Rozvrhování

Zdeněk Hanzálek a Přemysl Šůcha
hanzalek@fel.cvut.cz

ČVUT FEL Katedra řídicí techniky

9. června 2011

1. Základní notace
2. Rozvrhování na jednom zdroji
   - Minimalizace C_{max}
     - Bratleyův algoritmus pro $1\left|r_j, \widetilde{d}_j\right| C_{max}$
   - Minimalizace $\sum w_j C_j$
     - Metoda větví a mezí s LP pro $1\left|\text{prec}\right| \sum w_j C_j$
3. Rozvrhování na paralelních identických zdrojích
   - Minimalizace C_{max}
4. Project Scheduling
   - Temporální omezení
   - Minimalizace C_{max}
   - Formulace pomocí ILP pro $PS1\left|\text{temp}\right| C_{max}$
   - Formulace pomocí ILP pro $PSm, 1\left|\text{temp}\right| C_{max}$
   - Modelování pomocí temporálních omezení
     - $PS1\left|\text{temp}\right| C_{max}$ - jeden zdroj
     - $PSm, 1\left|\text{temp}\right| C_{max}$ - m dedikovaných zdrojů

Rozvrhování - základní pojmy a předpoklady

- **množina n úloh** $\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$
- m typů zdrojů (procesorů, strojů, lidí,...), každý s kapacitou R_k,
  $\mathcal{P} = \left\{P_1^1, \ldots, P_1^{R_1}, P_2^1, \ldots, P_2^{R_2}, \ldots\ldots, P_m^1, \ldots, P_m^{R_m}\right\}$
- **rozvrhování je přiřazování úloh na zdroje v čase**
- každá z úloh musí být **dokončena**
  - na rozdíl od plánování jehož cílem je rozhodnout, které úlohy budou rozvrhovány a prováděny
- v okamžiku spuštění rozvrhovacího algoritmu je množina úloh dána (někdy používáme přesnější pojem **off-line rozvrhování**)
  - na rozdíl on on-line rozvrhování - například rozvrhovač v jádře operačního systému, který podle určité politiky (např. podle priority úloh) rozvrhuje nově vznikající úlohy
- výsledkem je rozvrh určující na jakém zdroji úloha poběží a kdy poběží - často reprezentujeme pomocí **Ganttova diagramu**

Základní a přídavná omezení

Základní omezení:

- každá úloha je v danou chvíli prováděna **maximálně jedním zdrojem** (úloha je uvnitř sekvenční)
- každý zdroj je v danou chvíli schopen provádět **maximálně jednu úlohu**

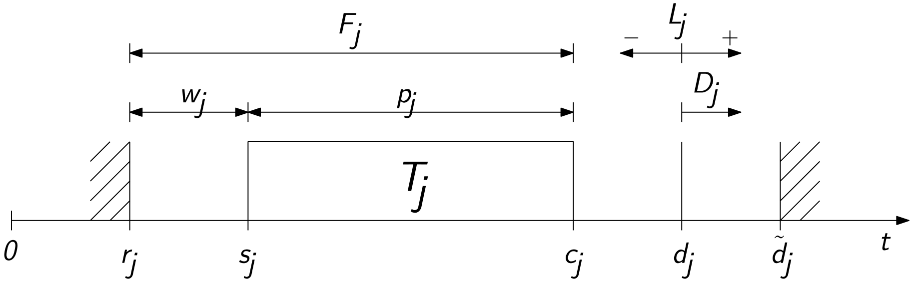
Některá přídavná omezení:

- úloha T_i musí být vykonávána v časovém intervalu $\left\langle r_i, \widetilde{d}_i \right\rangle$
- pokud existuje relace následnosti z úlohy T_i do úlohy T_j, neboli $T_i \prec T_j$, potom vykonávání úlohy T_j nesmí začít před dokončením úlohy T_i
- v případě nepreemptivního rozvrhování nesmí být žádná z úloh přerušena
- v případě preemptivního rozvrhování musí být počet přerušení konečný

Parametry úlohy

- **release time** r_j
- **start time** s_j
- **completion time** C_j (doba dokončení)
- **due date** d_j, úloha T_j by měla být dokončena do této doby
- **deadline** $\widetilde{d}_j$, úloha T_j musí být dokončena do této doby

- waiting time w_j
- **processing time** p_j
- flow time $F_j = C_j - r_j$
- **lateness** $L_j = C_j - d_j$
- tardiness $D_j = \max\{C_j - d_j, 0\}$



Standardní (Grahamova) notace $\alpha\,|\beta|\,\gamma$

Klasifikace rozvrhovacích problémů podle charakteristky
zdrojů | **úloh** | **kritéria**

Například: $P2\,|\text{pmtn}|\,C_{max}$ je rozvrhovací problém na dvou paralelních identických zdrojích, je povolená preempce úloh, kritériem je minimalizace doby dokončení poslední úlohy

α - **zdroje**

- **paralelní zdroje** - úloha může být vykonávána na libovolném zdroji (existuje jeden typ zdroje s kapacitou R, neboli $\mathcal{P} = \left\{P^1, \ldots, P^R\right\}$).
- **dedikované zdroje** - úloha může být vykonávána na jediném zdroji (m typů zdrojů, každý s kapacitou 1, neboli $\mathcal{P} = \{P_1, P_2 \ldots, P_m\}$).
- **Project Scheduling** - m typů zdrojů, každý s kapacitou R_k, neboli $\mathcal{P} = \left\{P_1^1, \ldots, P_1^{R_1}, P_2^1, \ldots, P_2^{R_2}, \ldots\ldots, P_m^1, \ldots, P_m^{R_m}\right\}$.

Charakteristika zdrojů α_1, α_2

α_1	=	1	jeden zdroj
		P	**paralelní identické** zdroje
		Q	**paralelní uniformní** zdroje, kde doba výpočtu je nepřímo úměrná rychlosti zdroje
		R	**paralelní rozdílné** zdroje, kde doba výpočtu je zadána jako matice (zdroje x úlohy)
		O	dedikované zdroje - **open-shop** - úlohy jsou nezávislé
		F	dedikované zdroje - **flow-shop** - úlohy v jobech (skupina úloh) jsou vykonávány ve stejném pořadí, každý zdroj je v rámci jobu využíván právě jednou
		J	dedikované zdroje - **job-shop** - pořadí úloh v jobech je libovolné, zdroj je v rámci jobu využíván libovolně-krát
		PS	**Project Scheduling** - nejobecnější (různé typy a kapacity zdrojů, zobecněné relace následností mezi úlohami)
α_2	=	$\emptyset$	obecný **počet** zdrojů
		2	2 zdroje (resp. jiný specifický počet zdrojů)
		m, R	m typů zdrojů s kapacitou R (u Project Scheduling)

Charakteristika úloh $\beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8$

β_1	$=$	pmtn	povolena **preempce úloh**
		$\emptyset$	bez preempce
β_2	$=$	prec	**libovolné relace následností**
		in-tree,out-tree	orientované stromy
		chain	řetězec
		tmpn	**temporální** rel. násl. (u Project Scheduling)
		$\emptyset$	bez relací následností
β_3	$=$	r_j	release time
β_4	$=$	$p_j = k$	konstantní processing time
		$p_L \leq p_j \leq p_U$	omezený processing time
		$\emptyset$	**libovolný processing time**
β_5	$=$	$\widetilde{d}_j, d_j$	deadline, due-date
β_6	$=$	$n_j \leq k$	omezený počet úloh v Jobu
β_7	$=$	no-wait	buffery s nulovou kapacitou
β_8	$=$	set-up	čas potřebný na **změnu** konfigurace zdroje

Kritérium optimality γ

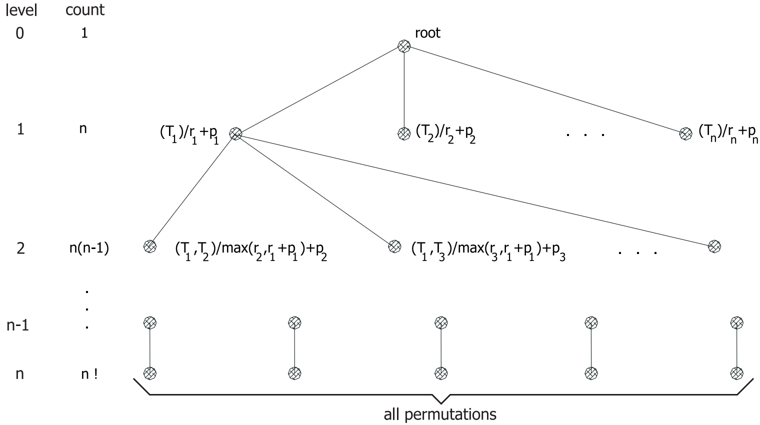
γ	$=$	C_{max}	minimalizace délky rozvrhu $C_{max} = \max\{C_j\}$ (makespan, t.j. doba dokončení poslední úlohy)
		$\sum C_j$	minimalizace součtu dob dokončení
		$\sum w_j C_j$	minimalizace váženého součtu dob dokončení
		L_{max}	minimalizace max. zpoždění $L_{max} = \max\{C_j - d_j\}$
		$\emptyset$	rozhodovací problém, zda existuje proveditelný rozvrh
		$\ldots$	

Například: $P \,||\, C_{max}$ znamená:
libovolný počet paralelních identických zdrojů, bez preempce, nezávislé úlohy (bez relací následností), všechny úlohy jsou k dispozici v čase 0, doby výpočtu úloh jsou různé, kritériem je minimalizace doby dokončení poslední úlohy.

Rozvrhování na jednom zdroji
Minimalizace C_{max}

- $1 \left| prec \right| C_{max}$ - snadný problém
  - pokud jsou úlohy vykonávány v libovolném pořadí, jež vyhoví relacím následností, potom je $C_{max} = \sum_{j=1}^{n} p_j$
- $1 \left|\right| C_{max}$ - snadný problém
- $1 \left| r_j \right| C_{max}$ - snadný problém
  - úlohy jsou vykonány v pořadí neklesajících r_j (neboli T_j s nejmenším r_j nejdříve)
- $1 \left| \widetilde{d}_j \right| C_{max}$ - snadný problém
  - úlohy jsou vykonány v pořadí neklesajících $\widetilde{d}_j$
  - lze použít EDF - Earliest Deadline First - z připravených úloh vyber tu, která má nejmenší deadline
  - proveditelný rozvrh nemusí existovat
- $1 \left| r_j, \widetilde{d}_j \right| C_{max}$ - NP-obtížný problém
  - převod ze 3-Partition problému
  - pro $p_j = 1$ lze nalézt polynomiální algoritmus

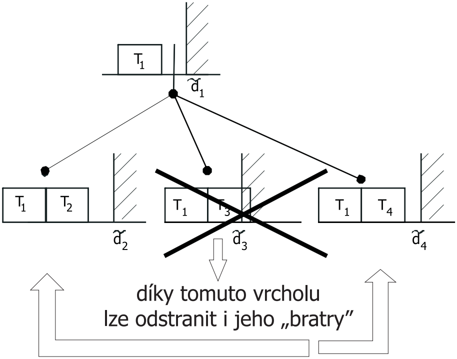
Algoritmus **větví a mezí**, neboli B&B (**Branch&Bound**) algoritmus. Větvení - pokud nepoužijeme meze, pak **větvení vyčíslí (enumerative method) strom (i nepřípustných) řešení**. Každý vrchol má označení: (pořadí úloh v částečném rozvrhu)/doba dokončení poslední z úloh.



Omezení velikosti stromu - prořezávání

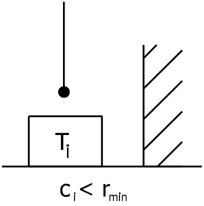
(i) **odstranění vrcholu** překračujícího deadline (včetně jeho "bratrů")

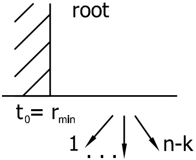
- jestliže se v následující úrovni pod vrcholem v nachází vrchol překračující deadline, potom jsou všechny vrcholy pod vrcholem v odstraněny
- kritická úloha (zde úloha T_3) bude muset být dříve či později vykonána



(ii) **dekompozice problému** díky uvolnění zdroje (idle waiting) - pokud pracovník nečinně čeká na materiál, potom jeho dosavadní práce byla optimální

- pokud pro vrchol v v úrovni k platí, že C_i poslední naplánované úlohy je menší než r_i všech zbylých úloh, potom se není potřeba vracet nad tento vrchol (prohledávat v "bratrech" a "předcích" vrcholu v)
- postačí vytvořit z vrcholu v nový root a prohledat zbylých $n-k$ úrovní (t.j. rozvrhnout $n-k$ úloh) jenom pod tímto vrcholem





Test optimality (konec prohledávání) v Bratleyově alg.

Definice BRTP (Block with Release Time Property)

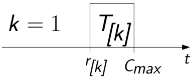
BRTP je množina k úloh, pro kterou platí:

- první úloha $T_{[1]}$ začíná ve svém release date
- všechny úlohy do konce rozvrhu jsou prováděny bez “idle waiting”
- $r_{[1]} \leq r_{[i]}$ pro všechna $i = 2 \dots k$

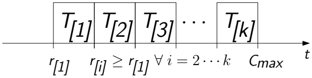
Pozn: “do konce rozvrhu” implikuje, že BRTP je maximálně jeden

Věta - postačující podmínka optimality

Pokud v rozvrhu existuje BRTP, potom je optimální (konči prohledávání).



- toto je časově optimální, jelikož poslední úlohu $T_{[k]}$ nebylo možno vykonat dříve
- na pořadí předchozích úloh nezáleží - viz (ii)



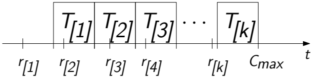
- žádnou z úloh z BRTP nelze vykonat před $r_{[1]}$
- po C_{max} už nic není

BRTP je postačující i nutná podmínka optimality

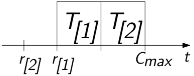
Věta - nutná podmínka optimality

Pokud je rozvrh pro $1 \left| r_j, \widetilde{d}_j \right| C_{max}$ optimální, potom v něm existuje BRTP.

Důkaz sporem: ukážeme, že každý rozvrh bez BRTP je neoptimální. Existují dva případy, kdy je rozvrh bez BRTP:



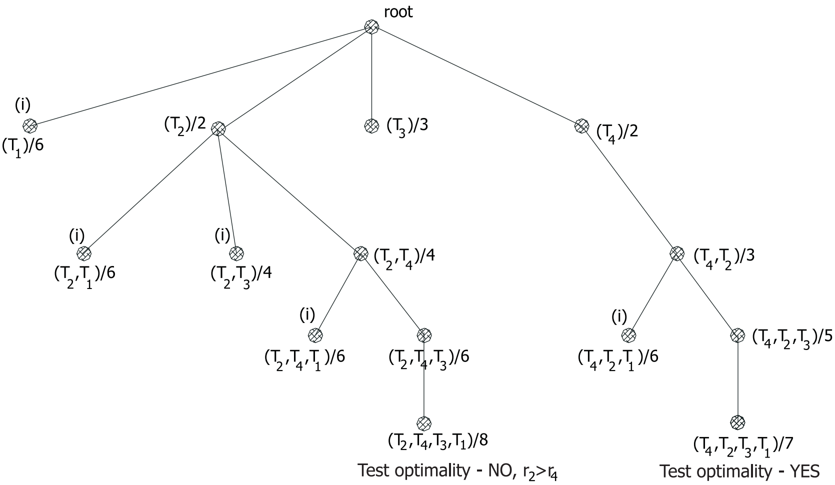
1. blok na konci rozvrhu nezačíná v $r_{[1]}$ neboli:
   - $r_{[1]} < s_{[1]}$
   - $s_{[1]} < r_{[i]} < s_{[i]}\ \forall i = 2 \dots k$ (povšimněte si, že pokud by existovalo $i = 2 \dots k$ takové, že $r_{[i]} = s_{[i]}$ potom by existoval BRTP od $T_{[i]}$)
   - blok lze posunout vlevo při zachování pořadí



2. některá z úloh by mohla být před $T_{[1]}$, neboli existuje $i = 2 \dots k$ takové, že $r_{[i]} < s_{[1]}$
   - rozvrh lze vylepšit umístěním $T_{[2]}$ před $T_{[1]}$

Bratleyův algoritmus - příklad

$r = [4,1,1,0]$, $p = [2,1,2,2]$, $\widetilde{d} = [8,5,6,4]$



Rozvrhování na jednom zdroji
Minimalizace $\sum w_j C_j$

- $1 \,||\, \sum C_j$ - snadný problém
  - pravidlo SPT (Shortes Processing Time first) - přiřadit úlohy v pořadí neklesajících p_j
- $1 \,||\, \sum w_j C_j$ - snadný problém
  - pravidlo WSPT (Weighted SPT) - úlohy v pořadí neklesajících $\frac{p_j}{w_j}$
- $1 \,|r_j|\, \sum C_j$ - NP-obtížný problém
- $1 \,|\text{pmtn}, r_j|\, \sum C_j$ - lze vyřešit modifikovaným SPT
- $1 \,|\text{pmtn}, r_j|\, \sum w_j C_j$ - NP-obtížný problém
- $1 \left| \widetilde{d}_j \right| \sum C_j$ - lze vyřešit modifikovaným SPT
- $1 \left| \widetilde{d}_j \right| \sum w_j C_j$ - NP-obtížný problém
- $1 \,|\text{prec}|\, \sum C_j$ - NP-obtížný problém

Metoda větví a mezí s LP pro $1\,|prec|\sum w_j C_j$

Nejdříve problém formulujeme pomocí ILP:

- zavedeme rozhodovací proměnnou $x_{ij} \in \{0,1\}$ tak, že $x_{ij} = 1$ právě když T_i předchází T_j nebo $i = j$
- relace následností zformulujeme do parametru $e_{ij} \in \{0,1\}$ tak, že $e_{ij} = 1$ právě když v grafu relací následností G existuje orientovaná cesta z T_i do T_j nebo $i = j$
- kritérium - vyjdeme z pozorování, že doba dokončení úlohy T_j se skládá ze součtu p_j a výpočetních dob předchůdců T_j:

$$\begin{aligned} C_j &= \textstyle\sum_{i=1}^{n} p_i \cdot x_{ij} \\ w_j \cdot C_j &= \textstyle\sum_{i=1}^{n} p_i \cdot x_{ij} \cdot w_j \\ J = \textstyle\sum_{j=1}^{n} w_j \cdot C_j &= \textstyle\sum_{j=1}^{n} \sum_{i=1}^{n} p_i \cdot x_{ij} \cdot w_j \end{aligned}$$

  ze všech proveditelných rozvrhů x hledáme ten, který má minimální $J(x)$, neboli $\min_x J(x)$

ILP formulace pro $1\,|prec|\sum w_j C_j$

$$
\begin{array}{lll}
\text{min} & \sum_{j=1}^{n}\sum_{i=1}^{n} p_i \cdot x_{ij} \cdot w_j & \\
\text{subject to:} & & \\
x_{i,j} \geq e_{i,j} & i,j \in 1..n & \text{předchází-li } T_i \text{ před } T_j \text{ v } G, \\
 & & \text{pak ji předchází i v rozvrhu} \\
x_{i,j} + x_{j,i} = 1 & i,j \in 1..n, i \neq j & \text{buď } T_i \text{ je předchůdcem } T_j, \\
 & & \text{nebo naopak} \\
1 \leq x_{i,j} + x_{j,k} + x_{k,i} \leq 2 & i,j,k \in 1..n, & \text{neexistence cyklu v orient.} \\
 & i \neq j \neq k & \text{grafu proměnné } x \\
x_{i,i} = 1 & i \in 1..n &
\end{array}
$$

parameters: $p_{i\in 1..n} \in \mathbb{R}_0^+ \; e_{i\in 1..n, j\in 1..n} \in \{0,1\}$

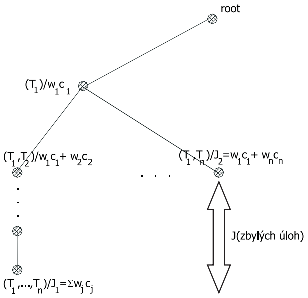
variables: $x_{i\in 1..n, j\in 1..n} \in \{0,1\}$

Algoritmus větví a mezí s LP prořezáváním

V předchozí formulaci ILP relaxujeme na celočíselnost proměnné x

- neboli zavedeme $0 \leq x_{ij} \leq 1$ a $x_{i \in 1..n, j \in 1..n} \in \mathbb{R}$
- tím sice nezískáme užitečné řešení rozvrhu, ale využijeme J^{LP}, hodnotu kritéria této LP formulace na podmnožině $\mathcal{T}$ jako dolní mez "hodnoty zbývající práce"

Algoritmus větví a mezí bude vytvářet strom řešení jako Bratley.



- Nechť J_1 je hodnota nejlepšího doposud známého řešení
- Při procházení stromu odstraníme částečné řešení s hodnotou J_2 nejen když $J_2 \geq J_1$, ale dokonce když $J_2 + J^{LP}(\text{zbylých úloh}) \geq J_1$. Víme totiž, že $J(\text{zbylých úloh}) \geq J^{LP}(\text{zbylých úloh})$ jelikož prostor řešení ILP je podprostorem řešení LP.

Rozvrhování na paralelních identických zdrojích
Minimalizace C_{max}

- $P2 \| C_{max}$ - NP-obtížný problém
  - na dvou paralelních identických zdrojích rozvrhujeme n nepreemptivních úloh při minimalizaci dokončení poslední z nich
  - problém je NP-obtížný, jelikož **2 partition problem** (viz přednáška ILP) lze převést na $P2 \| C_{max}$ tak, že porovnáme nalezené optimální C_{max} s prahem $0.5 * \sum_{i \in 1..n} p_i$.
- $P |\text{pmtn}| C_{max}$ - snadný problém
  - lze vyřešit pomocí **McNaughtonova** algoritmu v $O(n)$
- $P \left|\text{pmtn}, r_j, \widetilde{d}_j\right| C_{max}$ - snadný problém
  - lze formulovat jako úlohu **maximálního toku** (viz přednáška toky)
- $P |\text{prec}| C_{max}$ - NP-obtížný problém
  - **LS** - aproximační algoritmus s faktorem $r_{LS} = 2 - \frac{1}{R}$, kde R je počet paralelních identických zdrojů
- $P \| C_{max}$ - NP-obtížný problém
  - **LPT** - aproximační algoritmus s faktorem $r_{LPT} = \frac{4}{3} - \frac{1}{3R}$
  - **dynamické programování** - Rothkopfův pseudopolynomiální alg.
- $P |\text{pmtn}, \text{prec}| C_{max}$ - NP-obtížný problém
  - úrovňový algoritmus [Muntz,Coffman] s faktorem $r_{MC} = 2 - \frac{2}{R}$

McNaughtonův algoritmus pro $P\,|\text{pmtn}|\,C_{max}$

Vstup: Počet paralelních identických zdrojů R. Počet preemptivních úloh n a výpočetní časy úloh $[p_1, p_2, ..., p_n]$.

Výstup: n-prvkové vektory s^1, s^2, z^1, z^2 kde s_i^1 (resp. s_i^2) je start time první (resp. druhé) části úlohy T_i a z_i^1 (resp. z_i^2) je číslo zdroje na němž je umístěna první (resp. druhá) část úlohy T_i.

$s_i^1 = s_i^2 = z_i^1 = z_i^2 := 0$ pro všechny $i \in 1 \dots n$;
$t := 0; v := 1; i := 1;$
$C^*_{max} = \max\left\{\max_{i=1\dots n}\{p_i\}, \frac{1}{R}\sum_1^n p_i\right\};$
while $i \leq n$ **do**
  if $t + p_i \leq C^*_{max}$ **then**
    $s_i^1 := t; z_i^1 := v;\ t := t + p_i; i := i + 1;$
  else
    $s_i^2 := t; z_i^2 := v;\ p_i := p_i - (C^*_{max} - t); t := 0; v := v + 1;$
  end
end

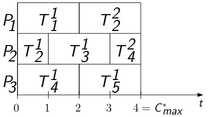
Časová složitost algoritmu je $O(n)$.

McNaughtnonův algoritmus pro $P\,|\mathrm{pmtn}|\,C_{max}$

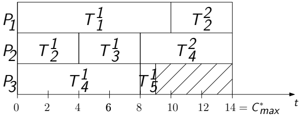
Ve výrazu $C^*_{max} = \max\left\{\max_{i=1\ldots n}\{p_i\}, \frac{1}{R}\sum_1^n p_i\right\}$

- složka $\max_{i=1\ldots n}\{p_i\}$ reprezentuje skutečnost, že úloha je uvnitř sekvenční (i když je rozdělena na různé zdroje, tak nemůže být prováděna paralelně, t.j. naráz na více než jednom zdroji). Povšimněme si, že každá úloha může být rozdělena maximálně na 2 části.
- složka $\frac{1}{R}\sum_1^n p_i$ reprezentuje situaci, kdy všechny zdroje pracují bez prodlevy

Příklad 1:
$p = [2, 3, 2, 3, 2], R = 3$
spočítáme $C^*_{max} = \max\left\{3, \frac{12}{3}\right\} = 4$



Příklad 2:
$p = [10, 8, 4, 14, 1], R = 3$
spočítáme $C^*_{max} = \max\left\{14, \frac{37}{3}\right\} = 14$



List Scheduling - Aproximační algoritmus pro $P\,|prec|\,C_{max}$

Vstup: Počet paralelních identických zdrojů R. Počet nepreemptivních úloh n a výpočetní časy $[p_1, p_2, ..., p_n]$. Or. graf relací následností.
Výstup: n-prvkové vektory s, z kde s_i je start time T_i a z_i je číslo zdroje.

$t_v := 0$ pro všechny $v \in 1 \dots R$; `// čas dostupnosti zdroje`
$s_i = z_i := 0$ pro všechny $i \in 1 \dots n$;
Úlohy uspořádej do listu L;
for $count := 1$ **to** n **do** `// pro všechny úlohy`
  $k = arg\min_{v=1 \dots R} \{t_v\}$; `// vyber zdroj s nejnižším` t_v
  Z listu L vyjmi první úlohu T_i, která je volná;
  $s_i = \max\{t_k, \max_{j \in Pred(T_i)}\{s_j + p_j\}\}; z_i = k$; `// přiřaď` T_i `na` P_k
  $t_k = s_i + p_i$; `// aktualizuj čas dostupnosti zdroje` P_k
end

Úloha T_i je **volná**, pokud se v listu L nenachází žádná úloha ze které vede orientovaná cesta do T_i (t.j. všichni předchůdci T_i již byli vykonáni).
$Pred(T_i)$ je množina indexů úloh, jež jsou **předchůdci** T_i. Složitost $O(n)$.

List Scheduling - Aproximační algoritmus pro $P\,|\text{prec}|\,C_{max}$

List Scheduling (LS) je obecná heuristika používaná v řadě aplikací.

- Máme daný list (uspořádanou n-tici) úloh a ve chvíli, kdy se uvolní některý ze zdrojů, bere si první volnou úlohu z listu.
- Přesnost LS záleží na kritériu a na způsobu seřazení listu.

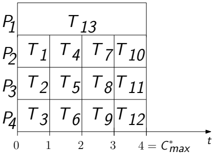
Faktor LS algoritmu [Graham 1966]

Pro $P\,|\text{prec}|\,C_{max}$ (zároveň i pro $P\,||\,C_{max}$) a libovolný (nesetříděný) list L je List Scheduling aproximačním algoritmem s faktorem $r_{LS} = 2 - \frac{1}{R}$

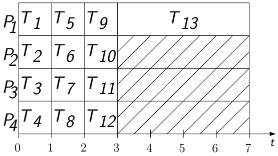
Příklad ilustrující dosažení hranice:

$n = (R-1)\cdot R + 1,$
$p = [1, 1, \ldots, 1, R],$
$\prec$ prázdné.

Obrázky pro $R = 4$ a
$r_{LS} = 2 - \frac{1}{4} = \frac{7}{4}$



$L = [T_n, T_1, \ldots, T_{n-1}]$



$L' = [T_1, T_2, \ldots, T_n]$

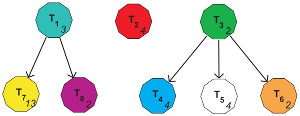
Rozvrhovací anomálie v List Scheduling algoritmu

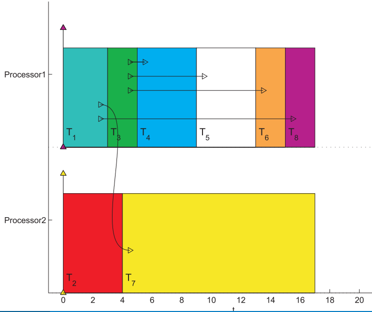
LS algoritmus je závislý nejenom na **pořadí úloh v listu L**, ale navíc vykazuje **anomálie** (t.j. při “uvolnění” vstupních parametrů se C_{max} zvýší) vlivem

1. **zkrácení doby vykonávání** úloh p_i
2. **odstranění některých relací následnosti** $T_i \prec T_j$
3. **zvýšení počtu zdrojů** R

Příklad na anomálie LS algoritmu:

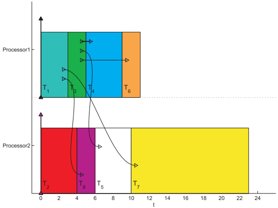
$R = 2, \; n = 8, \; p = [3, 4, 2, 4, 4, 2, 13, 2]$



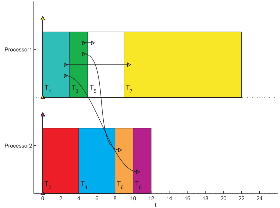


Pro seřazení $L = [T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8]$ nalezne LS optimální řešení s $C^*_{max} = 17$.

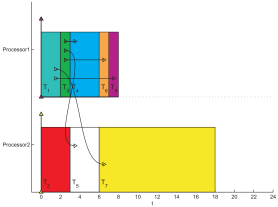
Záměna úloh T_7 a T_8 v listu $L = [T_1, T_2, T_3, T_4, T_5, T_6, T_8, T_7]$.



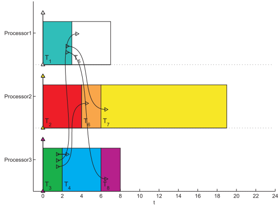
Odebrání relace následnosti $T_3 \prec T_4$.



Snížení p_i všech úloh o jedna.



Přidání zdroje ($R = 3$).



LPT (Longest Processing Time first) - Aproximační algoritmus pro $P \,||\, C_{max}$

Faktor LS algoritmu lze snížit strategií, která se označuje jako **Longest Processing Time first** (LPT)

- v inicializaci LS setřídíme list L v pořadí nerostoucích p_i

Faktor LPT algoritmu [Graham 1966]

LPT Algoritmus pro $P||C_{max}$ je aproximačním algoritmem s faktorem $r_{LPT} = \frac{4}{3} - \frac{1}{3R}$

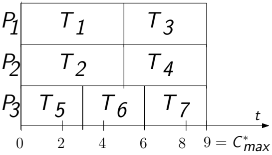
Časová složitost LPT algoritmu je $O(n \cdot log(n))$ díky třídění.

LPT (Longest Processing Time first) - Aproximační algoritmus pro $P||C_{max}$

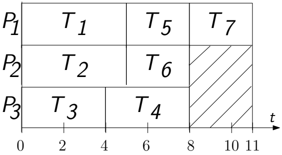
Příklad ilustrující horní hranici:
$p = [2R-1, 2R-1, 2R-2, 2R-2, \ldots, R+1, R+1, R, R, R]$
$n = 2 \cdot R + 1, \prec$ prázdné,

Obrázky pro $R = 3$

optimum:



LPT:



$r_{LPT} = \frac{4}{3} - \frac{1}{9} = \frac{11}{9}$

Faktor LPT algoritmu

Pro velký počet úloh lze nalézt ještě lepší hranici v závislosti na k, počtu úloh přiřazených zdroji, který poslední končí: $r_{LPT} = 1 + \frac{1}{k} - \frac{1}{kR}$

Dynamické programování pro $P \,||\, C_{max}$ [Rothkopf]

Pseudopolynomiální algoritmus - rozsah diskrétních hodnot je omezen pod určitou mez a pro takto omezený problém nalezneme polynomiální algoritmus.

- zavedeme binární proměnnou $x_i(t_1, t_2, \dots, t_R)$ kde
  - $i = 1, 2 \dots n$ je index úlohy
  - $v = 1, 2, \dots R$ je index zdroje
  - $t_v = 0, 1, 2, \dots UB$ je čas zdroje
  - UB je horní odhad C_{max}
- $x_i(t_1, t_2, \dots, t_R) = 1$ iff úlohy $T_1, T_2, \dots, T_i$ mohou být umístěny na zdrojích tak, že každý zdroj P_v je obsazen na intervalu $\langle 0, t_v \rangle$; $v = 1, 2, \dots R$

Dynamické programování pro $P \,||\, C_{max}$ [Rothkopf]

Vstup: Počet paralelních identických zdrojů R. Počet nepreemptivních úloh n a výpočetní časy $[p_1, p_2, ..., p_n]$.

Výstup: n-prvkové vektory s, z kde s_i je start time T_i a z_i je číslo zdroje.

for $(t_1, t_2, \ldots, t_R) \in \{1, 2, \ldots UB\}^R$ **do** $x_0(t_1, t_2, \ldots, t_R) := 0$;
$x_0(0, 0, \ldots, 0) := 1$;
for $i := 1$ **to** n **do** `// pro všechny úlohy`
  for $(t_1, t_2, \ldots, t_R) \in \{0, 1, 2, \ldots UB\}^R$ **do** `// v celém prostoru`
    $x_i(t_1, t_2, \ldots, t_R) := \mathrm{OR}_{v=1}^{R} x_{i-1}(t_1, t_2, \ldots, t_v - p_i, \ldots t_R)$;
    `// v novém prostoru` $x_i() = 1$ `pokud ve starém prostoru`
    `// existoval` $x_{i-1}() = 1$ `‘‘menší’’ o` p_i `v libovolném směru`
  end
end
$C^*_{max} = \min_{x_i(t_1, t_2, \ldots, t_R)=1} \{\max_{v=1,2,\ldots R} \{t_v\}\}$;
Od času C^*_{max} přiřazuj pozpátku úlohy $T_n, T_{n-1}, \ldots, T_1$;

Časová složitost algoritmu je $O(n \cdot UB^R)$. Příklad na tabuli.

Úrovňový algoritmus pro $P\,|\mathrm{pmtn}, \mathrm{prec}|\,C_{max}$ [Muntz,Coffman 1970]

Princip

- úlohy vybíráme **z listu seřazeného podle úrovní**
- **úroveň úlohy** T_j - součet p_i (včetně p_j) po **nejdelší cestě** z T_j do koncové úlohy (úloha bez následovníka)
- v daném okamžiku, když více úloh stejné úrovně má být přiřazeno méně zdrojům, získá úloha z daného zdroje **část kapacity** β
- algoritmus se posune na okamžik τ, **pokud jedna z úloh skončí**, nebo pokud by úloha s nižší úrovní byla vykonávána větší kapacitou β než úloha s vyšší úrovní

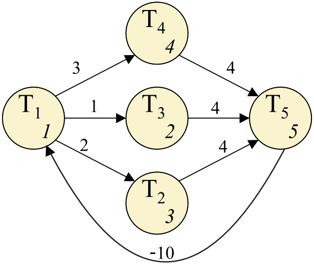
Algoritmus je **exaktní** pro $P2\,|\mathrm{pmtn}, \mathrm{prec}|\,C_{max}$ a $P\,|\mathrm{pmtn}, \mathrm{forest}|\,C_{max}$.
Alg. je **aproximační** pro $P\,|\mathrm{pmtn}, \mathrm{prec}|\,C_{max}$ s faktorem $r_{MC} = 2 - \frac{2}{R}$.
Časová složitost algoritmu je $O(n^2)$.

Vstup: Počet paralelních identických zdrojů R. Počet preemptivních úloh n a výpočetní časy $[p_1, p_2, ..., p_n]$. Orient. graf relací následností.
Výstup: n-prvkové vektory s, z kde s_i je start time T_i a z_i je číslo zdroje.

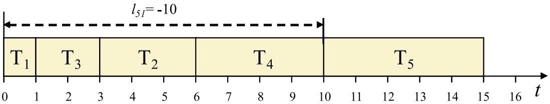
Úrovňový algoritmus pro $P\,|\text{pmtn, prec}|\,C_{max}$

spočítej úroveň všech úloh; t:=0; h:=R; `// h je poč. voln. zdrojů`
while *existuje nedokončená úloha* **do**
  vytvoř $\mathcal{Z}$; `// podmnožina` $\mathcal{T}$ `z volných úloh v čase` t
  while $h > 0$ *and* $|\mathcal{Z}| > 0$ **do** `// volné zdroje a volné úlohy`
    vytvoř $\mathcal{S}$; `// podmnožina` $\mathcal{Z}$ `z úloh nejvyšší úrovně`
    if $|\mathcal{S}| > h$ **then** `// více úloh na méně zdrojů`
      každé úloze z $\mathcal{S}$ přiřaď část kapacity $\beta := \frac{h}{|\mathcal{S}|}$; $h := 0$;
    else
      každé úloze z $\mathcal{S}$ přiřaď jeden zdroj; $\beta := 1$; $h := h - |\mathcal{S}|$;
    end
    $\mathcal{Z} := \mathcal{Z} \setminus \mathcal{S}$;
  end
  spočítej τ; `// okamžik, kdy jedna z úloh skončí`
  sniž úroveň úloh o $(\tau - t) \cdot \beta$; `// vykonaná část úlohy`
  $t := \tau; h := R$;
end
McNaughtonem přerozvrhni části, kde je více úloh na méně zdrojích;

- Množina nepreemtivních úloh $\mathcal{T} = \{T_1, T_2, ..., T_n\}$ je reprezentována vrcholy orientovaného grafu G.
- Každý uzel je ohodnocen dobou vykonávání úlohy p_i.



- Hrany reprezentují temporální omezení. Každá hrana z uzlu T_i do uzlu T_j má přiřazenu váhu l_{ij}.
- Jedno temporální omezení je vyjádřeno jednou nerovnicí $s_i + l_{ij} \leq s_j$.

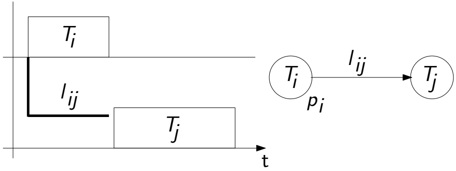


Temporální omezení $s_i + l_{ij} \leq s_j$ s kladným l_{ij}

Temporální omezení (občas se nazývá **zobecněná relace následností** nebo **pozitivní-negativní časová prodleva**)
- omezuje start time jedné úlohy v závislosti na start time druhé úlohy.

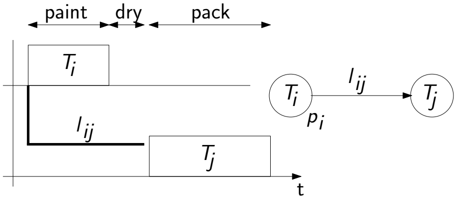
a) $l_{ij} = p_i$

- "normální" relace následností
- vykonávání následující úlohy může začít po dokončení předchozí úlohy



b) $l_{ij} > p_i$

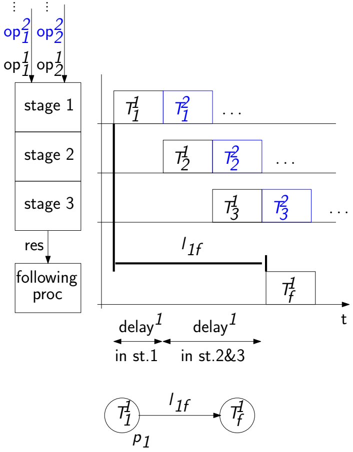
- vykonávání následující úlohy může začít **až nějaký čas** po dokončení předchozí úlohy
- b.1) například schnutí laku po operaci lakování, při dostatečně velkém prostoru pro sušení



Temporální omezení $s_i + l_{ij} \leq s_j$ s kladným l_{ij}

b.2) pipe-lined ALU - další příklad využívající $l_{ij} > p_i$

- předpokládejme, že processing time je totožný ve stage 1, 2 a 3
- **výsledek je k dispozici** l_{1f} tiků poté, co stage 1 načetla operandy
- **stage 1 načítá nové operandy** každých p_1 tiků
- **stage 2 a 3 nemodelujeme**, jelikož máme dostatečně velké množství těchto zdrojů a jsou synchronizovány se stage 1

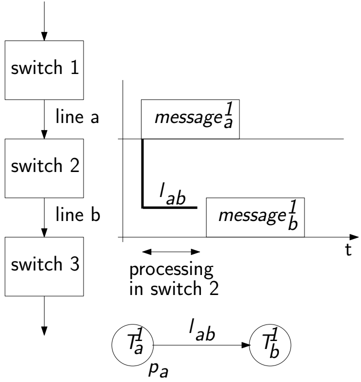


Temporální omezení $s_i + l_{ij} \leq s_j$ s kladným l_{ij}

c) $0 < l_{ij} < p_i$

Dílčí výsledky předchozí úlohy lze použít pro zahájení následující úlohy. Například **cut-through** mechanismus kde switch na výstupním portu zahájí přeposílání zprávy před tím, než na vstupním portu přijme celou zprávu.

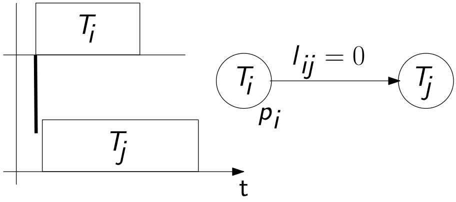
- předpokládáme časem řízený protokol (Protfinet IO IRT)
- zdroje jsou **komunikační linky**
- l_{ab} **reprezentuje dobu průchodu** (jednoho bitu) skrz switch
- **různé části** téže zprávy jsou v danou chvíli komunikovány několika navazujícími linkami



Temporální omezení $s_i + l_{ij} \leq s_j$ s nulovým nebo záp. l_{ij}

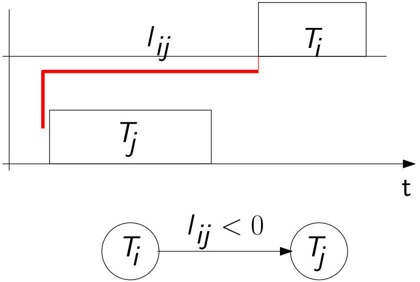
d) $l_{ij} = 0$

- úloha T_i musí začít dříve nebo ve stejný okamžik jako úloha T_j



e) $l_{ij} < 0$

- úloha T_i musí začít dříve nebo **maximálně o** $|l_{ij}|$ **později** než úloha T_j
- pozbývá význam "normální" relace následností (neboli zde T_i nemusí předcházet před T_j)
- představuje **relativní deadline** T_i vůči start-time T_j



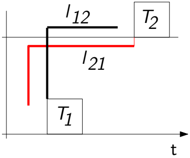
Cykly a relativní časové okno

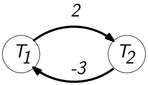
- Neexistence kladného cyklu v or. grafu G
  - je nutnou podmínkou pro rozvrhnutelnost instance
  - je nutnou a postačující podmínkou pro rozvrhnutelnost instance s nekonečnou kapacitou zdrojů (rozvrh je omezen pouze temporálními omezeními - snadná formulace pomocí LP)
- K původnímu or. grafu G lze vytvořit úplný or. graf G' kde váha l_{ij} je délka nejdelší orientované cesty z T_i do T_j v G (pokud neexistuje orientovaná hrana v G nebo v G', potom je $l_{ij} = -\infty$). V dalším textu budeme uvažovat l_{ij} v úplném or. grafu G' nejdelších cest.
  - $s_j \geq \max_{\forall i \in 1 \dots n} l_{ij}$, neboli start time úlohy T_j je zdola omezen nejdelší or. cestou z libovolného vrcholu

Příklad - relativní časové okno

Pokud existuje $l_{ij} \geq 0$ a $l_{ji} < 0$, potom jsou úlohy T_i a T_j navzájem omezeny relativním časovým oknem.

- délka (záporného) cyklu procházejícího těmito vrcholy určuje "volnost" relativního časového okna
- například štukování čerstvě nahozené omítky





- $PS1\,|\text{temp}|\,C_{max}$ - NP-obtížný problém
  - Vstup: Počet nepreemptivních úloh n a výpočetní časy $[p_1, p_2, ..., p_n]$. Temporální omezení daná orientovaným grafem G.
  - Výstup: n-prvkový vektor s, kde s_i je start time úlohy T_i
  - ukážeme časem-indexovanou ILP formulaci a ILP formulaci s relativním pořadím
- $PSm, 1\,|\text{temp}|\,C_{max}$ - NP-obtížný problém
  - Vstup: Počet nepreemptivních úloh n a výpočetní časy $[p_1, p_2, ..., p_n]$. Temporální omezení daná orientovaným grafem G.
    Počet dedikovaných zdrojů m a přiřazení úloh na zdroje $[a_1, a_2, ..., a_n]$, kde a_i je index zdroje, na kterém bude vykonávána úloha T_i.
  - Výstup: n-prvkový vektor s, kde s_i je start time úlohy T_i
  - ukážeme ILP formulaci s relativním pořadím

Formulace pomocí ILP pro $PS1 \, |\text{temp}| \, C_{max}$

Úlohu lze formulovat dvěma způsoby:

- časem-indexovaný model - ILP model je založen na proměné x_{it}, která je rovna 1, pokud úloha T_i má začátek vykonávání $s_i = t$. V ostatních případech je rovna nule. Doba vykonávání je kladné celé číslo.
- model s relativním pořadím - ILP model je založen na proměnné x_{ij}, která je rovna 1, pokud úloha T_i předchází úlohu T_j. V ostatních případech je rovna nule. Doba vykonávání je nezáporné reálné číslo.

Oba modely obsahuji dva typy omezení:

- temporální omezení
- omezení zdroje - zabraňuje překryvu jednotlivých úloh

Časem-indexovaný model pro $PS1\,|\text{temp}|\,C_{max}$

min C_{max}

$$\begin{array}{ll}
\sum_{t=0}^{UB-1}(t \cdot x_{it}) + l_{ij} \le \sum_{t=0}^{UB-1}(t \cdot x_{jt}) & \forall l_{ij} \ne -\infty \text{ a } i \ne j \text{ (temp. omez.)} \\
\sum_{i=1}^{n}\left(\sum_{k=\max(0,t-p_i+1)}^{t} x_{ik}\right) \le 1 & \forall t \in \{0, \dots UB-1\} \text{ (zdroje)} \\
\sum_{t=0}^{UB-1} x_{it} = 1 & \forall i \in \{1, \dots n\} \text{ (} T_i \text{ je rozvržena)} \\
\sum_{t=0}^{UB-1}(t \cdot x_{it}) + p_i \le C_{max} & \forall i \in \{1, \dots n\}
\end{array}$$

variables: $x_{it} \in \{0, 1\},\ C_{max} \in \{0, \dots UB\}$

UB - horní odhad C_{max} (např. $UB = \sum_{i=1}^{n} \max\left\{p_i, \max_{i,j \in \{1,\dots,n\}} l_{ij}\right\}$).

Začátek vykonávání úlohy T_i lze vyjádřit jako $s_i = \sum_{t=0}^{UB-1}(t \cdot x_{it})$.

Model obsahuje $n \cdot UB + 1$ proměnných a $|E| + UB + 2n$ omezení.

Konstanta $|E|$ označuje počet temporálních omezení (hran v G).

Časem-indexovaný model pro $PS1\,|\text{temp}|\,C_{max}$

$\mathcal{T} = \{T_1, T_2, T_3\}$, $p = [1, 2, 1]$, $UB = 5$

T_1 je rozvržen:

	0	1	2	3	4
T_1	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
T_2	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}
T_3	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}

$\Sigma = 1$

Resource constr. v čase 2:

	0	1	2	3	4
T_1	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}
T_2	x_{20}	x_{21}	x_{22}	x_{23}	x_{24}
T_3	x_{30}	x_{31}	x_{32}	x_{33}	x_{34}

$\Sigma \leq 1$

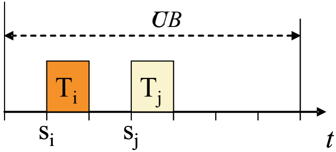
Model s relativním pořadím pro $PS1 \,|\text{temp}|\, C_{max}$

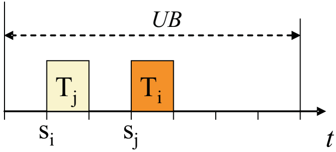
Omezení zdroje: $p_j \leq s_i - s_j + UB \cdot x_{ij} \leq UB - p_i$

Omezení využívá princip “big M” (zde UB - horní odhad C_{max}).

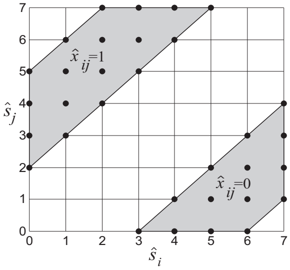
Pokud $x_{ij} = 1$, potom T_i **předchází** úlohu T_j a omezení zdroje má tvar $s_i + p_i \leq s_j$.

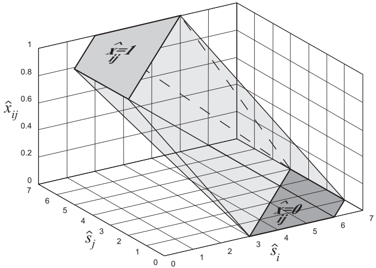
Pokud $x_{ij} = 0$, potom T_i **následuje** úlohu T_j a omezení zdroje má tvar $s_j + p_j \leq s_i$.





Příklad polytopu, který je tvořen podmínkou procesoru pro dvojici úloh T_i a T_j s $p_i = 2$ a $p_j = 3$. Úlohy nejsou navzájem omezeny relací následností a horní odhad délky rozvrhu je $UB = 8$.





Model s relativním pořadím pro $PS1\,|\text{temp}|\,C_{max}$

min C_{max}

$$s_i + l_{ij} \leq s_j \qquad \forall l_{ij} \neq -\infty \text{ a } i \neq j$$
(temporální omezení)

$$p_j \leq s_i - s_j + UB \cdot x_{ij} \leq UB - p_i \qquad \forall i, j \in \{1, \ldots, n\} \text{ a } i < j$$
(omezení zdrojem)

$$s_i + p_i \leq C_{max} \qquad \forall i \in \{1, \ldots, n\}$$

variables: $x_{ij} \in \{0, 1\},\ C_{max} \in \langle 0, UB \rangle,\ s_i \in \langle 0, UB \rangle$

Model obsahuje $n + \left(n^2 - n\right)/2 + 1$ proměnných a $|E| + \left(n^2 - n\right) + n$ omezení. Konstanta $|E|$ označuje počet temporálních omezení (hran v G).

Srovnání modelů

Každý ze zmiňovaných modelů se hodí pro jiný typ úloh se zobecněnými relacemi následností.

Časem-indexovaný model:

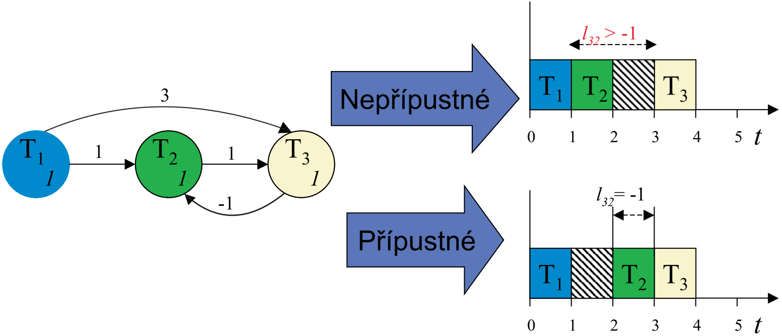
- (+) Snadno jej lze rozšíří na paralelní identické procesory.
- (+) ILP formulace nevyžaduje velké množství omezení.
- (-) Velikost ILP modelu roste s velikostí UB.

Model s relativním pořadím:

- (+) Velikost ILP modelu není závislá na UB.
- (-) Vyžaduje větší množství omezení než časem-indexovaný model.

Test přípustnosti v heuristickém algoritmu

Pokud částečný rozvrh (nalezený například hladovým algoritmem vkládajícím úlohy v topologickém pořadí, tak jak je to nejdřív možné, nebo částečné řešení uvnitř algoritmu větví a mezí) poruší některé časové omezení ještě to neznamená, že příslušné pořadí úloh je nepřípustné.



Pokud bychom znali optimální pořadí úloh v rozvrhu (neboli proměnné x_{ij} by byly konstanty), potom by nalezení start time úloh bylo snadné (například LP formulací zahrnující pouze časová omezení).

Model s relativním pořadím pro $PSm, 1 |\text{temp}| C_{max}$

Součástí vstupu je počet zdrojů m a **přiřazení úloh na zdroje** $[a_1, a_2, ..., a_n]$, kde a_i je index zdroje, na kterém bude vykonávána úloha T_i.

min C_{max}

$s_i + l_{ij} \leq s_j$	$\forall l_{ij} \neq -\infty$ a $i \neq j$ (temporální omezení)
$p_j \leq s_i - s_j + UB \cdot x_{ij} \leq UB - p_i$	$\forall i, j \in \{1, \ldots, n\}$ a $i < j$ a $\underline{a_i = a_j}$ (omezení na každém zdroji zvlášť)
$s_i + p_i \leq C_{max}$	$\forall i \in \{1, \ldots, n\}$

variables: $x_{ij} \in \{0, 1\},\ C_{max} \in \langle 0, UB \rangle,\ s_i \in \langle 0, UB \rangle$

Model obsahuje méně než $n + \left(n^2 - n\right)/2 + 1$ proměnných (záleží na počtu úloh na každém ze zdrojů).

Modelování pomocí temporálních omezení

Pomocí $PS1\left|\text{temp}\right|C_{max}$ lze modelovat:

- $1\left|r_j, \widetilde{d}_j\right|C_{max}$
- rozvrhování na **dedikovaných zdrojích** $PSm, 1\left|\text{temp}\right|C_{max}$

Pomocí $PSm, 1\left|\text{temp}\right|C_{max}$ lze modelovat:

- rozvrhování s **multiprocesorovými úlohami** - jedna úloha vyžaduje více než jeden zdroj
- rozvrhování s **časem na výměnu** - dvě po sobě jdoucí úlohy T_i, T_j musí mít mezi sebou dostatečný časový odstup, který například dovolí vyměnit nástroj.

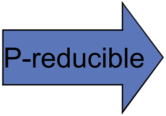
Dokazuje, že $PS1 \left|\text{temp}\right| C_{max}$ je NP-obtížný.

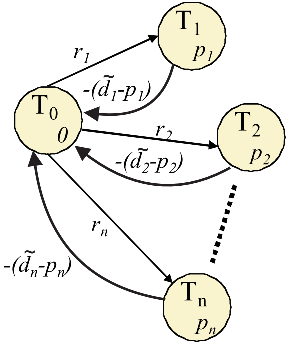
Instance $1 \left| r_j, \widetilde{d}_j \right| C_{max}$

$r = [r_1, r_2, \ldots, r_n]$

$p = [p_1, p_2, \ldots, p_n]$

$\widetilde{d} = [\widetilde{d}_1, \widetilde{d}_2, \ldots, \widetilde{d}_n]$





Převod $PSm, 1 \,|\text{temp}|\, C_{max}$ na $PS1 \,|\text{temp}|\, C_{max}$

Transformace $PSm, 1 \,|\text{temp}|\, C_{max}$ na $PS1 \,|\text{temp}|\, C_{max}$ je založena na převodu rozvrhu **jednoho zdroje do samostatného časového okna**. Neboli rozvrh na zdroji P_j se převede do intervalu $\langle (j-1) \cdot UB, j \cdot UB \rangle$.

Transformace se skládá ze dvou kroků:

- Jsou přidány **pomocné úlohy** T_0 a T_{n+1} s $p_0 = p_{n+1} = 0$.
  - Úloha T_0, prováděná na zdroji P_1, je předchůdcem všech úloh $T_i \in \mathcal{T}$, tj. $s_0 \leq s_i$.
  - Úloha T_{n+1}, prováděná na zdroji P_m, je následníkem všech úloh $T_i \in \mathcal{T}$, tj. $s_i + p_i \leq s_{n+1}$.
- **Původní temporální omezení** jsou transformována na $l'_{ij} = l_{ij} + (a_j - a_i) \cdot UB$.

Převod $PSm, 1 \,|\text{temp}|\, C_{max}$ na $PS1 \,|\text{temp}|\, C_{max}$

Nový začátek vykonávání každé úlohy s_i' na zdroji a_i je:
$s_i' = s_i + (a_i - 1) \cdot UB$.

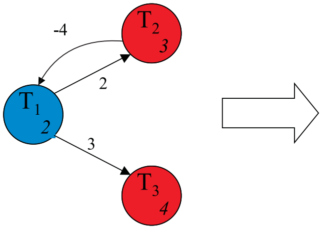
Temporální omezení $s_i + l_{ij} \leq s_j$ jsou transformována na:

$$s_i' - (a_i - 1) \cdot UB + l_{ij} \leq s_j' - (a_j - 1) \cdot UB$$
$$s_i' + l_{ij} + (a_j - a_i) \cdot UB \leq s_j'$$

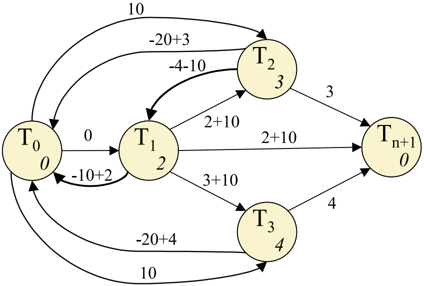
Potom transformované temporální omezení má tvar $s_i' + l_{ij}' \leq s_j'$ kde:

$$l_{ij}' = l_{ij} + (a_j - a_i) \cdot UB$$

Převod $PSm, 1\,|temp|\,C_{max}$ na $PS1\,|temp|\,C_{max}$



2 dedikované procesory



1 procesor

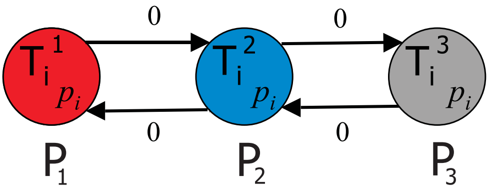
Minimalizace času dokončení úlohy T_{n+1} "tlačí vlevo" úlohy T_1, T_2 a T_3 přes hrany vstupující do T_{n+1}.

Víceprocesorové úlohy

Transformace rozvrhování víceprocesorových úloh na $PSm, 1\,|\text{temp}|\,C_{max}$

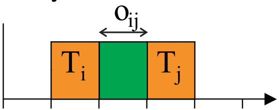
- založíme tolik virtuálních úloh, na kolika zdrojích má běžet víceprocesorová úloha
- zajistíme, aby tyto virtuální úlohy začínaly ve stejný čas - to je realizováno pomocí dvou opačně orientovaných hran s váhou $l_{ij} = l_{ji} = 0$, následkem toho $s_i \leq s_j$ a $s_j \leq s_i$

Př: Úloha T_i vyžaduje ke svému vykonání zdroje $[P_1, P_2, P_3]$.



Čas na výměnu nástroje (changover time, sequence dependent set-up time)

Čas na výměnu nástroje o_{ij} je čas, který je potřeba počkat mezi prováděním úloh T_i a T_j, aby bylo možné například vyměnit nástroj nebo zásobník s barvou.
Jelikož pořadí úloh není dopředu známo, tak nelze ani dopředu určit, který z těchto časů se projeví a který nikoli.



Transformace rozvrhování s časem na výměnu nástroje na $PSm, 1 \,|\text{temp}|\, C_{max}$

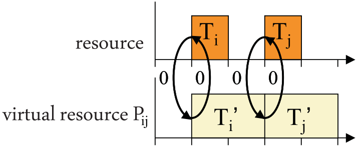
- založíme virtuální zdroj a dvojici prodloužených virtuálních úloh pro každou uspořádanou dvojici úloh s tímto časem
- zavedeme relace nařizující, aby úlohy začínaly ve stejný čas.

Čas na výměnu nástroje

Ke každé dvojici úloh, pro které platí, že jejich čas na změnu nástroje je $o_{ij} > 0$ nebo $o_{ji} > 0$, se zavede virtuální dvojice úloh T'_i a T'_j.

- Úloha T'_i má $p'_i = p_i + o_{ij}$ a úloha T'_j má $p'_j = p_j + o_{ji}$.
- Obě úlohy jsou prováděny na společném virtuálním zdroji P_{ij}.
- Úloha T'_i (resp. T'_j) je svázána s původní úlohou T_i vztahem:

$$s_i \leq s'_i \quad s'_i \leq s_i \quad \text{resp.} \quad s_j \leq s'_j \quad s'_j \leq s_j$$



Literatura

J. Błażewicz, K. Ecker, G. Schmidt, and J. Węglarz.
Scheduling Computer and Manufacturing Processes.
Springer, second edition, 2001.

Klaus Neumann, Christoph Schwindt, and Jürgen Zimmermann.
Project Scheduling with Time Windows and Scarce Resources.
Springer, 2003.