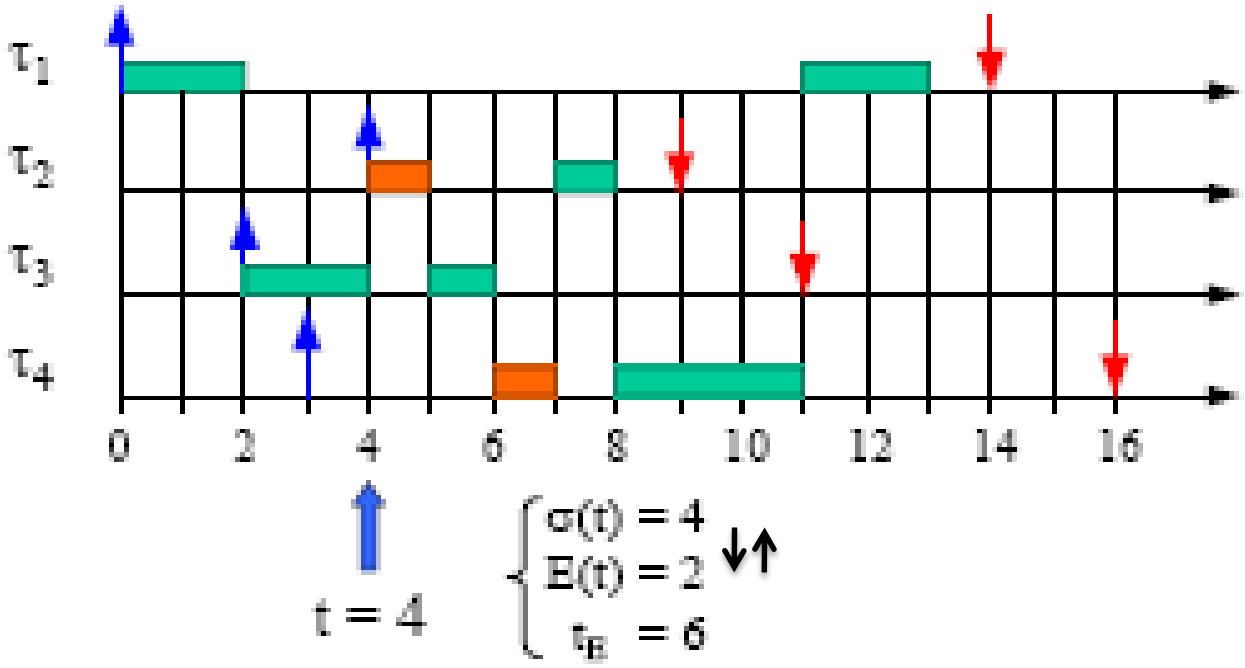
```
for (t = 0  to  Dmax−1)
    if (σ(t) ≠ E(t)) {
        σ(tE) = σ(t);
        σ(t) = E(t);
    }
```



EDF optimalnost

$$\begin{cases} \sigma(t) = \text{posao u intervalu } [t, t+1) \\ E(t) = \text{posao sa min } d \text{ u trenutku } t \\ t_E = \text{vreme izvrsenja posla } E \end{cases}$$

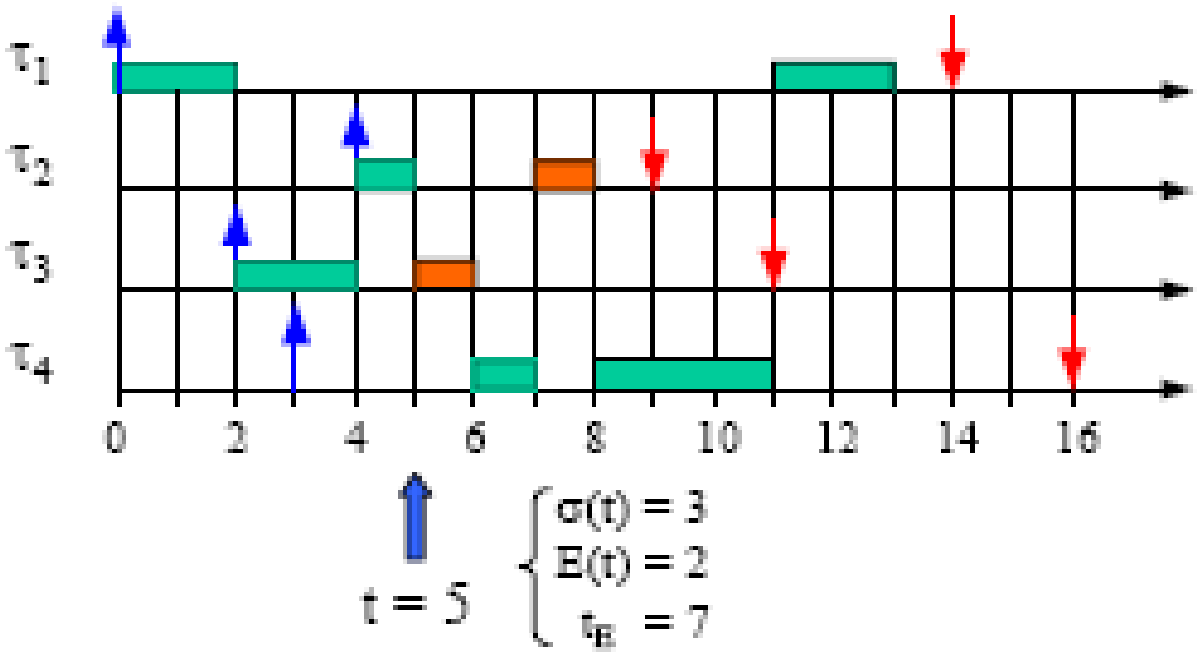
```
for (t = 0  to  Dmax−1)
    if (σ(t) ≠ E(t)) {
        σ(tE) = σ(t);
        σ(t) = E(t);
    }
```



EDF optimalnost

$$\begin{cases} \sigma(t) = \text{posao u intervalu } [t, t+1) \\ E(t) = \text{posao sa min } d \text{ u trenutku } t \\ t_E = \text{vreme izvrsenja posla } E \end{cases}$$

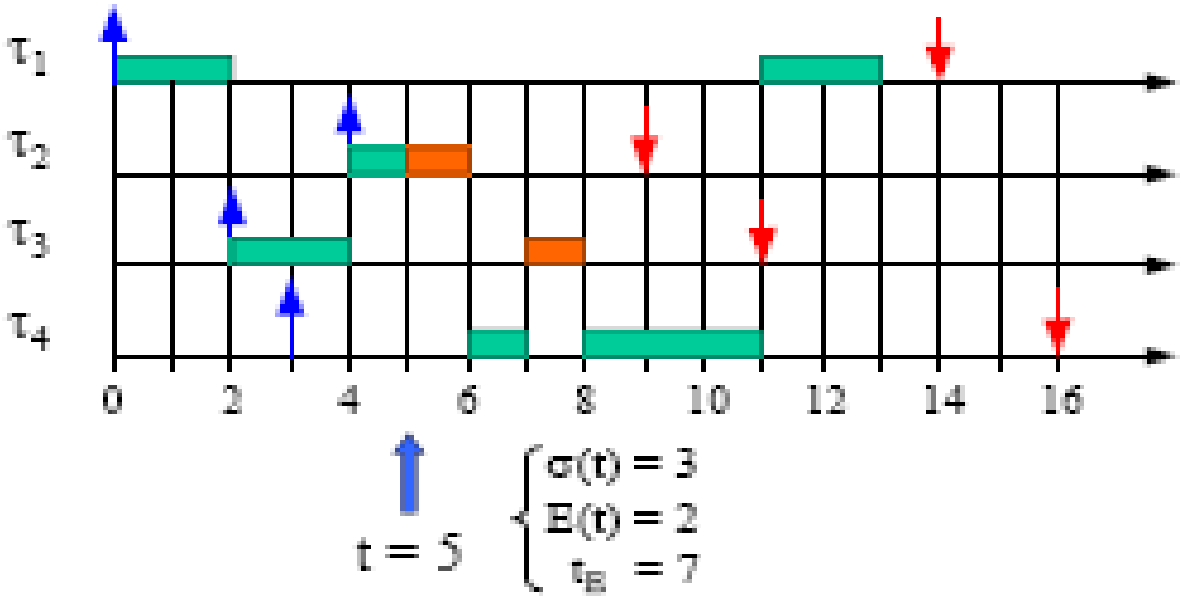
```
for (t = 0  to  D_max−1)
    if (σ(t) ≠ E(t)) {
        σ(t_E) = σ(t);
        σ(t) = E(t);
    }
```



EDF optimalnost

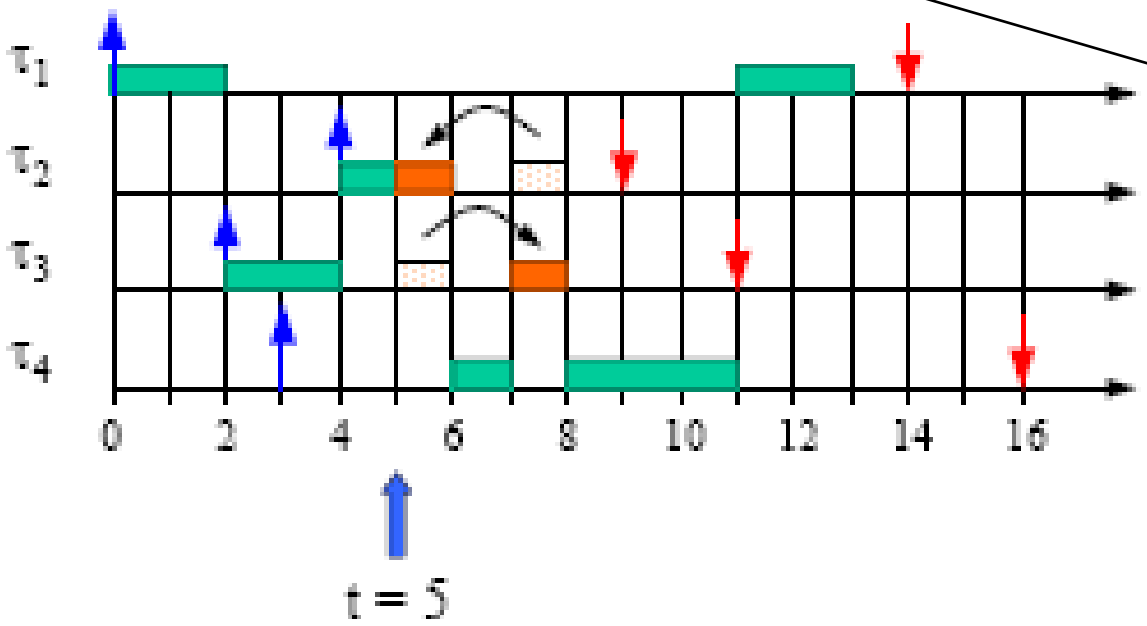
$$\begin{cases} \sigma(t) = \text{posao u intervalu } [t, t+1) \\ E(t) = \text{posao sa min } d \text{ u trenutku } t \\ t_E = \text{vreme izvrsenja posla } E \end{cases}$$

```
for (t = 0   to  Dmax−1)
      if (σ(t) ≠ E(t)) {
            σ(tE) = σ(t);
            σ(t) = E(t);
      }
```

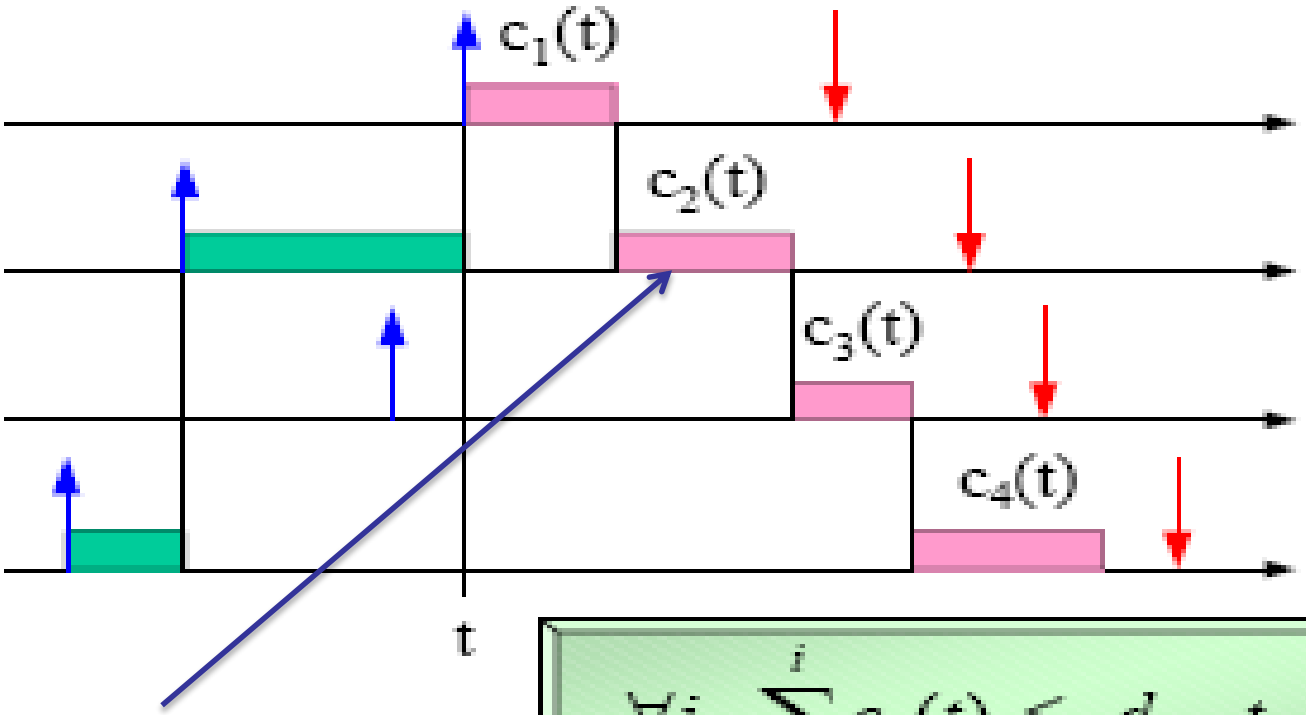


EDF optimalnost

- Izloženi algoritam premeštanja će ***sačuvati rasporedljivost*** skupa poslova:
  - To je ***očigledno*** za ***slic*** koji se ***pomera unapred***
  - Za ***slic*** koji se ***pomera unazad***, rasporedljivost slica je, takođe, očuvana! $[7,8) \le d_2 \wedge d_2 \le d_3 \Rightarrow [7,8) \le d_3$



Test garancije EDF-a (on line)



Rezidualno vreme izvršavanja

$$\forall i \quad \sum_{k=1}^{i} c_k(t) \leq d_i - t$$

Složenost algoritama

- ***EDD***
- ***O(n logn)*** uređenja skupa poslova
- ***O(n)*** garancija čitavog skupa poslova

- ***EDF***
- ***O(n)*** ubaciti novi posao u red spremnih (***lista***)
- ***O(logn)*** ubaciti novi posao u red spremnih (***heap***)
- ***O(n)*** garancije za novi posao

Raspoređivanje bez prekidanja

- Kada

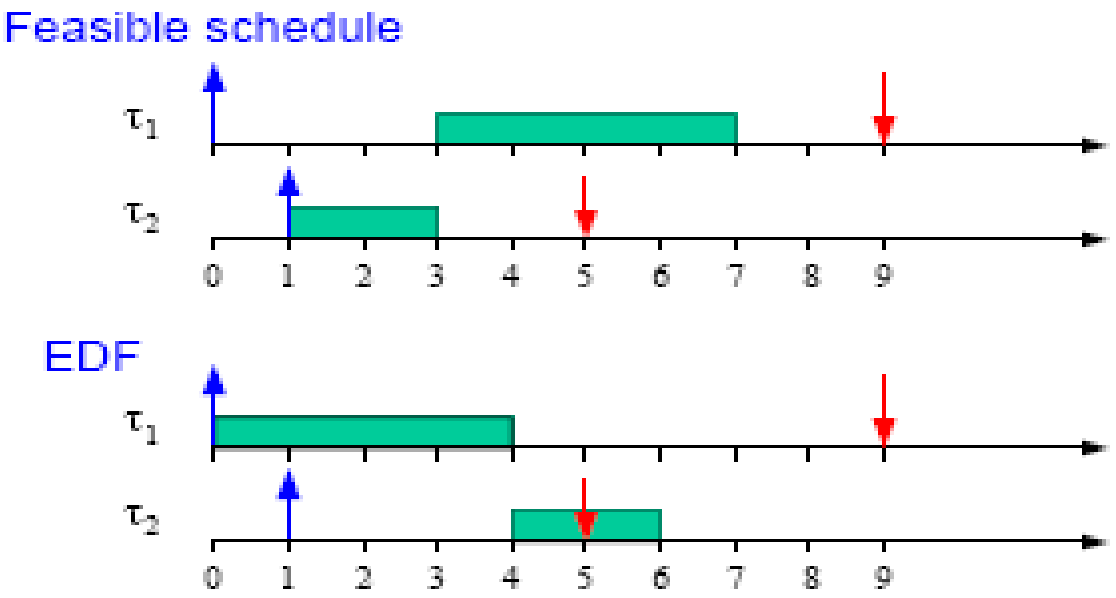
(i) ***Prekidanje*** izvršenja ***nije dozvoljeno***

(ii) Poslovi se ***aktiviraju u slučajnom trenutku***

- Problem ***minimiziranja maksimalnog prekoračenja*** – **NP- hard**
- Problem ***nalaženja izvodljivog rasporeda*** – **NP- hard**

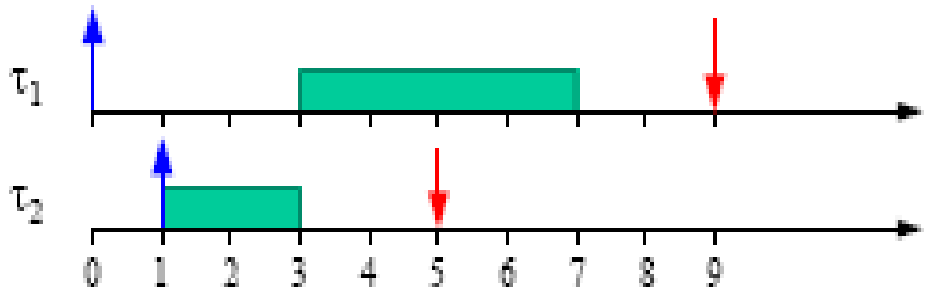
Raspoređivanje bez prekidanja

- ***EDF*** pod tim uslovima ***nije više optimalan***:



Raspoređivanje bez prekidanja

- Da bi se dostigla optimalnost, ***algoritam mora biti "vidovit"***, i odluči da procesor ostane slobodan (idle) i ako postoji spreman posao za izvršenje:



Algoritm raspoređivanja koji ***ne dozvoljava*** procesoru da bude u idle- stanju ako postoji aktivan posao – ***non-idle algoritam***

Raspoređivanje bez prekidanja

- Ako se ***ograničimo na non- idle algoritme*** raspoređivanja – **EDF je još uvek optimalan** algoritam iako predpostavimo neprekidajući model posla

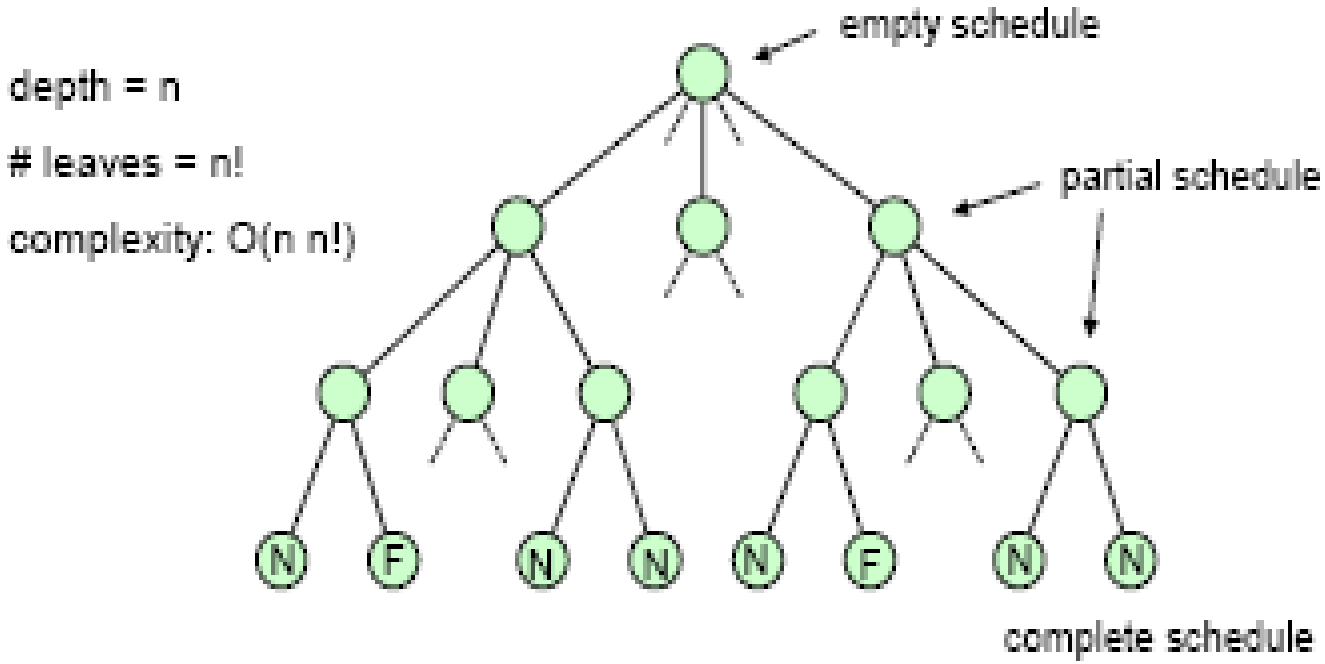
- Razmatramo ***probleme raspoređivanja***:

(i) vremena aktiviranja (a_i) su ***poznata unapred***

(ii) ***ne-prekidajući model*** posla se predpostavlja

Ne-prekidajući algoritmi

- **Problem** nalaženja izvodljivog rasporeda (***off line***) se često povezuje sa ***algoritmom pretraživanja stabla***
- ***Eksponencijalna složenost*** u najgorem slučaju



Ne-prekidajući algoritmi

- Složenost problema se ***može redukovati ograničavanjem prostora pretraživanja*** a time redukujemo složenost proračuna u algoritmu
  - ***Korišćenjem dodatnih informacija*** o poslovima koji su nam na raspolaganju
  - ***Korišćenje heurističkog pristupa*** u nalaženju verovatnog puta kroz stablo
- Algoritmi u ***polinomialnom vremenu***
- ***Heuristički algoritmi*** mogu naći izvodljiv raspored u polinomijalnom vremenu ali nema garancija da će ga naći

Bratley-ev algoritam

(*1* | *no-preem.* | *feasible*)

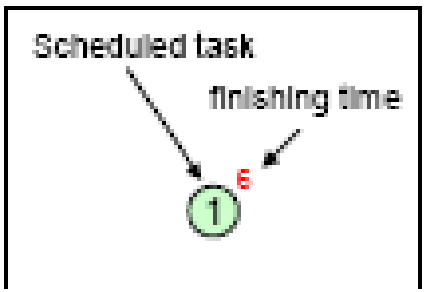
- Redukuje prosečnu složenost (prostor pretraživanja) tehnikom ***orezivanja***:
- Stablo se ***širi samo u slučaju*** kada je ***parcijalni raspored stogo izvodljiv***
- Za ***parcijalni raspored*** se kaže da je ***strogo izvodljiv*** ako ***dodavanjem*** bilo kog od preostalih poslova on i dalje ***ostaje izvodljiv***

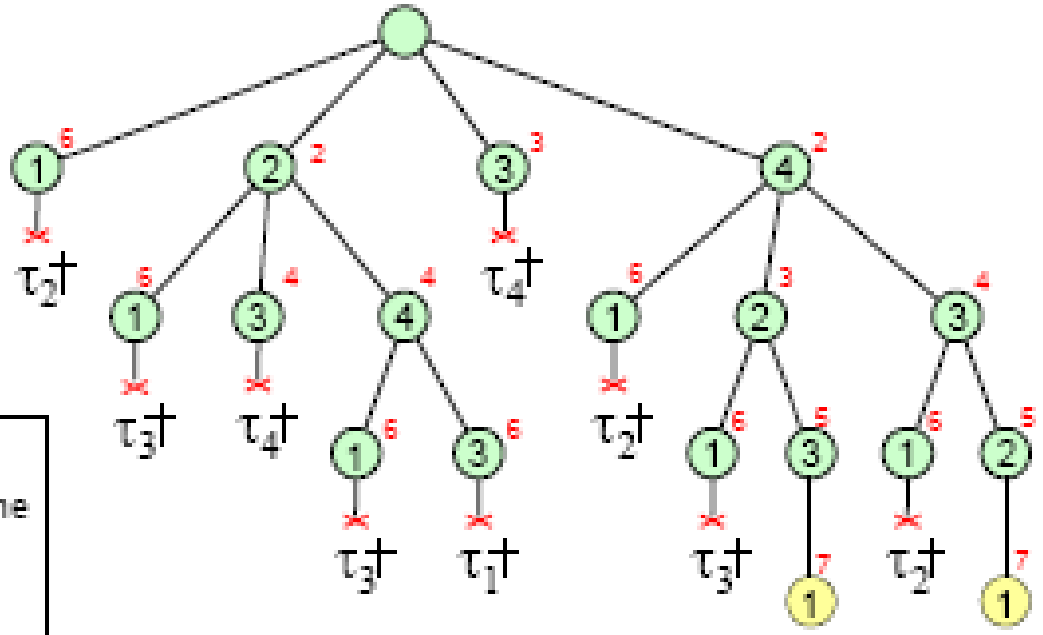
Bratley-ev algoritam

- Redukuje prosečnu složenost ***tehnikom orezivanja***:

Example

	a_i	C_i	d_i
τ_1	4	2	7
τ_2	1	1	5
τ_3	1	2	6
τ_4	0	2	4





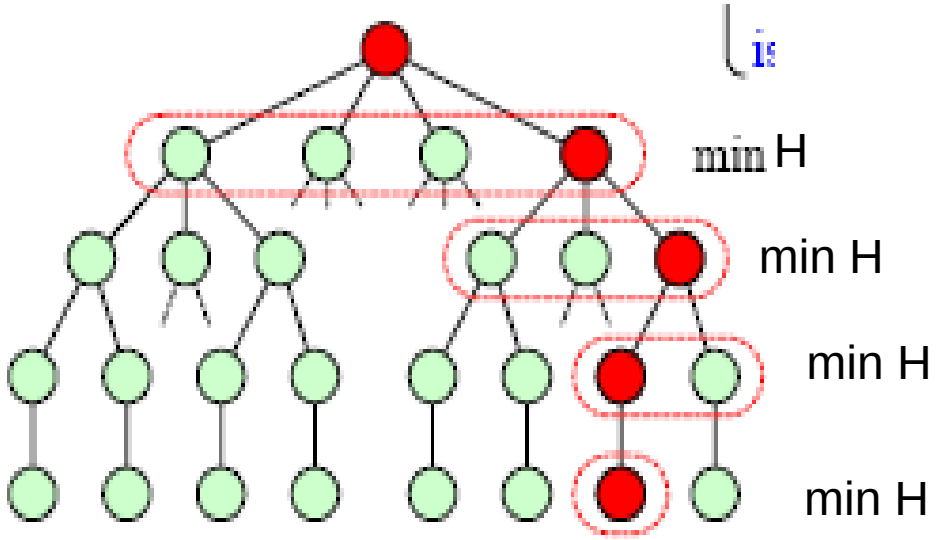
$\sigma_1 = \{4, 2, 3, 1\}$

$\sigma_2 = \{4, 3, 2, 1\}$

Napomena: Ovi rasporedi generalno ***omogućuju idle stanje***!

Primer heurističkog algoritma

- ***Spring algoritam*** (Stankovic & Ramamritham)
- ***Raspored za n poslova je formiran u n koraka***
- ***U svakom koraku*** algoritam ***selektuje posao*** koji ***minimizira heurističku funkciju***



- Moguće je pretraživanje i unazad

Spring algoritam

- ***Mehanizam zasnovan na heurističkoj funkciji (H)*** je veoma ***fleksibilan mehanizam***
- Omogućuje ***jednostavno definisanje i modifikaciju*** politike raspoređivanja

- Primeri ***prostih*** heurističkih funkcija:
  - $H = r_i \rightarrow FCFS$
  - $H = C_i \rightarrow SJF$
  - $H = d_i \rightarrow EDF$

Spring algoritam

- ***Uslovljenost resursima*** **u Spring-u**
- Svaki posao τ_i određuje binarni niz resursa:

$$R_i = [R_1(i), \ldots , R_r(i)],$$

$R_k(i) = 1$ - resurs R_k se ekskluzivno ***koristi*** pri izvršenju posla τ_i

$R_k(i) = 0$ - resurs R_k ***nije potreban*** pri izvršenju posla

- ***U svakom koraku*** (parcijalni raspored), za svaki resurs R_k ***izračunava se***

EAT_k – ***najranije vreme raspoloživosti*** ***resursa*** R_k

Spring algoritam

- Znači, ***najranije startno vreme***, u kom posao τ_i može ***početi*** svoje ***izvršavanje***, dobijamo kao:

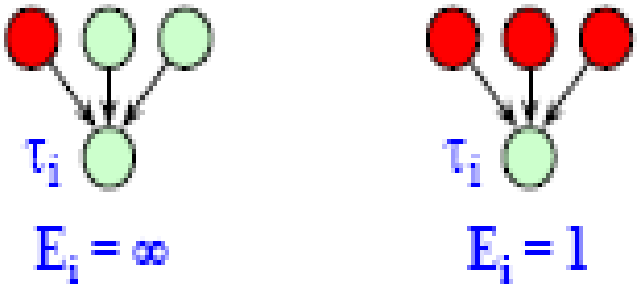
$$T_{est}(i) = max[a_i, max_k(EAT_k)]$$

- Kada se izračuna T_{est} za sve preostale poslove – ***moguća strategija*** - ***selektovanje posla sa najmanjom vrednošću*** T_{est}

- Primer ***složenih*** heurističkih funkcija:
  - $H = d_i + wC_i \rightarrow$ ***EDF + SJF***
  - $H = d_i + wT_{est} \rightarrow$ ***EDF + ESTF***

Spring algoritam

- **Mogućnost rukovanja uslovima prvenstva**
- Dodajemo ***novi parametar*** heurističkoj funkciji
- Parametar : ***podobnost*** (***kvalifikovanost***), $\mathbf{E_i}$



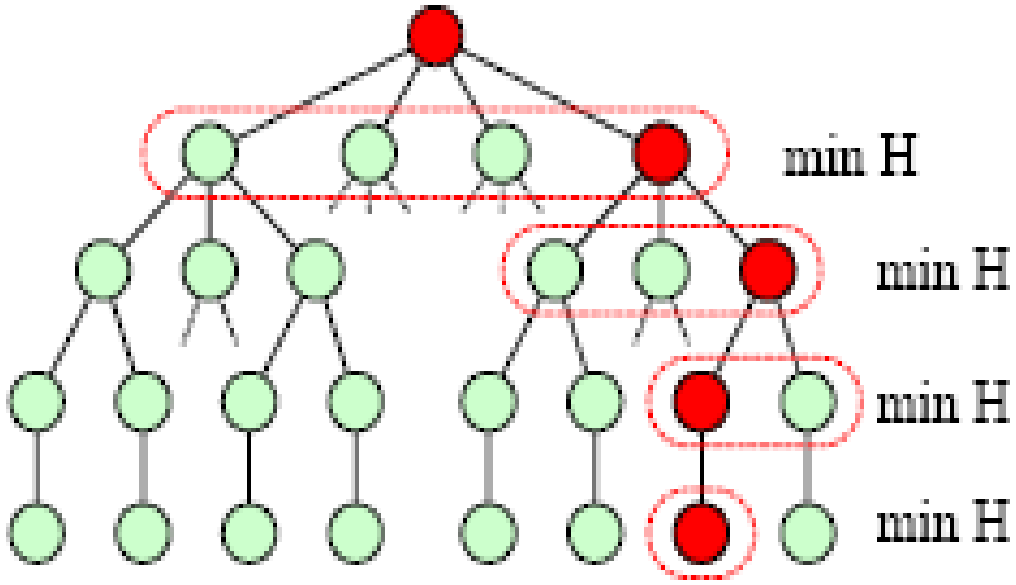
- Heurističke funkcije:

$$H = E_i(w_1 r_i + w_2 D_i)$$

$$H = E_i(w_1 C_i + w_2 d_i)$$

Spring algoritam

- ***Složenost***:
- Potpuno pretraživanje: $O(n \cdot n!)$
- Heurističko pretraživanje: $O(n^2)$
- k - Heurističko pretraživanje: $O(kn)$



Uslov prvenstva u rasporedu

- ***Problem nalaženja optimalnog rasporeda*** za skup poslova sa ***relacijama prvenstva***,
- Rešenje ovog problema se odlikuje ***NP-hard složenošću***
- Optimalni algoritmi, koji rešavaju dati problem u ***polinomijalnom vremenu***, mogu se naći uz neke ***predpostavke***:

  A) Sinhrona aktivacija

  Ako je dozvoljena ***dinamička aktivacija*** poslova

  B) Prekidajuće respoređivanje

Uslov prvenstva u rasporedu

- ***Latest Deadline Fitst, LDF*** (***1 | prec, sync | L_{max}***)
- ***Predpostavke algoritma:***

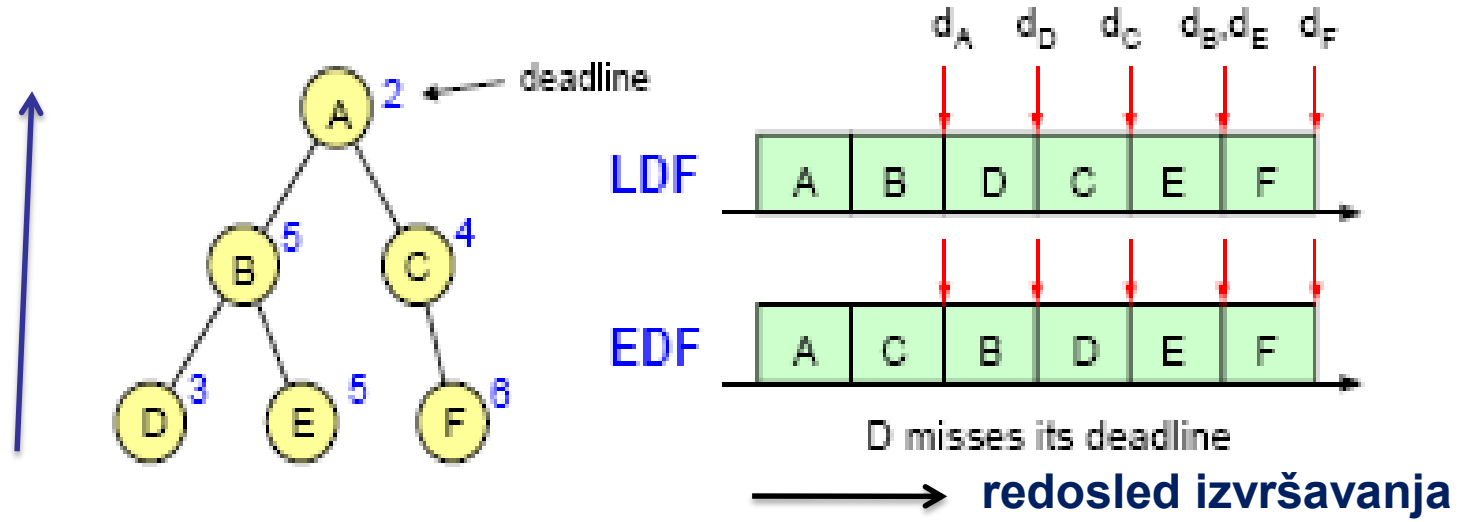
1. ***Ne-prekidajući*** algoritam
2. Dat je ***graf prvenstva*** (***DAG –*** usmereni aciklični)
3. ***Sinhronizacija u nailasku*** poslova za izvršenje

Postupak:

- Gradi se ***red*** (queue) ***raspoređivanja*** **–** ***koji se popunjava od repa ka glavi*** (***from tail to head***)

Uslov prvenstva u rasporedu

- ***Pravilo***:
- Među tačkama ***bez naslednika***, ***LDF selektuje*** onaj ***posao sa najkasnijim deadline-om***
- Procedura se ponavlja dok ima poslova u skupu



Vreme izvršavanja svih poslova jednak je $C_i = 1$!

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- *A priori* se znaju:

1. vremena aktiviranja poslova i

2. njihovi ***deadline***-ovi

- ***Osnovna ideja algoritma***: - Transformisati skup Γ ***zavisnih*** poslova u skup Γ^* ***nezavisnih*** poslova ***modifikovanjem*** vremenskih parametara

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF*** [Chetto et all 89]

1. Transformacija uslova prvenstva u vremenske uslove, modifikujući

- ***vremena aktiviranja*** i
- ***deadline-ove***

na osnovu datog grafa prvenstva $(\Gamma \rightarrow \Gamma^*)$

2. Primeniti EDF na tako modifikovan skup poslova

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]

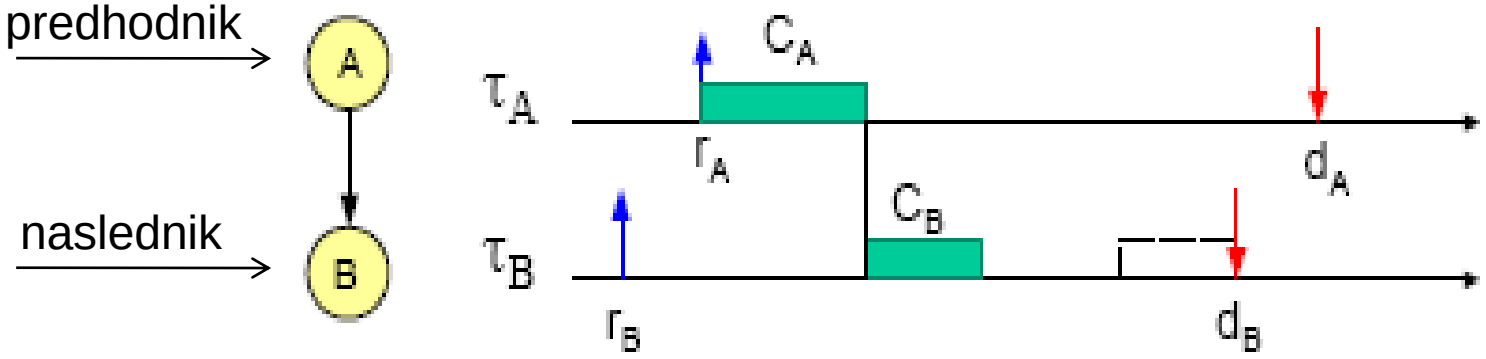
Teorema:

Izvorni skup Γ je ***rasporedljiv ako i samo ako*** je modifikovani skup Γ^* izvodljiv sa EDF algoritmom

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]



<u>Ideja je</u>:

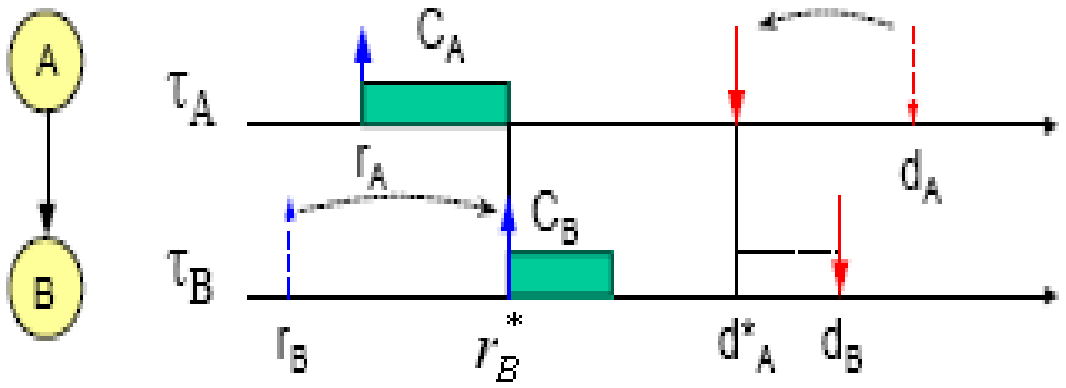
→ ***odložiti*** vreme nailaska ***naslednika***

→ ***pomeriti napred*** *deadline* ***predhodnika***

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]



Ideja je:

→ ***odložiti*** vreme nailaska naslednika $\mathbf{r^*_B = max(r_B, r_A+C_A)}$

→ **pomeriti napred** deadline predhodnika $\mathbf{d^*_A = min(d_A, d_B- C_B)}$

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Modifikacija vremena nailaska***:

Algoritam (***$O(n^2)$***):

(1) za sve početne tačke (korene u grafu), ***$r^*_i = r_i$***

(2) odrediti posao τ_i kome $\mathbf{r_i}$ nije modifikovano, a da kod ***svih*** njegovih neposrednih ***predhodnika*** (τ_k) jeste, inače **exit**

(3) $r_i^* = \max\left\{r_i, \max_{\tau_k \to \tau_i}\left(r_k^* + C_k\right)\right\}$

(4) ponovo od linije 2

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Deadline modifikacija***:

(1) za sve tačke (lišće), $d^*_i = d_i$

(2) odrediti posao τ_i kome $\mathbf{d_i}$ nije modifikovano, a da kod svih njegovih ***neposrednih naslednika*** (τ_k) jeste, inače **exit**

(3) $d_i^* = \min\left\{d_i, \min_{\tau_i \to \tau_k}\left(d_k^* - C_k\right)\right\}$

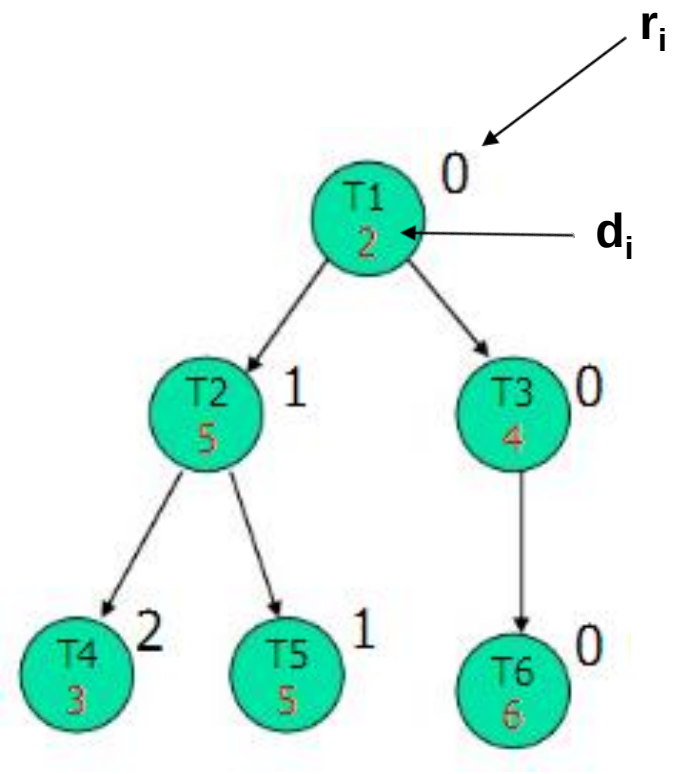
(4) ponovo od linije 2

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Primer***:

	T_1	T_2	T_3	T_4	T_5	T_6
C	1	1	1	1	1	1
d	2	5	4	3	5	6
r	0	1	0	2	1	0



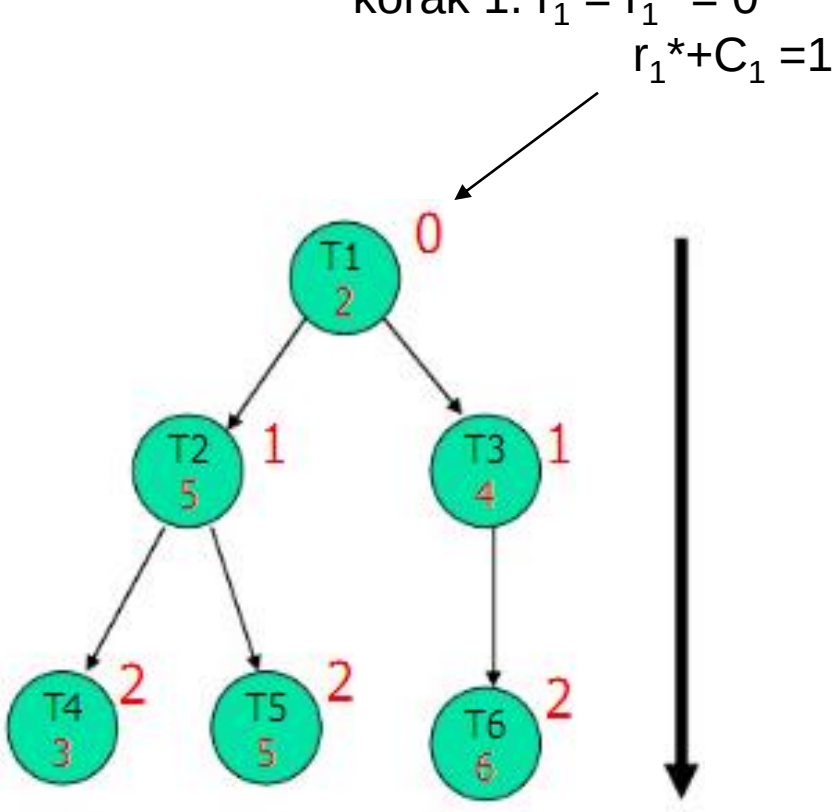
Da li je dati skup poslova rasporedljiv za date uslove?

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Primer***:
- ***Korak 1***. Modifikacija ***vremena nailaska***

	T_1	T_2	T_3	T_4	T_5	T_6
C	1	1	1	1	1	1
d	2	5	4	3	5	6
r	0	1	0	2	1	0



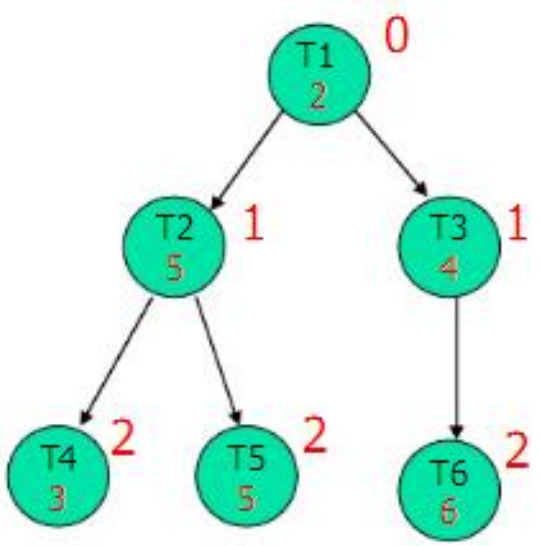
$$r_i^* = \max\left\{r_i, \max_{T_k \to T_i}\left(r_k^* + C_k\right)\right\}$$

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Primer***:
- ***Korak 1***. Modifikacija

  vremena nailaska



	T_1	T_2	T_3	T_4	T_5	T_6
C	1	1	1	1	1	1
d	2	5	4	3	5	6
r*	0	1	1	2	2	2

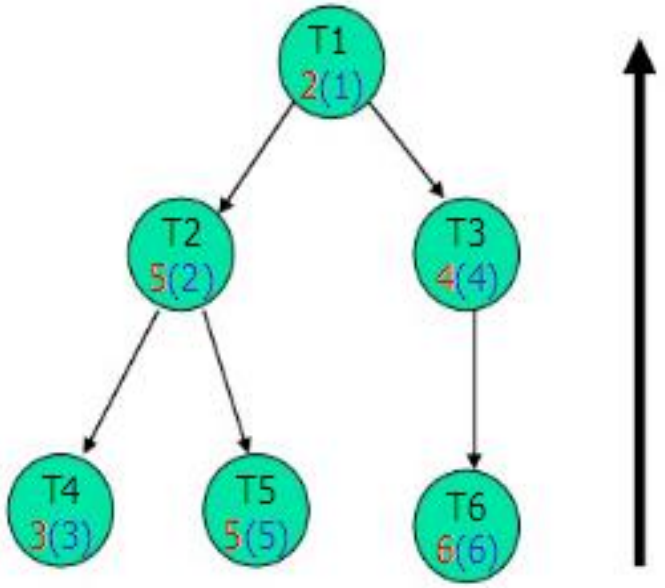
$$r_i^* = \max\left\{r_i, \max_{T_k \to T_i}\left(r_k^* + C_k\right)\right\}$$

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Primer***:
- ***Korak 2***. Modifikacija deadline-ova

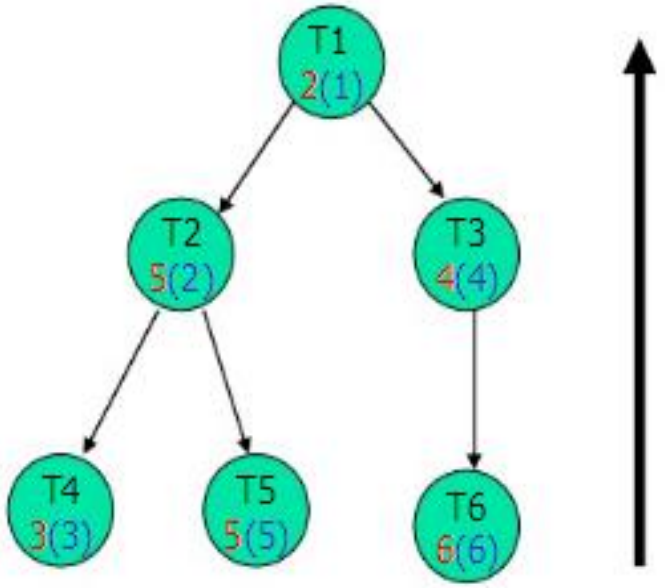
	T_1	T_2	T_3	T_4	T_5	T_6
C	1	1	1	1	1	1
d	2	5	4	3	5	6
r*	0	1	1	2	2	2



Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Primer***:
- ***Korak 2***. Modifikacija deadline-ova



	T_1	T_2	T_3	T_4	T_5	T_6
C	1	1	1	1	1	1
d*	1	2	4	3	5	6
r*	0	1	1	2	2	2

$$d_i^* = \min\left\{d_i, \min_{T_i \to T_k}\left(d_k^* - C_k\right)\right\}$$

Uslov prvenstva u rasporedu

1 | prec, preem | L_{max}

- ***EDF**** [Chetto et all 89]
- ***Primer***:
- ***Korak 3***. Primena EDF algoritma na skup poslova:

	T_1	T_2	T_3	T_4	T_5	T_6
C	1	1	1	1	1	1
d*	1	2	4	3	5	6
r*	0	1	1	2	2	2

Rezultujući raspored: $T_1, T_2, T_4, T_3, T_5, T_6$