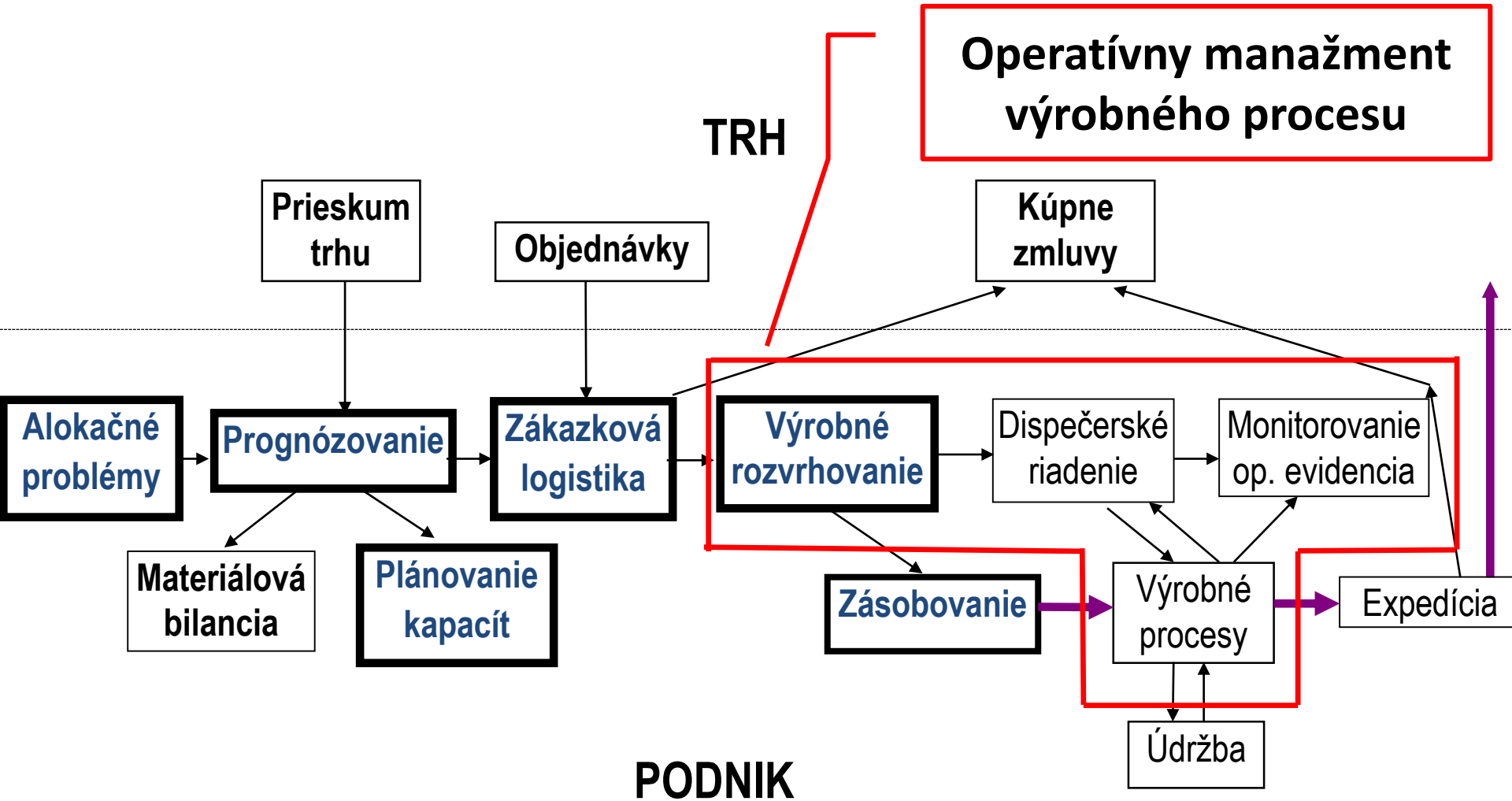
Rozvrhovanie

- Operatívny manažment výrobného procesu
- Plánovanie vs. rozvrhovanie
- Rozvrhovanie – hlavné a doplnkové charakteristiky, typy úloh
- Rozvrh, optimálny rozvrh, používané kriteriálne funkcie
- **Rozvrhovanie na paralelných strojoch/procesoroch**
  1. Rozvrhovanie na jednom stroji/procesore
  2. Rozvrhovanie na viacerých strojoch/procesoroch
- **Rozvrhovanie na dedikovaných (špecializovaných) strojoch**
  1. Open shop (riešiť ako flow shop)
  2. Flow shop
  3. Job shop

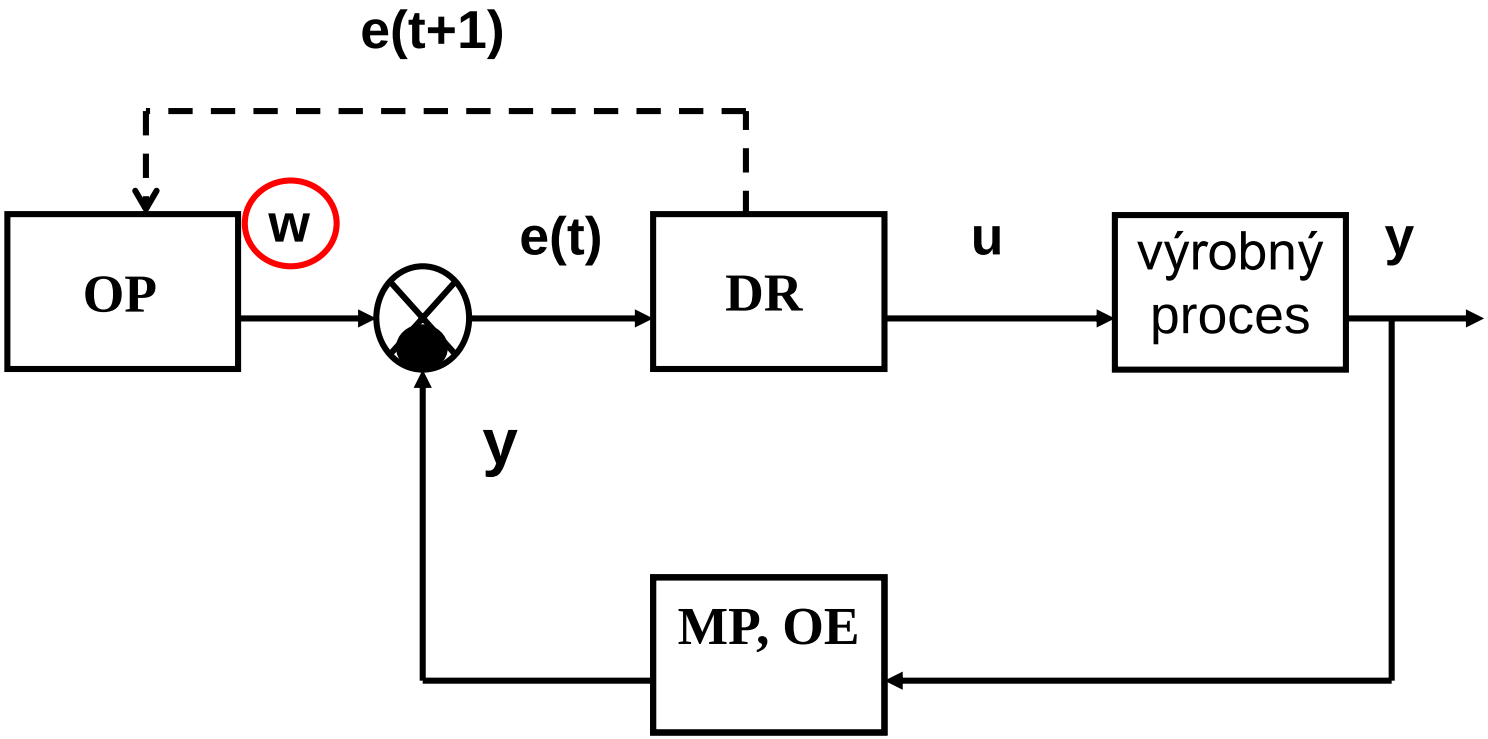
Operatívny manažment výrobného procesu

- **Definícia:** systém riadiacich činností, ktoré priamo zabezpečujú priebeh výrobného procesu.
- Pozostáva z:
  - Operatívneho plánovania
  - Dispečerského riadenia
  - Monitorovania procesov a operatívnej evidencie

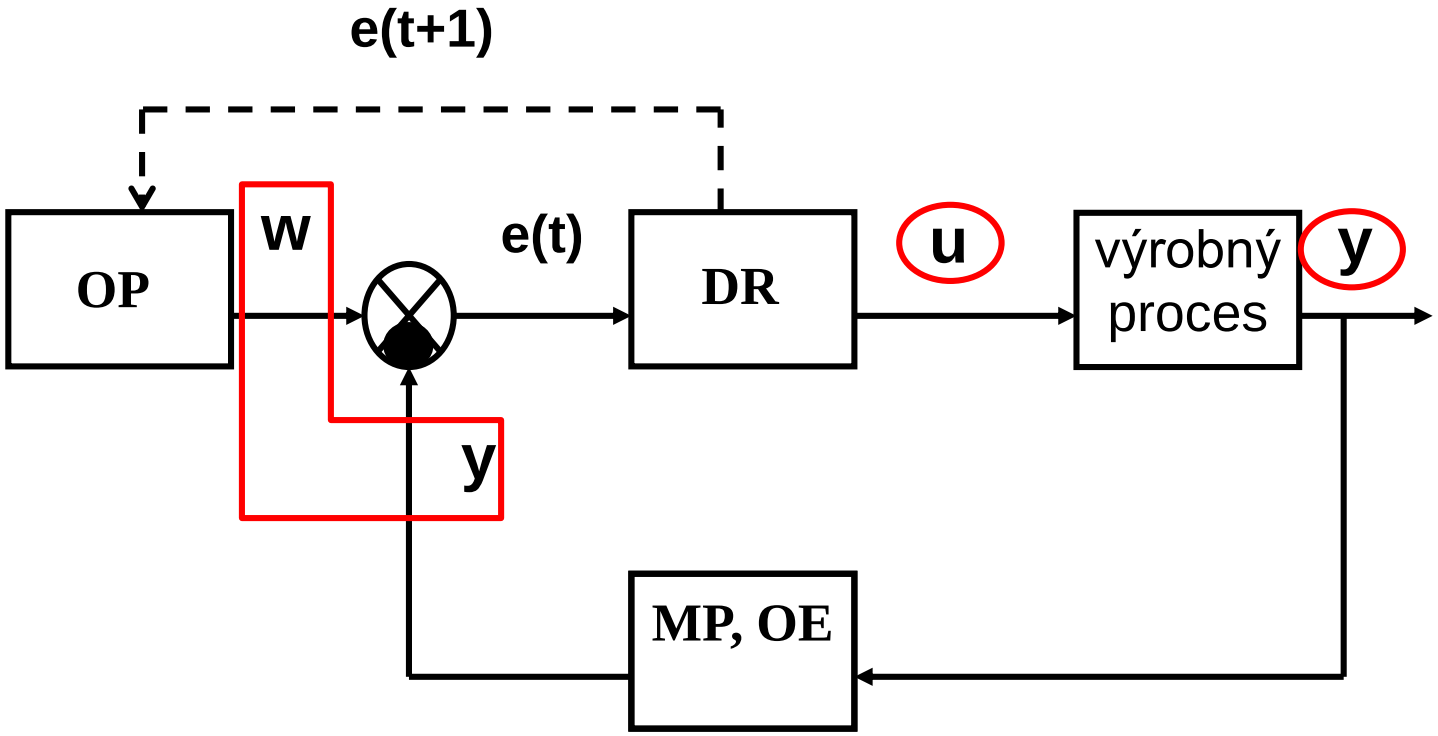
Štruktúra činností výrobnej logistiky



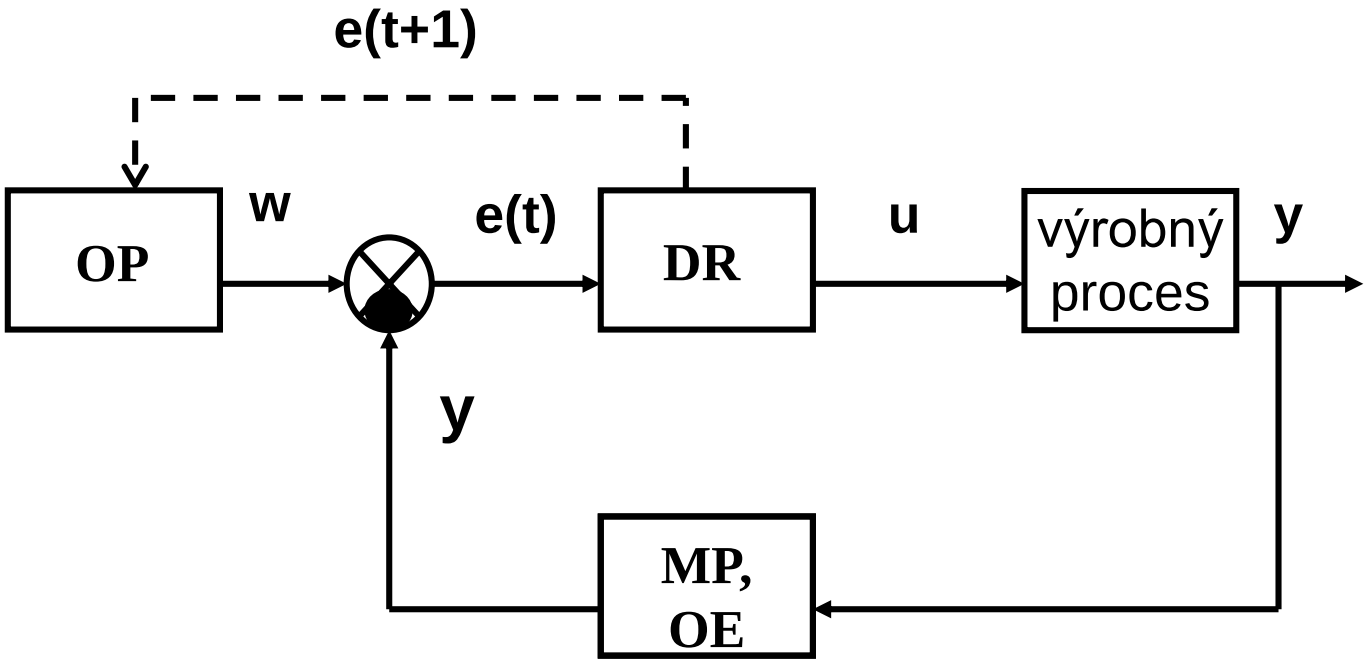
- **Operatívne plánovanie** (OP) – plní funkciu definovania cieľov výrobného procesu.
- Tieto ciele sú dané vo forme operatívnych plánov (rozvrhov) (w), ktoré sú výstupmi operatívneho plánovania.



- **Dispečerské riadenie** (DR) – zabezpečuje prenos cieľov na výrobný proces. Porovnávanie výstupov (y) výrobného procesu s jeho cieľmi (w), na základe ktorých zisťuje stav a prijíma rozhodnutia (u) tak aby sa dosiahol súlad medzi (y) a (w). Zabezpečuje realizáciu týchto rozhodnutí (u).



- **Monitorovanie procesov a operatívna evidencia** (MP, OE) – zabezpečuje zber dát o priebehu výrobného procesu a ich spracovanie do formy potrebnej pre analýzu a kontrolu výrobného procesu.



Plánovanie vs. rozvrhovanie

- **PLÁNOVANIE** je postup vytvárania plánu.
- **PLÁN** je postupnosť akcií, ktoré je potrebné aplikovať tak, aby sa systém dostal z počiatočného stavu do cieľového stavu pri dodržaní daných ohraničení.
- **ROZVRHOVANIE** je postup vytvárania rozvrhu. Pritom nás okrem postupnosti (vopred daných) úloh zaujíma aj ich umiestenie v čase a na jednotlivé stroje/procesory. **Cieľom je** nájsť (optimálny) rozvrh pri dodržaní daných ohraničení.
- **ROZVRH** je súbor údajov, z ktorého je zrejmé, v ktorých časových intervaloch a kde sa majú úlohy realizovať.

Rozdiely v úlohách rozvrhovania oproti plánovacím úlohám

- **Paralelizmus** – máme viac strojov a niektoré úlohy môžu byť vykonávané naraz.
- **Trvanie akcií** – je pri rozvrhovaní dôležité (pri plánovaní sa neuvažuje).
- **Precedenčné ohraničenia** – sú dané výrobným harmonogramom (niektoré úlohy musia predchádzať iné).
- **Implicitné ohraničenia** – na jednom stroji môžeme v jednom okamžiku vykonávať iba jednu úlohu.
- **Obmedzené zdroje** – obmedzený počet strojov, surovín a pod.
- **Špecializované prostriedky** – niektoré úlohy môžu byť vykonané iba na špecializovanom stroji.
- **Časy medzi akciami** – niektoré úlohy môžu začať až s určitým časovým odstupom po skončení predchádzajúcich úloh.

Charakteristiky úloh rozvrhovania

- **Hlavné charakteristiky**
  1. Stroje (procesory)
  2. Úlohy
- **Doplnkové charakteristiky**
  a) precedenčné ohraničenia (usporiadanie)
  b) disjunktné ohraničenia (implicitné, vyplývajúce zo zdieľanie strojov/procesorov)
  c) zákazky (*jobs*)
  d) pomocné zdroje (*resources*)

1. Stroje (procesory)

- Poznáme dva základné typy strojov (procesorov) z hľadiska rozvrhovania:

  a) paralelné – úloha môže bežať na ľubovoľnom stroji

  b) dedikované (špecializované) – úloha môže bežať len na špeciálne určenom stroji

- Podľa výkonnosti procesorov ich možno rozdeliť na:

  a) **identické** – na každom stroji j trvá spracovanie úlohy i rovnaký čas t_i

  b) **uniformné** – každý stroj (procesor) j má svoju rýchlosť, ktorá nezávisí na úlohe, takže vnáša pri spracovaní úloh konštantné zrýchlenie (resp. spomalenie), ktoré označíme b_j

  c) **nesúvzťažné** – rýchlosť stroja (procesora) j závisí na vykonávanej úlohe i, t.j. čas vykonania úlohy i na stroji j bude t_{ij}

2. Úlohy

- Množina úloh $T = \{T_1, T_2, \ldots, T_n\}$
- Povinné údaje o úlohách T_i:
  1. **Čas spracovania úlohy** t_i (dĺžka trvania – *task duration*) – vo všeobecnosti to môže byť vektor (pre každý stroj, resp. procesor iná dĺžka trvania) $[t_{i1}, t_{i2}, \ldots, t_{im}]^T$
     a) v prípade *identických* strojov/procesorov ide vlastne iba o jednu hodnotu t_i, t.j. $t_{ij} = t_i$ pre všetky stroje/procesory $j = 1 .. m$
     b) *uniformné* stroje/procesory: $t_{ij} = t_i / b_j$
     c) *nesúvzťažné* stroje/procesory: rôzne t_{ij}

2. Úlohy

2. **Čas pripravenosti** r_i (*release time*) – od akého okamžiku je úloha pripravená na realizáciu. Ak sú všetky úlohy pripravené naraz, potom r_i = 0 (pre všetky $i = 1 .. n$)
3. **Požadovaná doba splnenia** d_i (*due date*) – doba dokedy by mala byť úloha splnená.

- Nepovinné údaje, ktoré o úlohách môžu, ale nemusia byť zadané:
  4. **Dodacie časy** $\overline{d}_i$ (deadline) – neprekročiteľné časy ukončenia.
  5. **Priority** w_i – významnosť úlohy

Doplnkové charakteristiky

1. **Precedenčné ohraničenia** (usporiadanie)
   - $T_i < T_j$ (úloha T_i musí byť vykonaná pred úlohou T_j)
2. **Disjunktné ohraničenia** (zdieľanie strojov/procesorov)
   - $T_i < T_j \ \vee \ T_j < T_i$ (úlohy T_j a T_i sa neprekrývajú)
3. **Zakázky** (jobs)
   - Úlohy sa rozdeľujú do skupín (zákaziek) – používajú sa pri rozvrhovaní na dedikovaných strojoch
4. **Pomocné zdroje** (resources)
   - Napríklad spotreba materiálu, alebo energie

Typy rozvrhovacích úloh

1. **Paralelné procesory (stroje)**
   a) Rozvrhovanie **na jednom procesore** (jednostupňová výroby)
   b) Rozvrhovanie **na viacerých procesoroch** (viacstupňová výroba)
2. **Dedikované procesory (stroje)**
   - úlohy sa rozdeľujú do skupín, tzv. zákaziek: $J_k = [T_{1,k}, \ldots, T_{nk,k}]$
   - každá úloha v rámci zákazky J_k beží na inom stroji
   - Rozlišujeme **3 základné typy týchto úloh** – a) **open shop**, b) **flow shop** a c) **job shop** (viď. ďalej)

Rozvrhovanie na dedikovaných procesoroch (strojoch)

Typ rozvrhovacej úlohy	Počet úloh v rámci zákaziek n_k	Poradie úloh v rámci zákaziek J_k
a) **Open shop**	***Rovnaké*** pre všetky zákazky J_k	***Ľubovoľné*** *(napr. rozvrh hodín v škole)*
b) **Flow shop**	***Rovnaké*** pre všetky zákazky J_k	***Pevne dané, rovnaké*** pre všetky zákazky J_k *(napr. pásová výroba)*
c) **Job shop**	***Rôzne*** pre jednotlivé zákazky J_k	***Pevne dané, rôzne*** pre jednotlivé zákazky J_k *(napr. zákazková výroba)*

Rozvrh

- **ROZVRH** (R) **–** je súbor údajov, z ktorého je zrejmé, v ktorých časových intervaloch sa majú jednotlivé úlohy realizovať
- Nech $c_i(R)$ je čas ukončenia úlohy T_i v rozvrhu R. Potom je zrejmé, že každý ***prípustný rozvrh*** R je daný n-ticou [$c_1(R)$, ... , $c_n(R)$] za predpokladu, že spĺňa všetky ohraničenia.
- ***Dominantná množina rozvrhov*** (Dom) – je taká množina, že pre každý rozvrh R, ktorý nie je z dominantnej množiny Dom existuje taký rozvrh S z dominantnej množiny Dom taký, že pre každú úlohu T_i (i = 1 .. n) platí $c_i(S) \leq c_i(R)$ (tj. že v rozvrhu S nekončí neskôr ako v rozvrhu R).

$$\forall R \notin Dom : \{\exists S \in Dom, \quad c_i(S) \leq c_i(R), \quad \forall i = 1,..,n\}$$

Optimálny rozvrh

- **Kriteriálna funkcia** – je definovaná na množine všetkých prípustných rozvrhov spravidla ako nejaká reálna funkcia f času ukončenia jednotlivých úloh, t.j.

$$F(R) = f(c_1(R), \ldots , c_n(R))$$

- **Optimálny rozvrh** – je taký prípustný rozvrh, pre ktorý daná kriteriálna funkcia F nadobúda minimum na množine všetkých prípustných rozvrhov.
- **Regulárna kriteriálna funkcia** – $F(R)$ je regulárna vtedy, ak nie je možný jej nárast bez toho, aby sa nepredĺžil termín ukončenia aspoň jednej úlohy.

Regulárne kriteriálne funkcie (1)

$$F(R) = f\left(f_1(c_1(R)), \ \ldots \ , f_n(c_n(R))\right)$$

- Najčastejšie používané kriteriálne funkcie f (pričom f_i je tzv. funkcia nákladov, napr. c_i, l_i, f_i) sú tri:

I. **Suma (napr. C, L , F, T , n_T)**: $f = \sum_{i=1}^{n} f_i(c_i(R))$

II. **Maximum (napr. C_{max}, L_{max} , F_{max})**: $f = \max_i f_i(c_i(R))$

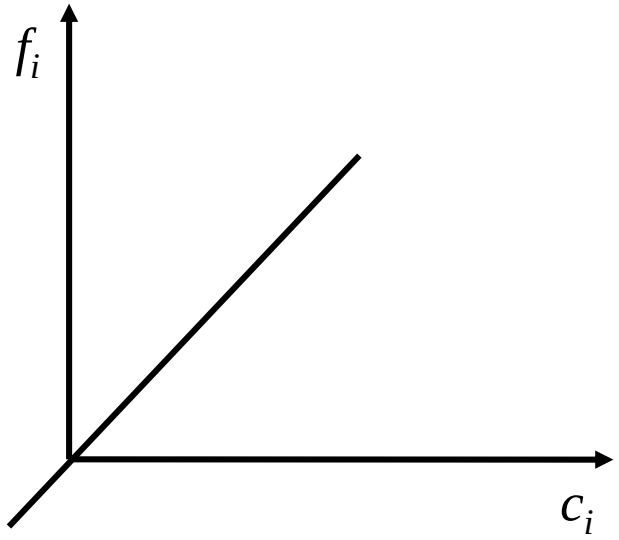
III. **Priemerná hodnota (napr. $\bar{C}$, $\bar{L}$, $\bar{F}$, $\bar{T}$)**: $f = \frac{\sum_{i=1}^{n} f_i(c_i(R))}{n}$

resp. vážený priemer X_w: $f = \frac{\sum_{i=1}^{n} w_i \cdot f_i(c_i(R))}{\sum_{i=1}^{n} w_i}$

Regulárne kriteriálne funkcie (2)

- C (completion time – čas ukončenia)

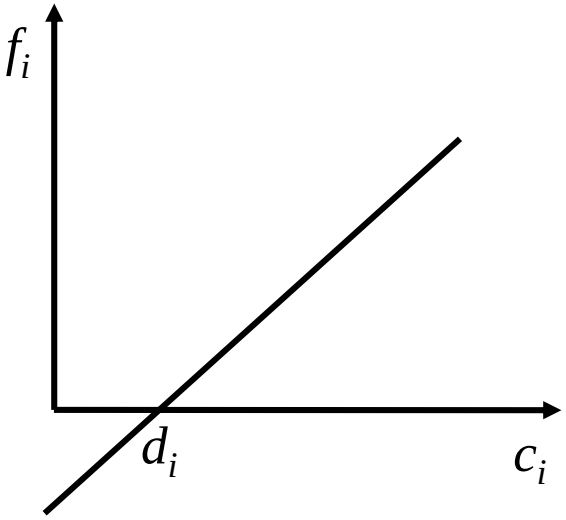
$$f_i = c_i(R) = c_i$$
$$f_i = w_i \cdot c_i(R)$$



Regulárne kriteriálne funkcie (2)

- C (completion time – čas ukončenia)
- L (lateness time – oneskorenie)

$$f_i = l_i = c_i - d_i$$
$$f_i = w_i \cdot (c_i - d_i)$$



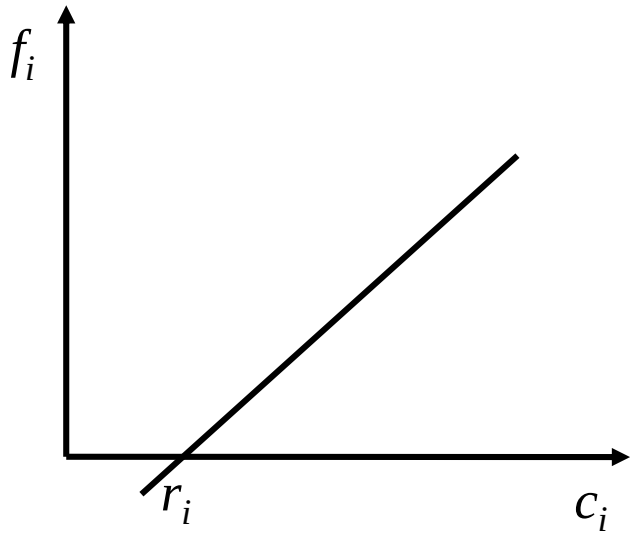
Regulárne kriteriálne funkcie (2)

- C (completion time – čas ukončenia)
- L (lateness time – oneskorenie)
- F (flow time – dĺžka spracovania)

$$f_i = c_i - r_i$$

$$f_i = w_i \cdot (c_i - r_i)$$

$$\overline{F} = \frac{F}{n} \quad resp. \quad F_W = \frac{\sum w_i \cdot (c_i - r_i)}{\sum w_i}$$



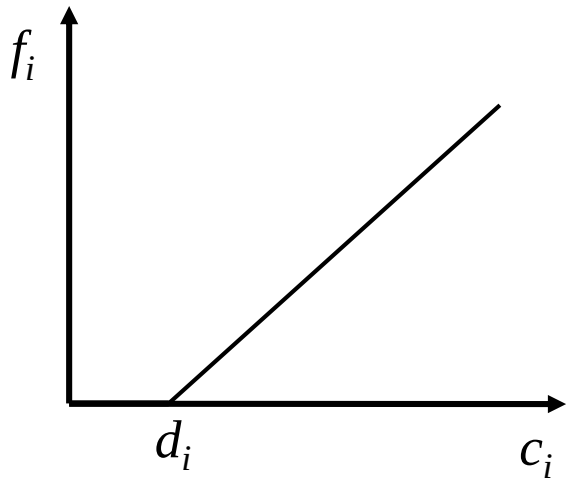
Regulárne kriteriálne funkcie (2)

- C (completion time – čas ukončenia)
- L (lateness time – oneskorenie)
- F (flow time – dĺžka spracovania)
- T (tardeness time – dĺžka omeškania)

$$f_i = \max(0, c_i - d_i)$$

$$f_i = \max(0, w_i \cdot (c_i - d_i))$$

$$\overline{T} = \frac{T}{n} \quad resp. \quad T_W = \frac{\sum w_i \cdot f_i}{\sum w_i}$$



Regulárne kriteriálne funkcie - sumár

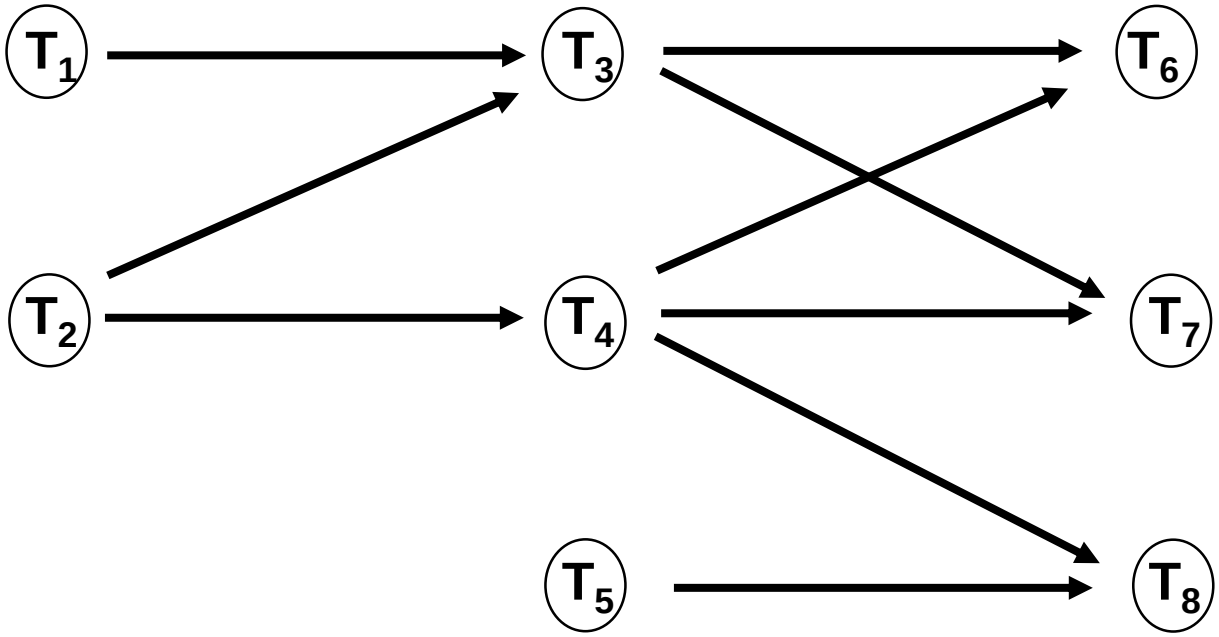
- C (completion time – čas ukončenia)
  - berú sa priamo hodnoty c_i
- L (lateness time – oneskorenie)
  - berú sa hodnoty $c_i - d_i$
- F (flow time – dĺžka spracovania)
  - berú sa hodnoty c_i - r_i
- T (tardeness time – dĺžka omeškania)
  - berú sa iba kladné hodnoty $c_i - d_i$
- n_T (počet omeškaných úloh)
  - počet kladných hodnôt $c_i - d_i$

Príklad

- Máme 3 identické paralelné procesory a daných 8 úloh s týmito parametrami:
  - m = 3, paralelné procesory, P = {P_1, P_2, P_3}
  - n = 8, úlohy, T = {T_1, T_2, …, T_8}
  - t_i = [3, 4, 1, 2, 1, 2, 3, 2] - časy spracovania úloh
  - r_i = 0 (i = 1, ... , 8) - časy pripravenosti
  - d_i = [5, 4, 5, 3, 7, 6, 9, 12] – požadované časy ukončenia úloh
  - w_i = [1, 2, 1, 3, 1, 2, 2, 2] – priority úloh
  - {T1< T3, T2< T3, T2< T4, T3< T6, T3< T7, T4< T6, T4< T7, T4< T8, T5< T8} – precedencie

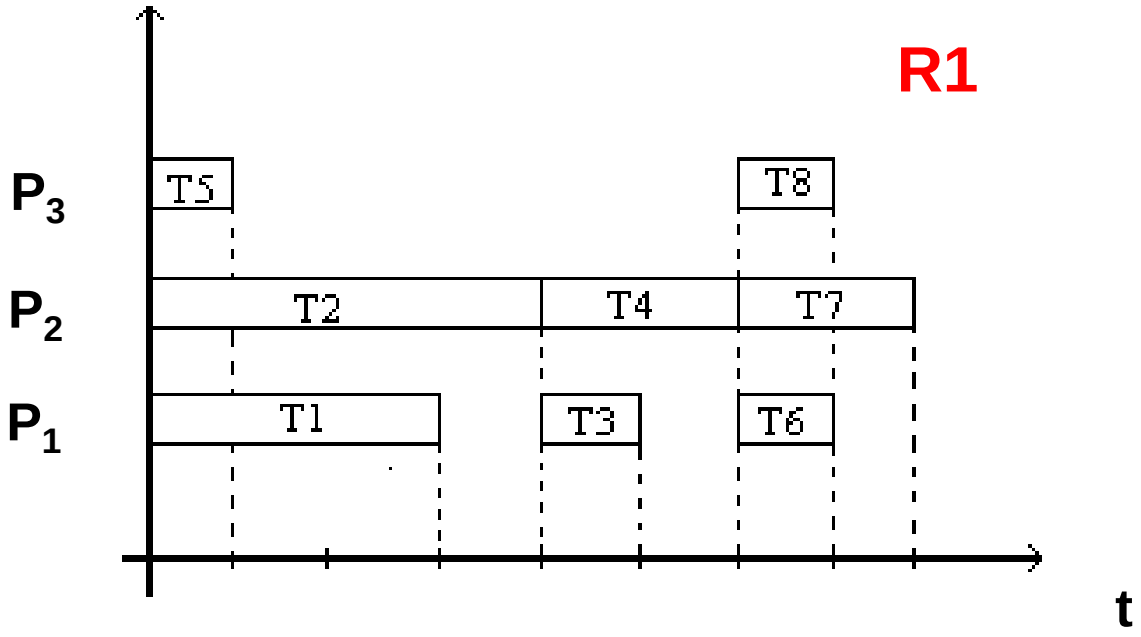
Precedenčný graf

- Úlohy sú reprezentované uzlami v grafe
- Precedencie sú reprezentované orientovanými hranami
- Graf konštruujeme postupne, pričom si uzly rozdelíme na vstupné, výstupné a medziľahlé



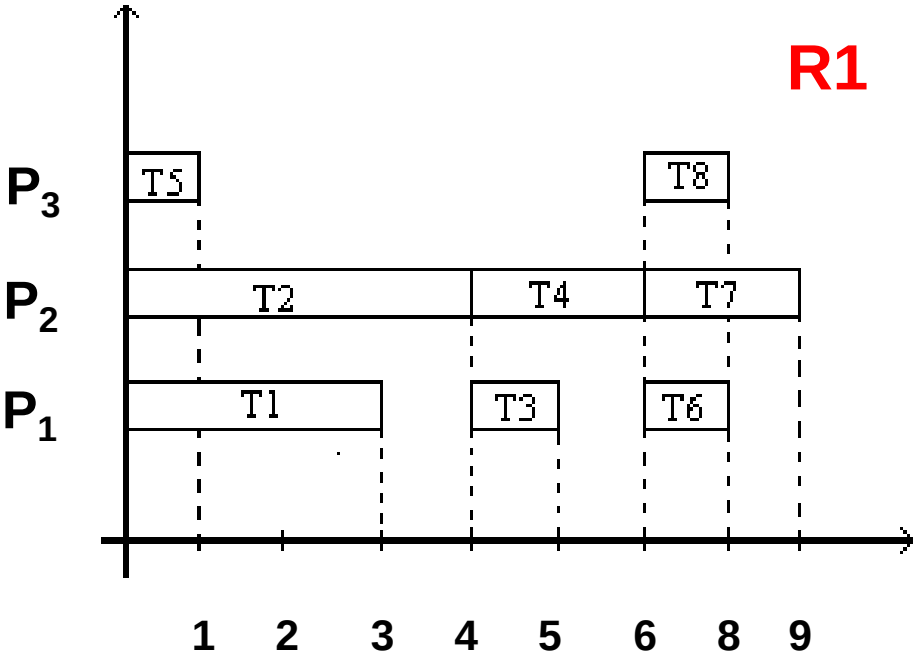
Ganttov diagram

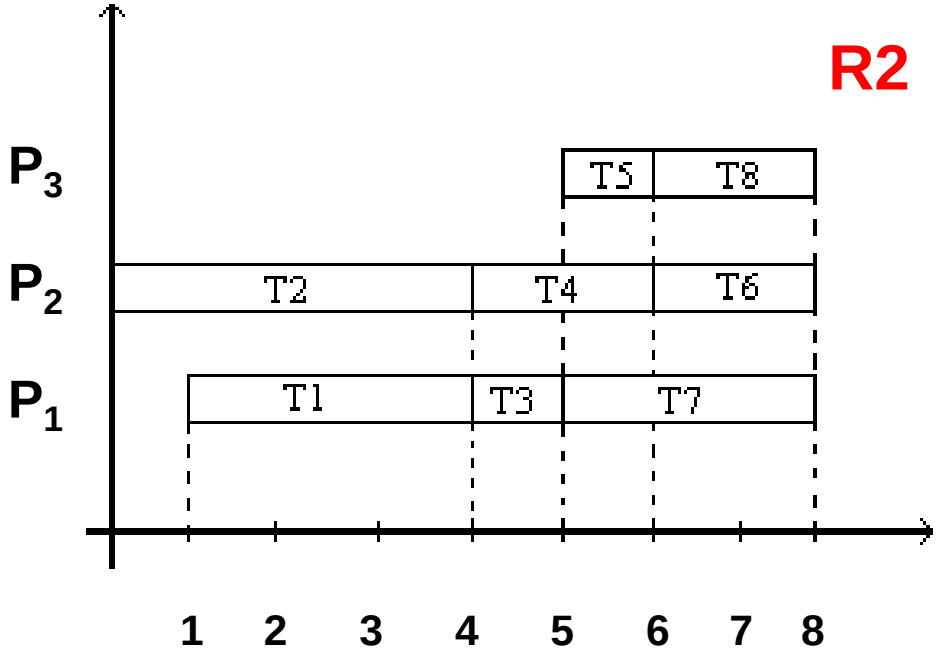
- Je grafickou reprezentáciou rozvrhu (os x reprezentuje čas na osi y sú jednotlivé stroje/procesory)
- Rozvrh môžeme skonštruovať na základe precedenčného grafu pričom dodržiavame precedencie a úlohy zaradzujeme na voľný stroj v najskoršom možnom čase



Príklad

- Vypočítajte hodnoty rôznych kriteriálnych funkcií pre rozvrhy R1 a R2 (typu *C, F, L, T*) a rozvrhy podľa nich porovnajte (v prípade rozvrhu R2 bola jedna precedencia zrušená).





	R1	R2
c_i(Ri)	[3, 4, 5, 6, 1, 8, 9, 8]	[4, 4, 5, 6, 6, 8, 8, 8]
d_i(Ri)	[5, 4, 5, 3, 7, 6, 9, 12]	[5, 4, 5, 3, 7, 6, 9, 12]
l_i(Ri)	[-2, 0, 0, 3, -6, 2, 0, -4]	[-1, 0, 0, 3, -1, 2, -1, -4]
f_i(Ri)	[3, 4, 5, 6, 1, 8, 9, 8]	[4, 4, 5, 6, 6, 8, 8, 8]
C (= F)	**44**	49
C_{max} (= F_{max})	9	**8**
$\overline{C} = \overline{F}$	**5,5**	6,125
L	**-7**	-2
L_{max}	**3**	**3**
$\overline{L}$	**-0,875**	-0,25
T	**5**	**5**
T_{max}	**3**	**3**
n_T	**2**	**2**
E	**12**	7

Rozvrhovanie na jednom stroji (typ úlohy 1a) – s prerušením

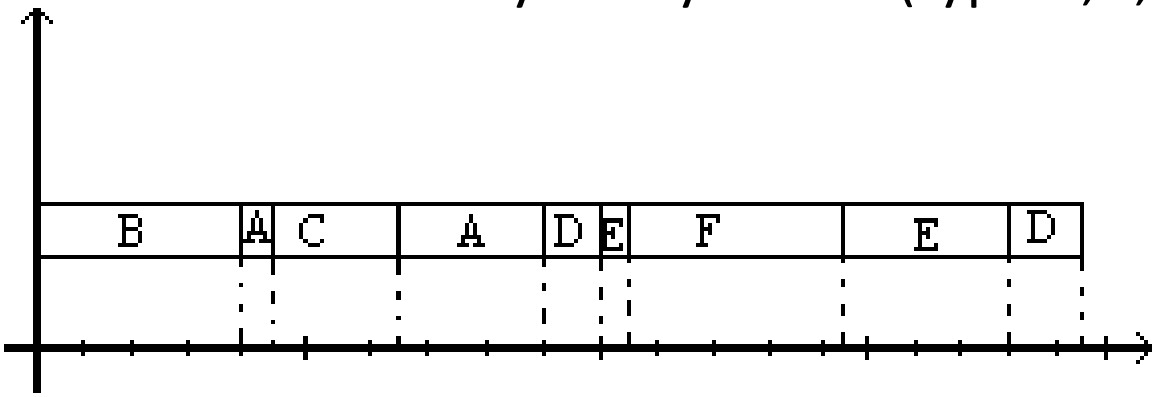
- JACKSONOV ALGORITMUS:
  - Máme n - úloh, rôzne r_i a d_i. Potom algoritmus pre nájdenie optimálneho rozvrhu v zmysle kritéria L_{max} funguje takto:

1. Vždy aktivuj úlohu s najskoršou dobou ukončenia (d_i).
2. Akonáhle začne byť úloha T_i pripravená a procesor je obsadený úlohou T_j, pozastav úlohu T_j v prospech úlohy T_i práve vtedy, ak čas ukončenia i-tej úlohy je skorší ako čas ukončenia j-tej úlohy, inak ponechaj bežať úlohu T_j.

Príklad – jednostupňová výroba

Úloha	t_i	r_i	d_i
A	6	4	32
B	8	0	27
C	4	9	22
D	5	15	43
E	8	20	38
F	8	21	36

- Vypočítajte rôzne typy kriteriálnych funkcií pre výsledný rozvrh (typu C, F, L, T)



Rozvrhovanie na jednom stroji (typ úlohy 1a) – bez prerušenia (1)

- Zložitejšia úloha ako v prípade s prerušením, nakoľko ide o permutačnú úlohu (n! možných rozvrhov), ktorú až na špeciálne prípady nemožno riešiť v polynomiálnom čase.
- Niektoré špeciálne prípady:

1. Úlohy T_i (i = 1, ..., n), r_i = 0 (pre všetky i = 1, ..., n), bez zadaných d_i, bez precedencií, bez priorít

a) Z hľadiska kriteriálnej funkcie C_{max}

sú všetky rozvrhy rovnako dobré

b) Z hľadiska kriteriálnej funkcie C

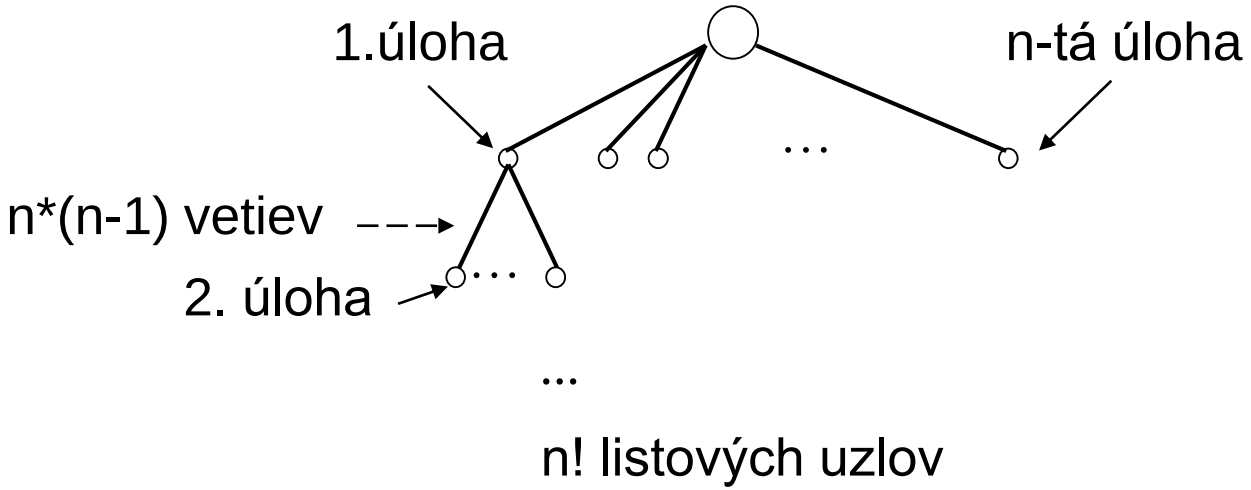
je optimálne **usporiadanie úloh** podľa neklesajúcej postupnosti ich dĺžok trvania, tj.: $t_{(1)} \leq t_{(2)} \leq \ldots \leq t_{(n)}$
čiže **od najkratšej úlohy po najdlhšiu**

Rozvrhovanie na jednom stroji (typ úlohy 1a) – bez prerušenia (2)

2. **Presne ako v predchádzajúcom prípade, ale s prioritami w_i**
   - Z hľadiska kriteriálnej funkcie C_w je optimálne **usporiadanie úloh** podľa nerastúcej postupnosti ich priorít, tj.: $w_{(1)} \geq w_{(2)} \geq \ldots \geq w_{(n)}$ čiže **od najvyššej priority po najnižšiu**
3. **Úlohy T_i (i = 1, ..., n), r_i = 0 (pre všetky i = 1, ..., n), ale rôzne d_i, bez precedencií, bez priorít**
   - Z hľadiska kriteriálnej funkcie L_{max} existuje viacero heuristík, napr. Moorov algoritmus vychádza z neklesajúcej postupnosti požadovaných časov ukončenia úloh, t.j. $d_{(1)} \leq d_{(2)} \leq \ldots \leq d_{(n)}$

Rozvrhovanie na jednom stroji (typ úlohy 1a) – bez prerušenia (3)

- Všetky ostatné úlohy vedú na permutačné rozvrhy a je možné ich riešiť napríklad metódou vetvenia a medzí, ktorá sa snaží efektívne prehľadať nasledujúci priestor prehľadávania:



Rozvrhovanie na viacerých (*nielen*) paralelných procesoroch

- Úlohy sa najprv **usporiadajú podľa zvolenej heuristiky** a potom sa priraďujú zaradom vždy na ten procesor, ktorý sa najskôr uvoľní. Pritom sa používajú rôzne heuristiky, napr.:
  - **LPT** (Longest Processing Time) - vyber úlohu s najdlhším trvaním (t_i)
  - **SPT** (Shortest Processing Time) - ... s najkratším trvaním (t_i)
  - **EST** (Earliest Starting Time) - ... s najskorším časom začiatku (r_i)
  - **LST** (Latest Starting Time) - ... s najneskorším časom začiatku (r_i)
  - **EFT** (Earliest Finishing Time) - ... s najskorším časom ukončenia (d_i)
  - **LFT** (Latest Finishing Time) - ... s najneskorším časom ukončenia (d_i)
  - **MWR** (Most Work Remaining) - vyber úlohu s najdlhšou zvyškovou prácou (súčet trvaní úloh, ktoré ešte musia byť vykonané za vybranou úlohou) (precedencie, resp. usporiadanie v zákazkách)

Príklad algoritmu s použitím heuristiky LPT

Pri inej heuristike stačí zmeniť toto (spôsob usporiadania)

begin

vytvor zoznam úloh usporiadaných

od najdlhšej po najkratšiu, t.j.: $t_1 \geq t_2 \geq \ldots \geq t_n$

for $j = 1$ **to** m $S_j = 0$;

$j := 1$

repeat

urči také k, **že** $S_k = \min_{1 \leq i \leq m}\{S_i\}$

priraď úlohu T_j **(prvá v aktuálnom zozname) na procesor** k

$S_k := S_k + t_j$;

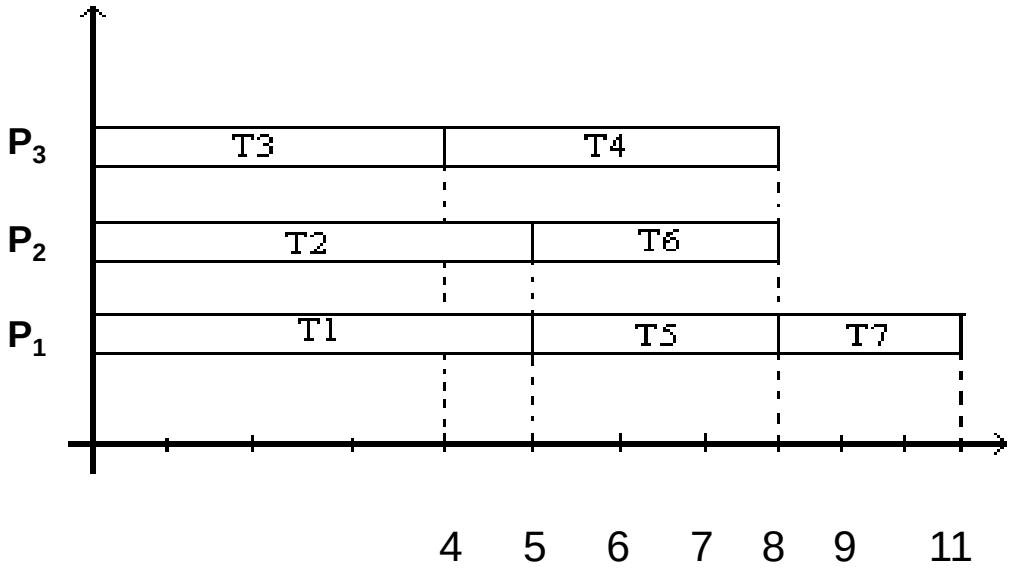
$j := j + 1$;

until $j = n$;

end;

Príklad (1)

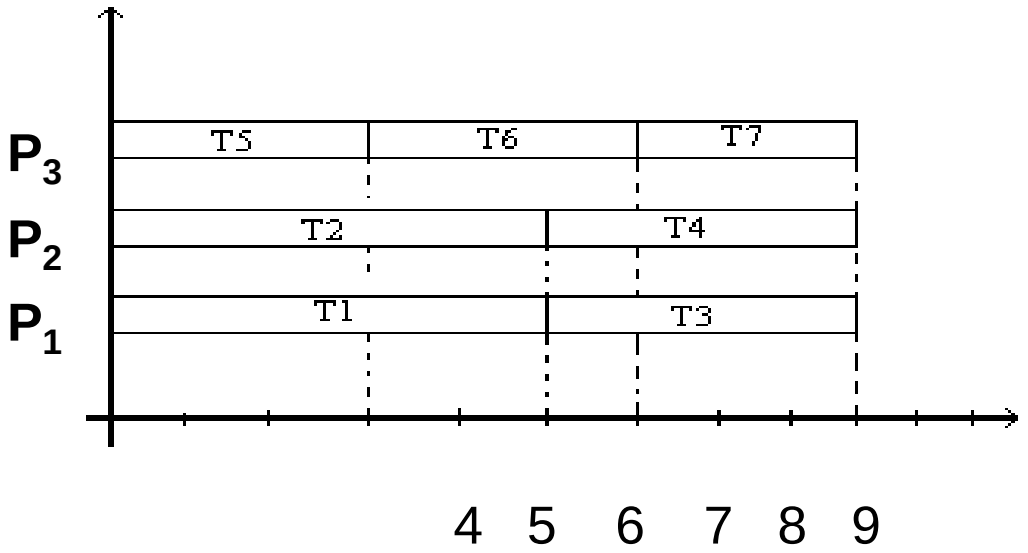
- Majme 3 paralelné procesory, na ktorých je potrebné rozvrhnúť 7 úloh s takýmito dĺžkami trvania: t_1=5, t_2=5, t_3=4, t_4=4, t_5=3, t_6=3, t_7=3
- Rozvrh podľa heuristiky LPT:



- Zostrojte rozvrh podľa heuristiky SPT
- Porovnajte oba rozvrhy podľa jednotlivých typov kriteriálnych funkcií

Príklad (2)

- Ale optimálny rozvrh v zmysle kritéria C_{max} je nasledovný:



- Dá sa dokázať, že rozvrh vygenerovaný podľa LPT nie je od optimálneho horší o viac ako:

$$Q_{LPT} = \frac{4}{3} - \frac{1}{3m}$$ (*m* je počet strojov)

t.j. pre $m = 3$: $$Q_{LPT} = \frac{4}{3} - \frac{1}{9} = \frac{12-1}{9} = \frac{11}{9}$$

Rozvrhovanie na viacerých dedikovaných procesoroch (typ úlohy 2b = flow shop)

1. **Flow shop**
   - Vo všeobecnosti ide o kombinatorickú optimalizáciu, existuje len niekoľko málo prípadov riešiteľných v polynomiálnom čase.
   - Jedným z takýchto špeciálnych prípadov je flow shop na dvoch procesoroch (s ľubovoľným počtom zákaziek *J*, všetky úlohy sú k dispozícii v čase 0), ktorý sa rieši Johnsonovým algoritmom.

Rozvrhovanie na viacerých dedikovaných procesoroch (typ úlohy 2b = flow shop)

- **Johnsonov algoritmus**:
  1. Z množiny všetkých zákaziek J vytvoríme dva zoznamy:

     $L_1 = \{J_i \mid t_{1i} \leq t_{2i}\}$ a $L_2 = J - L_1$
  2. Zoznam L_1 usporiadame podľa neklesajúcich procesných časov t_{1i} (t.j. od najkratšej operácie t_{1i} po najdlhšiu) a zoznam L_2 podľa nerastúcich časov t_{2i} (t.j. od najdlhšej operácie t_{2i} po najkratšiu)
  3. Optimálny rozvrh v zmysle kritéria C_{max} je tvorený zreťazením usporiadaných zoznamov L_1 a L_2

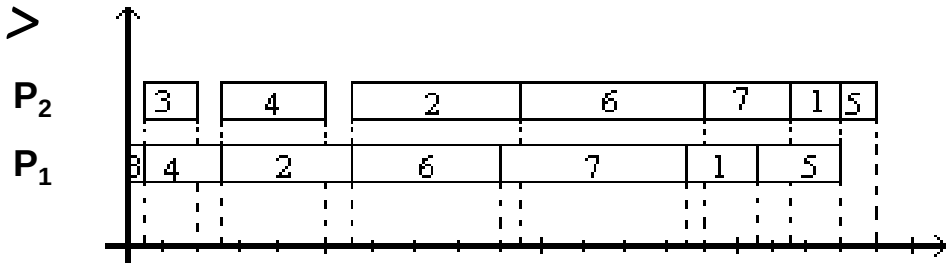
Príklad – flow shop na 2 procesoroch

- Majme 2 procesory a 7 zákaziek J_i (i = 1 .. 7) s danými procesnými časmi:

J_i	1	2	3	4	5	6	7
P_{1i}	4	6	1	4	5	7	9
P_{2i}	3	8	3	5	2	9	5

- Johnsonov algoritmus:
  1. $L_1 = \{J_2, J_3, J_4, J_6\} \Rightarrow L_2 = \{J_1, J_5, J_7\}$
  2. Usporiadané: $L_1 = \langle J_3, J_4, J_2, J_6 \rangle$, $L_2 = \langle J_7, J_1, J_5 \rangle$
  3. Takže výsledný optimálny rozvrh podľa C_{max} zodpovedá zreťazeniu L_1 a L_2

     $R = \langle J_3, J_4, J_2, J_6, J_7, J_1, J_5 \rangle$



- Skúste nájsť lepší rozvrh ak by bola úloha definovaná ako Open shop

Rozvrhovanie na viacerých dedikovaných procesoroch (typ úlohy 2a = open shop)

2. Open shop

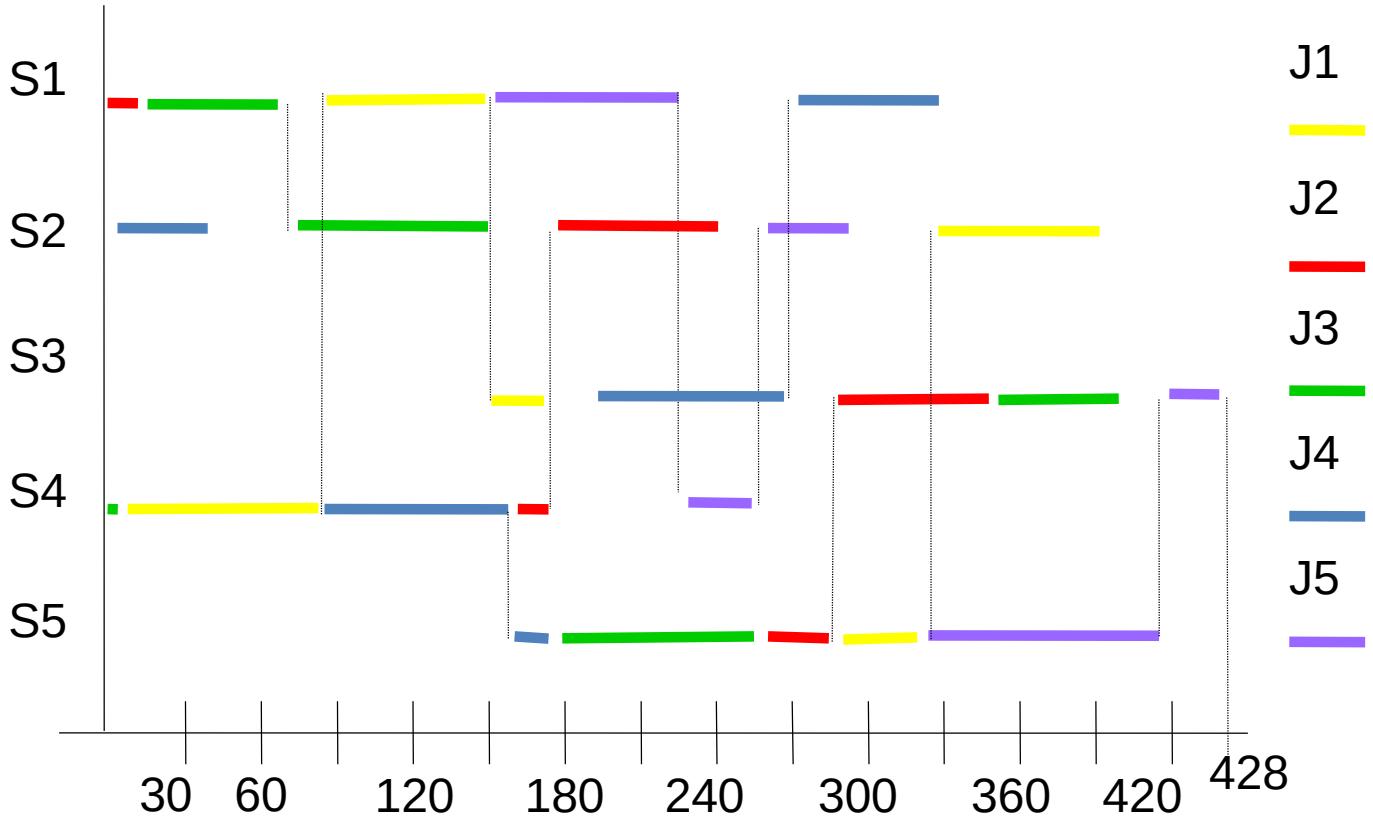
- Tieto úlohy môžeme považovať za istú analógiu úloh flow shop s tým, že nezáleží na poradí úloh.
- Riešenie úlohy flow shop je určite aj riešením open shop.
- Otázkou ostáva, či neexistuje lepšie riešenie.
- Podobne ako u flow shop riešenie v polynomiálnom čase je známe len pre prípad dvoch procesorov.

Rozvrhovanie na viacerých dedikovaných procesoroch (typ úlohy 2c = job shop)

3. Job shop

- *m* strojov, *n* výrobkov,
- výroba každého výrobku je členená na *m* operácií, každá je vykonávaná na inom stroji,
- trvanie operácií je presne dané,
- operácie bez prerušenia,
- každý stroj môže v jednom okamihu spracovávať nanajvýš jednu operáciu,
- presne stanovené poradia operácií pre jednotlivé výrobky (pritom pre každý výrobok to môže byť iné)
- Najzložitejší typ úloh, špeciálnym prípadom je iba prípad 2 zákaziek spracovávaných na ľubovoľnom počte procesorov.

Príklad rozvrhu (Ganttov diagram) pre úlohu job-shop rozmeru 5 x 5



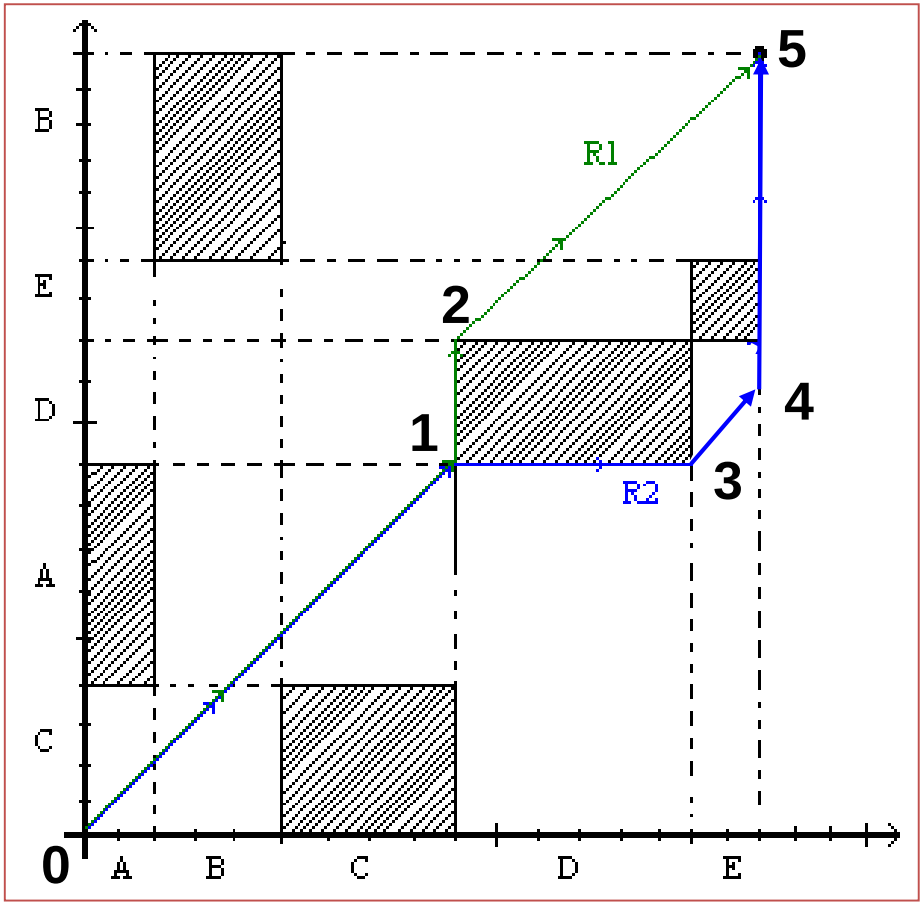
Job shop – špeciálny prípad (2 zákazky)

- Na vodorovnú os vynesieme časy spracovania úloh v rámci prvej zákazky v predpísanom poradí podľa jednotlivých procesorov.
- Na zvislú os vynesieme časy spracovania úloh v rámci druhej zákazky v predpísanom poradí podľa jednotlivých procesorov.
- Vyšrafujeme tzv. „zakázané oblasti“, t.j. oblasti v ktorých súradnice bodov na vodorovnej aj zvislej osi patria tomu istému procesoru.
- Rozvrh znázorňujeme tzv. „pracovnou čiarou“ - ide o lomenú čiaru pozostávajúcu z troch typov úsekov:
  - Úseky pod 45° uhlom zodpovedajú paralelnému spracovávaniu oboch úloh.
  - Vodorovný úsek zodpovedá spracovávaniu prvej zákazky, zatiaľ čo druhá čaká na uvoľnenie procesora.
  - Zvislý úsek znamená spracovávanie druhej zákazky, zatiaľ čo prvá čaká na uvoľnenie procesora.
- Najkratšia pracovná čiara potom zodpovedá optimálnemu rozvrhu.

Príklad:

J_1	Poradie	A	B	C	D	E	$\sum$
	Trvanie	2	3	4	6	2	17

J_2	Poradie	C	A	D	E	B	$\sum$
	Trvanie	4	5	3	2	6	20



$R_1: 0 \rightarrow 1 \rightarrow 2 \rightarrow 5$

$R_2: 0 \rightarrow 1 \rightarrow 3 \rightarrow 4 \rightarrow 5$

Dĺžka:

$R_1: 9 + 3 + 8 = 20$

$R_2: 9+6+2+1+2+6 = 26$

- Optimálny rozvrh je R_1
- Nakreslite Ganttov diagram pre R_1

Ďalšie metódy rozvrhovania (1)

1. **Úplné metódy**, ktoré zaručujú nájdenie (optimálneho) riešenia ak existuje
   - metóda vetvenia a medzí
   - úlohy s ohraničeniami
2. **Neúplné metódy**, ktoré neprehľadávajú celý priestor prehľadávania, iba jeho časť s tým že zaručujú iba nájdenie suboptimálneho riešenia (podrobne o týchto metódach pojednáva predmet Heuristické optimalizačné procesy)
   - genetické algoritmy
   - hill climbing
   - simulované žíhanie
   - tabu search

Rozvrhovanie ako úloha s ohraničeniami

- Ako **premenné** možno zvoliť časy kedy sa začne spracovávať daný výrobok na danom pracovisku (začiatok vykonávania *i*-tej operácie označme T_i).
- **Ohraničenia** sú potom reprezentované ako nerovnice, napr.
  - $T_1 \geq 7$ (ak je dané r_i – *t.j. vo všeobecnosti* $T_i \geq r_i$)
  - $T_3 + 4 \leq 10$ (ak je dané d_i – *t.j.* $T_i + t_i \leq d_i$)
  - $T_1 + 2 \leq T_2$ (precedencia, t.j. $T_i + t_i \leq T_j$)
  - $\forall\ k \neq l$ zdieľajúce jeden stroj:
    $T_k + t_k \leq T_l$ alebo $T_l + t_l \leq T_k$

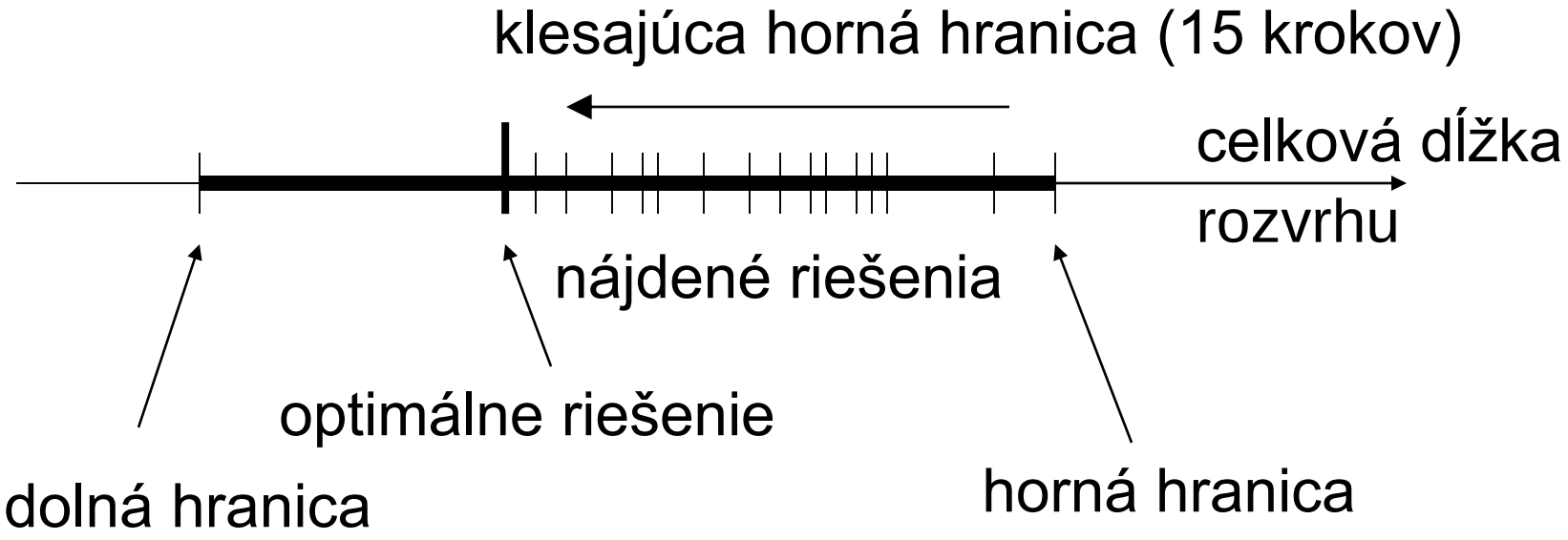
Ďalšie metódy rozvrhovania (2)

1. **Výpočtový čas verzus optimálnosť riešenia**
   - Pre nájdenie zaručene optimálneho riešenia je nutné použiť niektorú z úplných metód. Avšak tie narážajú na kombinatorickú explóziu. Čas výpočtu totiž exponenciálne narastá s veľkosťou úlohy (hlavným zdrojom zložitosti je počet disjunktných ohraničení).
2. **Špecifikácia problému** - každá z metód si vyžaduje svoju reprezentáciu úlohy.
   - Lineárne programovanie narába s množinou lineárnych nerovníc a kriteriálnou funkciou.
   - Metódu vetvenia a medzí možno použiť na každý optimalizačný problém, ak je k dispozícii spôsob, ako ohodnotiť kvalitu čiastočného riešenia.
   - Hill climbing, simulované žíhanie a tabu search záleží od funkcií susednosti, ktoré systém používa pre nájdenie nového riešenia.
   - Efektívnosť genetických algoritmov veľmi záleží na tom, ako sú reprezentovaní kandidáti na riešenie a ako je definovaná vyhodnocovacia funkcia.

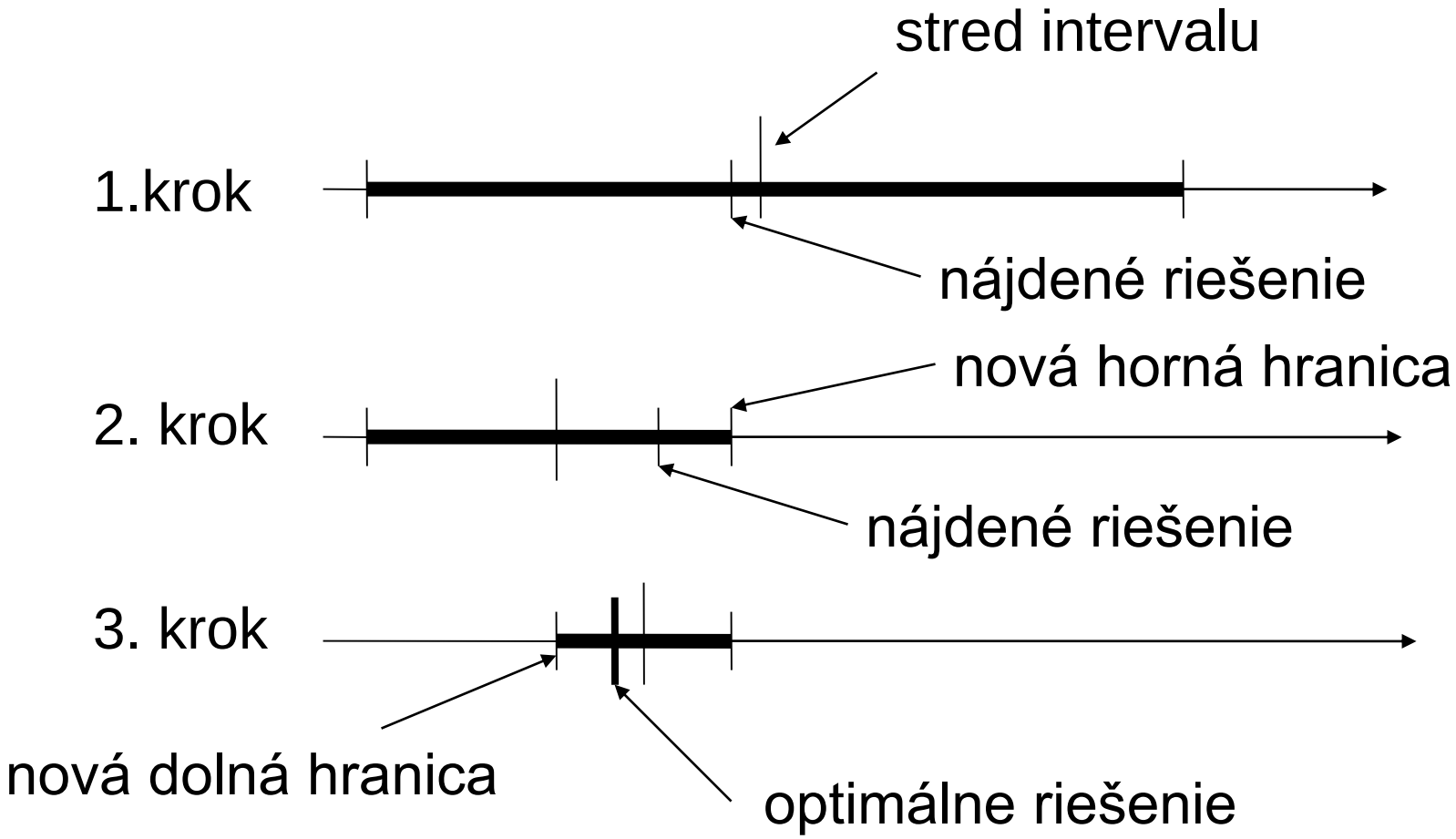
Niektoré naše výsledky pri riešení rozvrhovacích úloh

Optimalizačný algoritmus **min_max**

(zodpovedá klasickej metóde vetvenia a medzí)



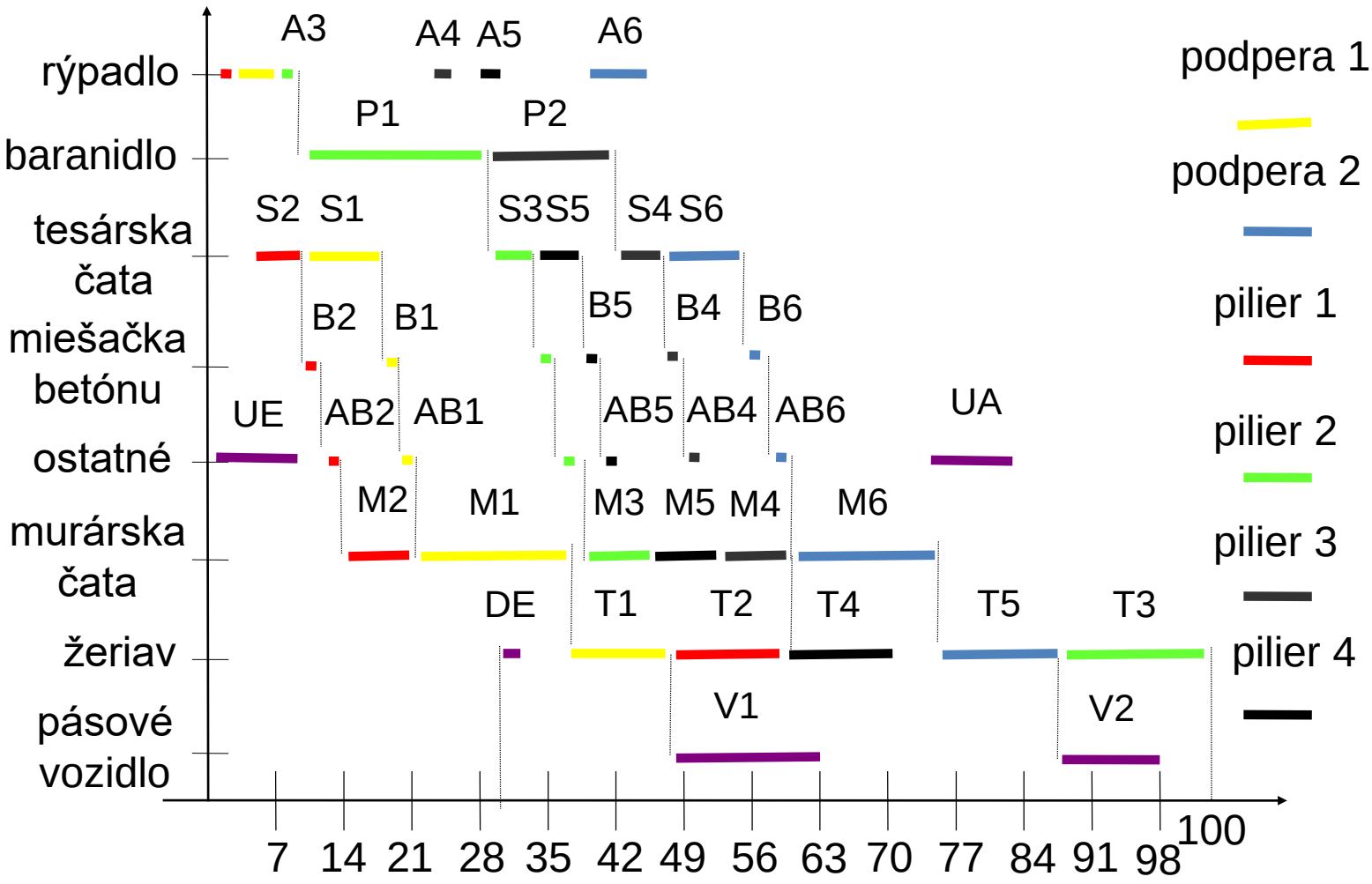
Nový optimalizačný algoritmus **log_min_max**



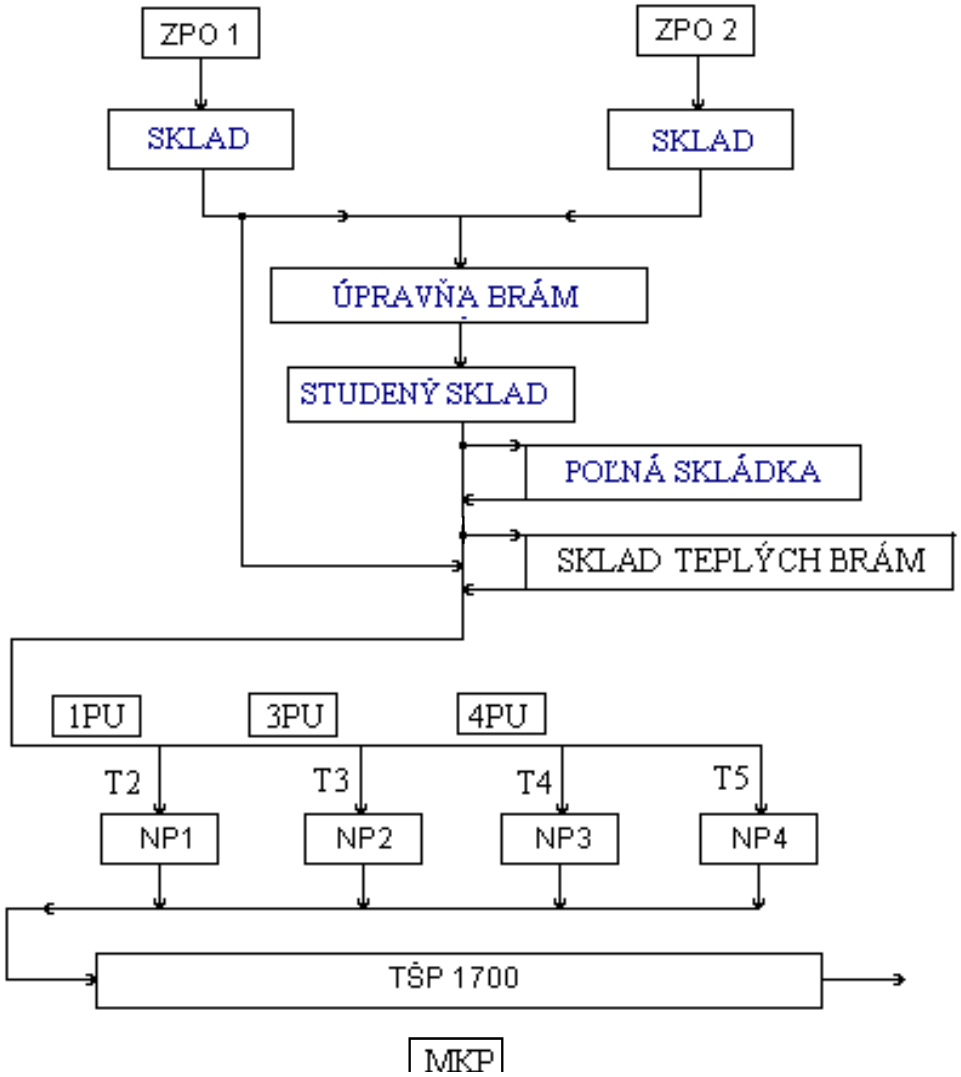
Výsledky testov

úloha	MIN-MAX		LOG-MIN-MAX		LOG-MINIMIZE	
	čas (s)	návraty	čas (s)	návraty	čas (s)	návraty
8-8-0	30.75	**69**	12.74	96	**10.25**	746
8-8-1	40.63	384	**5.55**	**95**	5.89	319
8-8-2	98.50	1331	48.71	**774**	**32.65**	1238
8-8-3	70.34	2019	**7.90**	**79**	9.99	744
8-8-4	68.70	3654	5.95	**56**	**5.47**	676
8-8-5	114.01	6534	24.26	**467**	**21.51**	873
8-8-6	31.13	50	3.80	**6**	**3.41**	452
8-8-7	42.06	356	**9.55**	**104**	11.01	755
8-8-8	29.75	**73**	**12.21**	173	12.50	604
8-8-9	109.03	1813	34.84	**667**	**30.94**	1073
10-10-0	808.42	13517	129.45	**1245**	**108.28**	2130
10-10-1	547.48	10287	51.69	**288**	**32.65**	883
10-10-2	1064.87	20233	499.71	8446	**351.28**	**8037**
10-10-3	1010.60	19271	324.59	**3965**	**289.85**	4851
10-10-4	2307.45	111701	**128.99**	**1907**	156.26	4506
10-10-5	382.99	8450	73.73	**830**	**56.30**	3286
10-10-6	126.64	1425	19.92	**148**	**18.68**	1043
10-10-7	10647.00	**116817**	10239.5	151256	**8404.28**	15082
10-10-8	590.05	27063	99.97	**1846**	**89.90**	2577
10-10-9	6256.24	111350	2565.99	38861	**1228.24**	**29234**
Most	**28821.2**	**109805**	29103.2	142772	32730.9	14277

Optimálny rozvrh výstavby mostu



Zavážanie brám do narážacích pecí




```
rozvrh_zavazania(S) :-
    pocet_peci(F), dlzka(L),
    bramy_vo_vnutri(S1),
    grafikon(G),
    defininuj_struktury(G, F, L, S2),
    definuj_ohranicenia(S2),
    spoj(S1, S2, S),
    definuj_naklady(S, Naklady),
    kontrola_obsadenia(S2, S),
    kontrola_nakladov(S, Naklady),
    hladaj_optimum(S, Naklady),
    vysledok(S).
```