Plánování úloh (na jednom stroji)

11. dubna 2022

Jeden stroj a paralelní stroj

- Dekompoziční problémy pro složité *(flexible) job shop* problémy používají
  - jeden stroj
  - paralelní stroj

  jako podproblémy při řešení
- Metody řešení:
  - řídící pravidla
  - metoda větví & mezí
  - paprskové prohledávání

Řídící pravidla pro jeden stroj: přehled

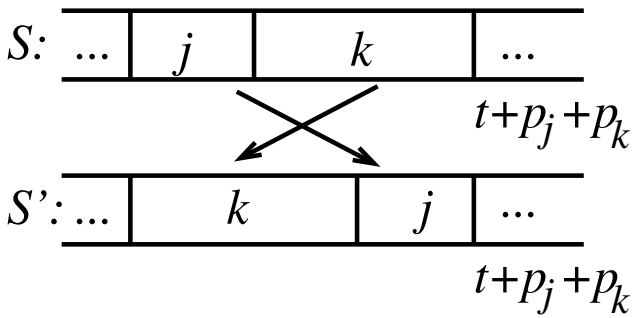
- C_{max} a $r_j = 0, d_j = \infty$ (snadné: nezávisí na rozvrhu)
- $\sum w_j C_j$ a $r_j = 0, d_j = \infty$
  - vážená nejkratší doba provádění (WSPT) je optimální
    - rozvrhuje úlohy v klesajícím pořadí podle w_j/p_j
- L_{max} a $r_j = 0$
  - nejdřívější termín dokončení (EDD) je optimální
    - rozvrhuje úlohy ve vzrůstající velikosti d_j
  - minimální rezerva (MS)
    - pravidlo příbuzné EDD ale dynamické
    - $\max(d_j - p_j - t, 0)$, kde je t aktuální čas
- L_{max} a rozdílné r_j
  - NP-těžký problém
  - základní podproblém v rámci známé heuristiky posouvání kritického místa
  - řešen pomocí metody větví a mezí nebo dynamického programování
- $\sum T_j$, $\sum w_j T_j$
  - mnohem obtížnější na optimalizaci
  - kompozitní řídící pravidlo *apparent tardiness cost* (ATC) využívající kombinaci WSPT a MS

Řídící pravidla a paralelní stroj: přehled

- C_{max}: důležité kritérium při balancování zátěže strojů
  - NP-těžký
  - nejdelší doba provádění (LPT)
    - kratší úlohy odloženy, protože je snadnější je narozvrhovat
    - nalezne řešení s garancí rozsahu 33% optima
- $\sum C_j$ a $r_j = 0$
  - nepreemptivní nejkratší doba provádění (SPT)
    - nepreemptivní SPT minimalizuje $\sum C_j$
    - nepreemptivní SPT zůstává optimální i při povolených přerušeních
- $\sum w_j C_j$ a $r_j = 0$
  - NP-těžký
  - WSPT garantuje řešení v rámci 22% optima
- $\sum w_j T_j$
  - ještě obtížnější problém
  - lze použít ATC (apparent tardiness cost), ale kvalita řešení nemusí být dobrá

Vážený součet koncových časů pro jeden stroj

- Řídící pravidlo vážená nejkratší doba trvání
  (weighted shortest processing time, WSPT)
  je optimální pro $1||\sum w_j C_j$.
  - WSPT rozvrhuje úlohy v klesajícím pořadí podle w_j/p_j
- Důkaz:
  - předpokládejme, že to není pravda a že S je optimální
  - pak existují dvě sousední úlohy, např. úloha j následovaná úlohou k takové, že $\frac{w_j}{p_j} < \frac{w_k}{p_k}$, tj. $\mathbf{p_k w_j < p_j w_k}$
  - po provedení párové výměny máme rozvrh S'



  - Příspěvky do účelové funkce pro j a k:
    $S : (t + p_j)w_j + (t + p_j + p_k)w_k = tw_j + p_j w_j + tw_k + \mathbf{p_j w_k} + p_k w_k$
    $S' : (t + p_k)w_k + (t + p_k + p_j)w_j = tw_k + p_k w_k + tw_j + \mathbf{p_k w_j} + p_j w_j$
  - $\sum w_j C_j$ pro $S' < \sum w_j C_j$ pro S spor s optimalitou S

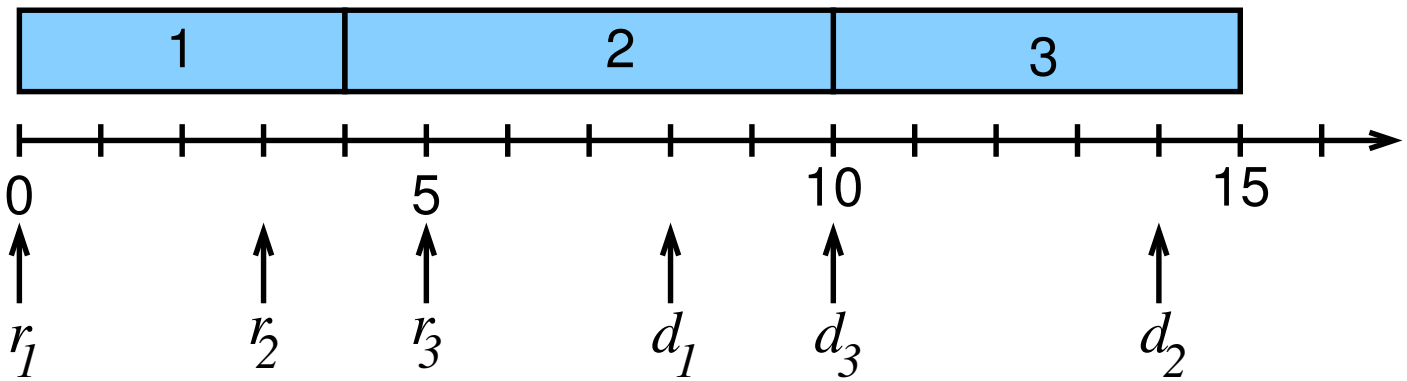
Zpoždění pro jeden stroj

- EDD optimální pro $1||L_{max}$
- Rozdílné termíny dostupnosti r_j, tj. problém $1|r_j|L_{max}$: NP-úplný problém
- Proč je tento problém tak obtížný?

$1|r_j|L_{max}$

Úlohy	1	2	3
p_j	4	6	5
r_j	0	3	5
d_j	8	14	10

Rozvrh pomocí EDD:

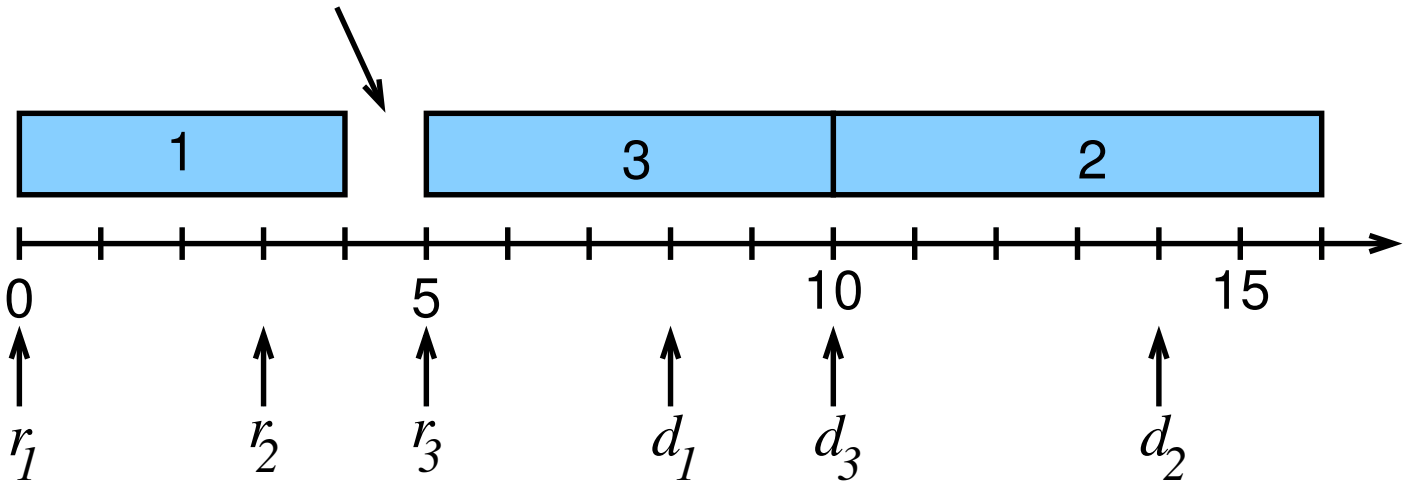


$$
\begin{aligned}
L_{max} &= \max(L_1, L_2, L_3) = \\
&= \max(C_1 - d_1, C_2 - d_2, C_3 - d_3) = \\
&= \max(4 - 8, 10 - 14, 15 - 10) = \\
&= \max(-4, -4, 5) = 5
\end{aligned}
$$

Existuje lepší řešení?

Rozvrh se zdržením pro $1|r_j|L_{max}$

Pozdržíme provádění úloh:



$$\begin{aligned} L_{max} &= \max(L_1, L_2, L_3) = \\ &= \max(C_1 - d_1, C_2 - d_2, C_3 - d_3) = \\ &= \max(4 - 8, 16 - 14, 10 - 10) = \\ &= \max(-4, 2, 0) = 2 \end{aligned}$$

Problém je obtížný, protože

optimální rozvrh není nutně bez zdržení

$1|r_j, prmp|L_{max}$

- Preemptivní úlohy: je možné přerušit jejich provádění
- Preemptivní verze řídících pravidel:
  - nečekáme až na dokončení prováděné úlohy
    pro výběr další úlohy k provádění
  - v každém časovém okamžiku je nutné zvážit, zda není k dispozici
    jiná prioritnější úloha (např. vzhledem k jejímu r_j)
  - pokud existuje prioritnější úloha,
    přerušíme aktuální úlohu a spustíme prioritnější úlohu
- Cvičení: aplikujte preemptivní EDD na předchozí příklad

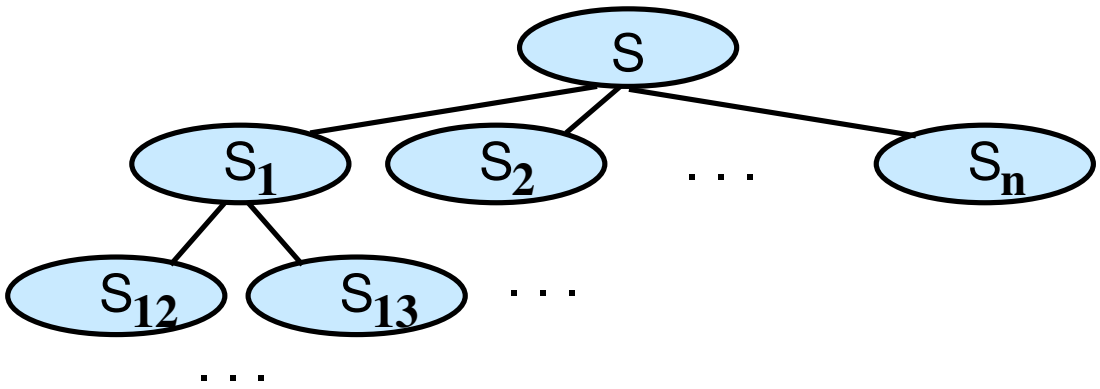
- Preemptivní EDD pravidlo je optimální pro preemptivní verzi problému

$$1|r_j, prmp|L_{max}$$

- Preemptivní EDD je optimální pro předchozí příklad

Metoda větví a mezí

- Prohledávací prostor se rychle zvětšuje se zvětšujícím počtem proměnných
- Metoda větví a mezí *(Branch & Bound search, BB)*
  - založena na myšlence postupného dělení prohledávacího prostoru



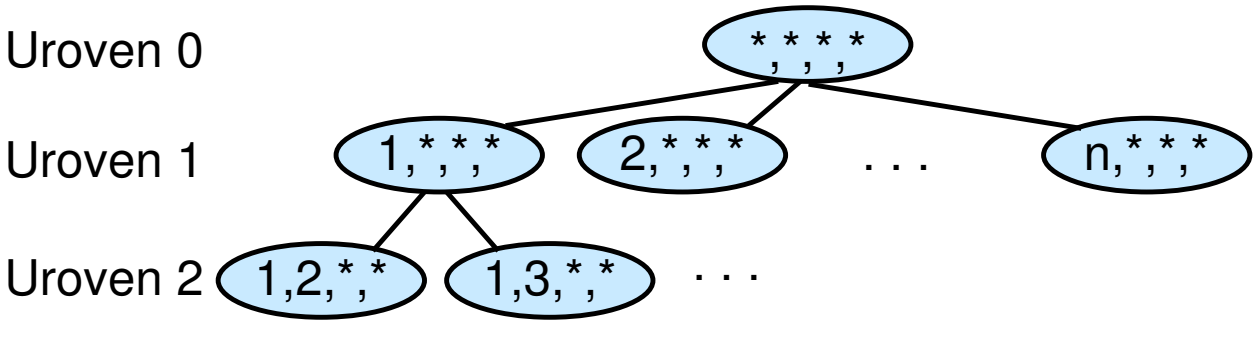
$$S = S_1 \cup S_2 \cup \cdots \cup S_n \qquad S_1 \cap S_2 \cap \cdots \cap S_n = \emptyset$$

  - potřebujeme spočítat hranici/mez na cenu řešení
  - části stavového prostoru, které dávají řešení horší než tato mez nemusíme prohledávat

Výpočet dolní hranice řešení

- Typický způsob, jak zjistit hranice je
  relaxovat (zjednodušit) původní problém
  (např. odstraněním některých požadavků)
  na snadno řešitelný problém
  - jestliže neexistuje řešení pro relaxovaný problém,
    pak neexistuje řešení pro původní problém
    (větev lze smazat)
  - jestliže je optimální řešení relaxovaného problému zároveň řešením původního problému,
    pak je pro něj také optimální
  - jestliže optimální řešení není řešením původního problému, pak dává hranici na jeho řešení
    (původní problém nebude mít zcela jistě lepší řešení)

Pravidla větvení pro $1|r_j|L_{max}$

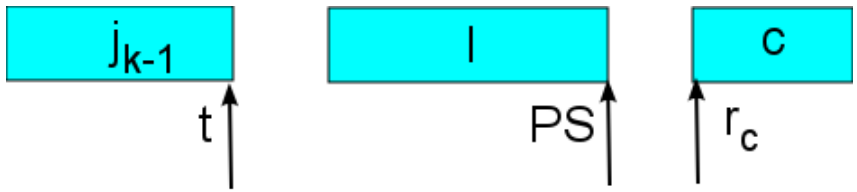


- Rozvrh je konstruován od času $t = 0$
- Úroveň 1:

  n větví, ve kterých je každá z n úloh rozvrhována první
- Úroveň $k - 1$:

  úlohy $j_1, \ldots, j_{k-1}$ jsou rozvrhovány v pořadí $1, \ldots, k - 1$
- Úlohu c uvažujeme na úrovni k (a odpovídající větvení) pokud:

$$r_c < \min_{l \in J} (\max(t, r_l) + p_l) = PS$$



  - J množina dosud nerozvržených úloh
  - t čas, kdy je skončena j_{k-1} a lze rozvrhovat další úlohu
  - pokud je $r_c \geq PS$, pak je třeba rozvrhovat úlohu minimalizující PS na pozici k a úlohu c na pozici $k + 1$ (nebo i později)
    (toto uspořádání úloh stejně vůbec neovlivní čas dokončení c)

Dolní hranice pro $1|r_j|L_{max}$

- Relaxace problému $1|r_j|L_{max}$ je problém $1|r_j, prmp|L_{max}$
  - neumožnění přerušení je omezení pouze v původním problému $1|r_j|L_{max}$
- Preemptivní EDD pravidlo je optimální pro $1|r_j, prmp|L_{max}$

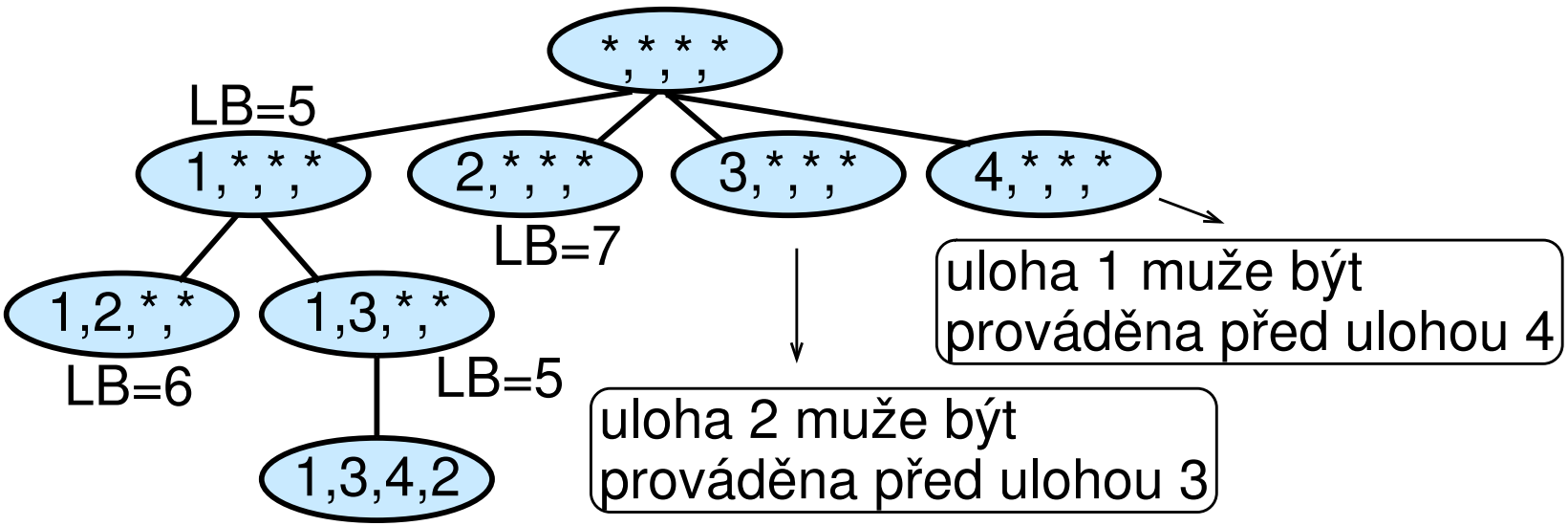
$\Rightarrow$ Preemptivní rozvrh (rozvrh bez zdržení) určitě nebude mít L_{max} větší než ne-preemptivní rozvrh (rozvrh potenciálně se zdržením)

$\Rightarrow$ Dolní hranice na úrovni $k-1$ může být založena na rozvrhu zbývajících úloh podle preemptivního EDD pravidla

$+$ Jestliže preemptivní EDD dává nepreemptivní rozvrh, pak můžeme všechny uzly s větší dolní hranicí zrušit

Příklad: BB pro $1|r_j|L_{max}$

Úlohy	1	2	3	4
p_j	4	2	6	5
r_j	0	1	3	5
d_j	8	12	11	10

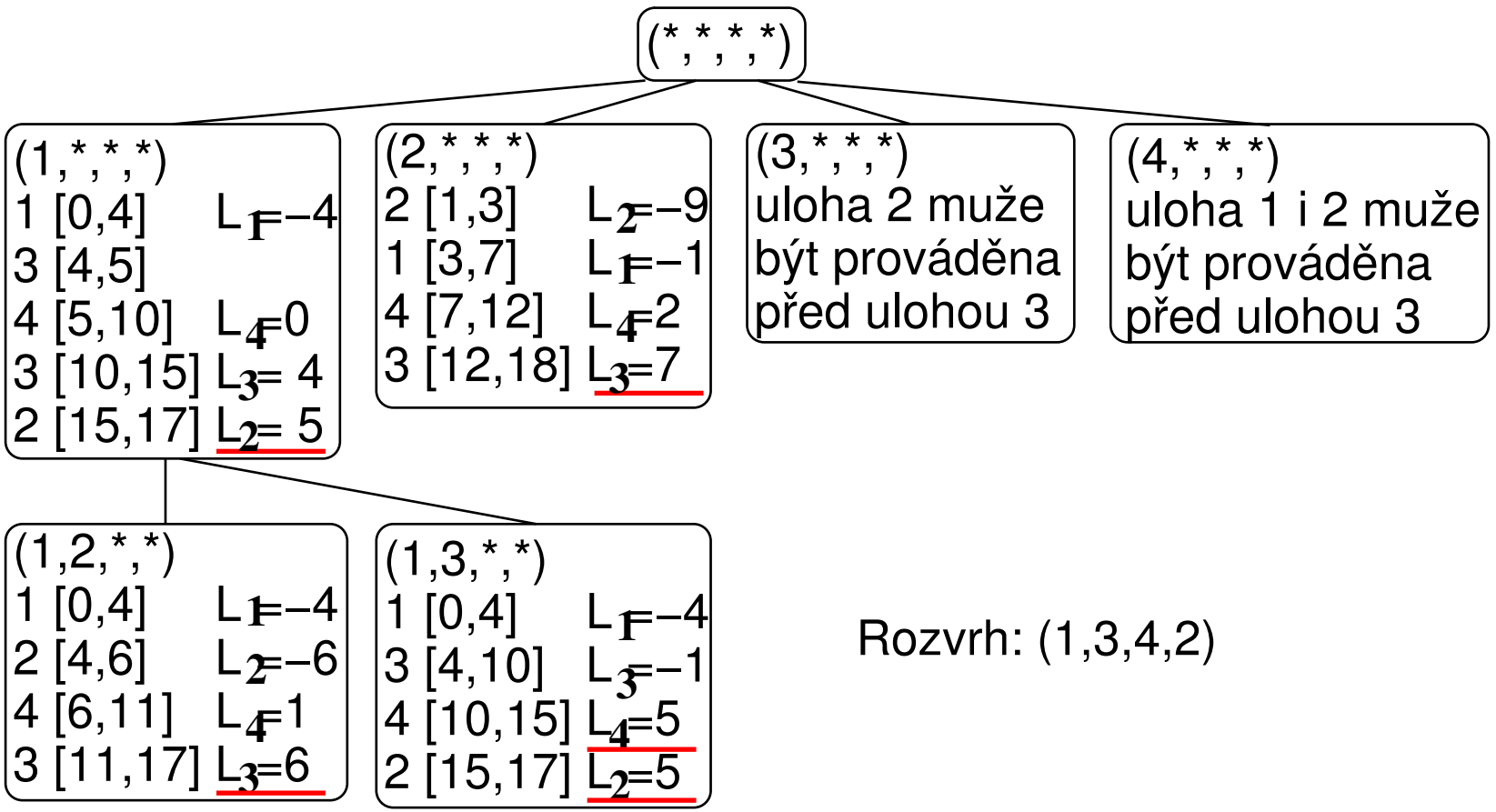


1,4,*,* nemá smysl prozkoumávat,

protože už v 1,3,*,* máme šanci na řešení s minimání LB=5

Příklad: dolní hranice BB pro $1|r_j|L_{max}$

Úlohy	1	2	3	4
p_j	4	2	6	5
r_j	0	1	3	5
d_j	8	12	11	10



Rozvrh: (1,3,4,2)

Paprskové prohledávání

- Metoda větví a mezí
  - uvažuje každý uzel
    - pro n úloh:
      na první úrovni n uzlů, na druhé úrovni $n(n-1)$ uzlů,
      na třetí úrovni $n(n-1)(n-2)$ uzlů, ...
    - $\Rightarrow$ $n!$ uzlů na n-té úrovni
  - garantuje optimum
  - obvykle příliš pomalá
- Paprskové prohledávání *(Beam Search, BS)*
  - uvažuje pouze slibné uzly
  - šířka paprsku *(beam width)*
    - udává, u kolika uzlů budeme na každé úrovni pokračovat v prohledávání
  - negarantuje optimální řešení
  - mnohem rychlejší

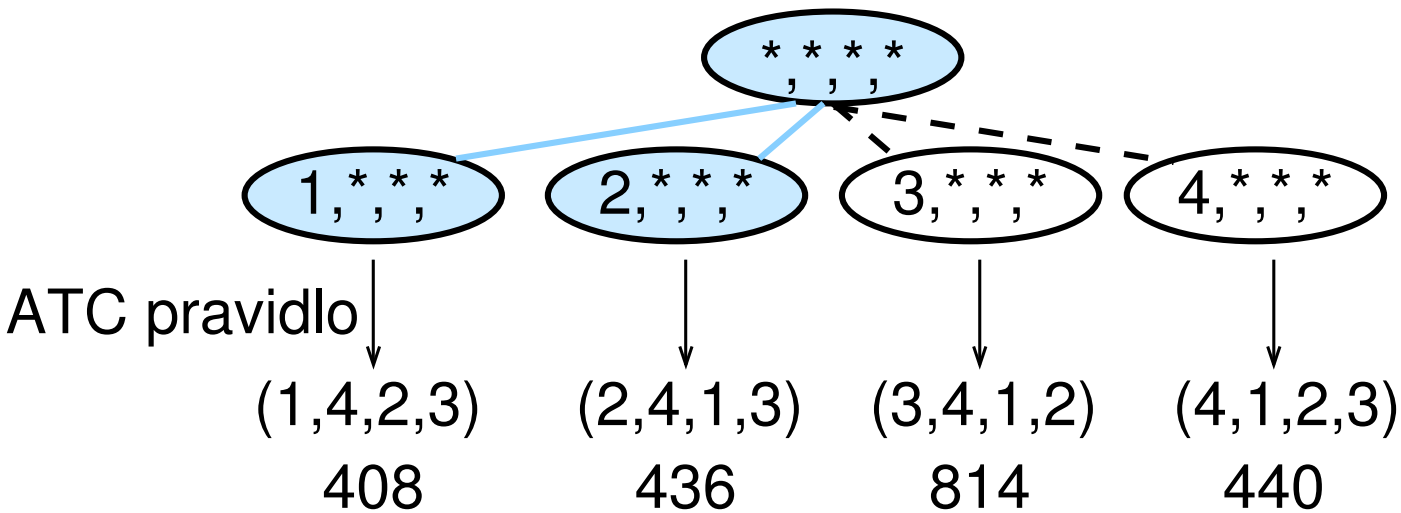
Větvení

Kriterium: $1||\sum_j w_j T_j$

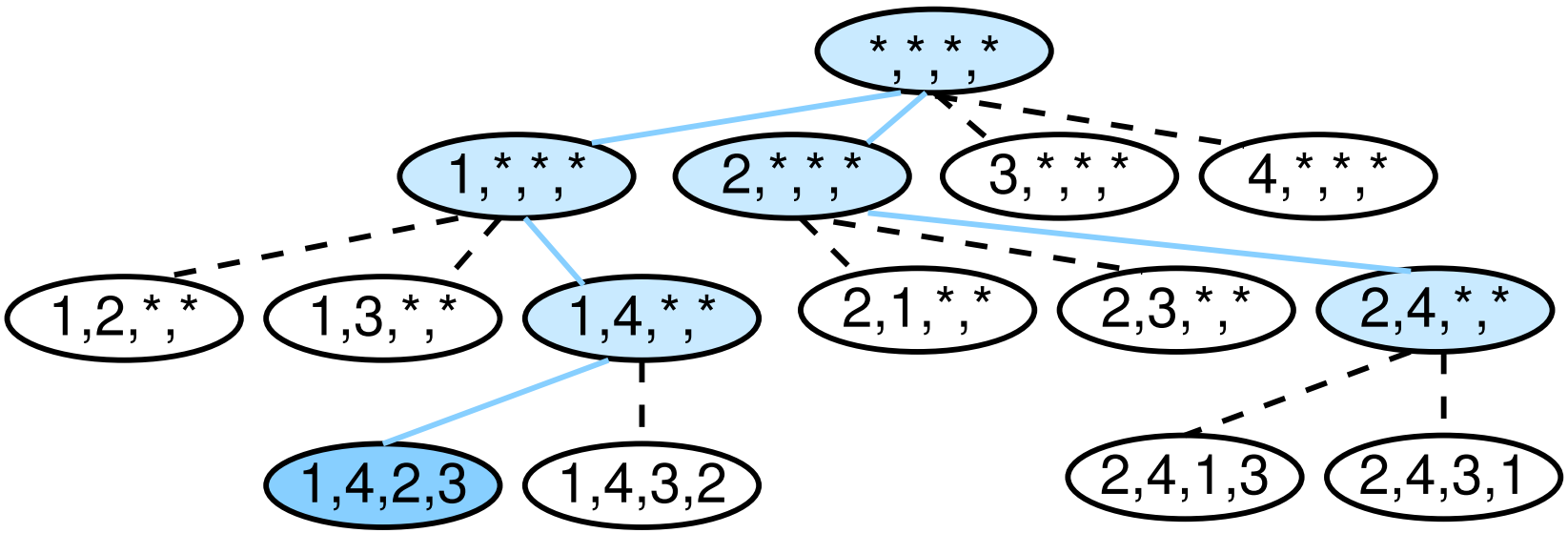
Problém:

Úlohy	1	2	3	4
p_j	10	10	13	4
r_j	4	2	1	12
d_j	14	12	1	12

Šířka paprsku 2:



Paprskové prohledávání



Diskuse

- Kompromisy při implementaci:
  - podrobná evaluace uzlů (kvůli přesnosti)
  - odhad evaluace uzlu (kvůli rychlosti)
- Dvoufázová procedura
  - odhad evaluace
    - je prováděn pro všechny uzly na úrovni k
  - podrobná evaluace
    - šířka filtru $>$ šířka paprsku
    - prováděna pro počet uzlů odpovídající šířce filtru

Matematické programování (opakování)

Celočíselný program

$$\text{minimalizace} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{za předpokladu:} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i \quad \forall i : 1 \leq i \leq m$$

$$x_j \geq 0 \quad \forall j : 1 \leq j \leq n$$

$$x_j \in \mathbb{Z} \quad \forall j : 1 \leq j \leq n$$

Model problému 1| | $\sum_{j=1}^{n} w_j C_j$

- Proměnné $x_{jt} = 1$ jestliže úloha j začne v čase t
  $= 0$ jinak
- Formulace celočíselného programu

  Minimalizace

$$\sum_{j=1}^{n} \sum_{t=0}^{C_{max}-1} w_j(t + p_j)x_{jt} \qquad \text{tj.} \sum_{j=1}^{n} w_j C_j$$

  za předpokladu

$$x_{jt} \in 0,1 \quad \forall j,t$$

každá úloha právě jednou začne:

$$\sum_{t=0}^{C_{max}-1} x_{jt} = 1 \quad \forall j$$

v každém čase běží právě jedna úloha:

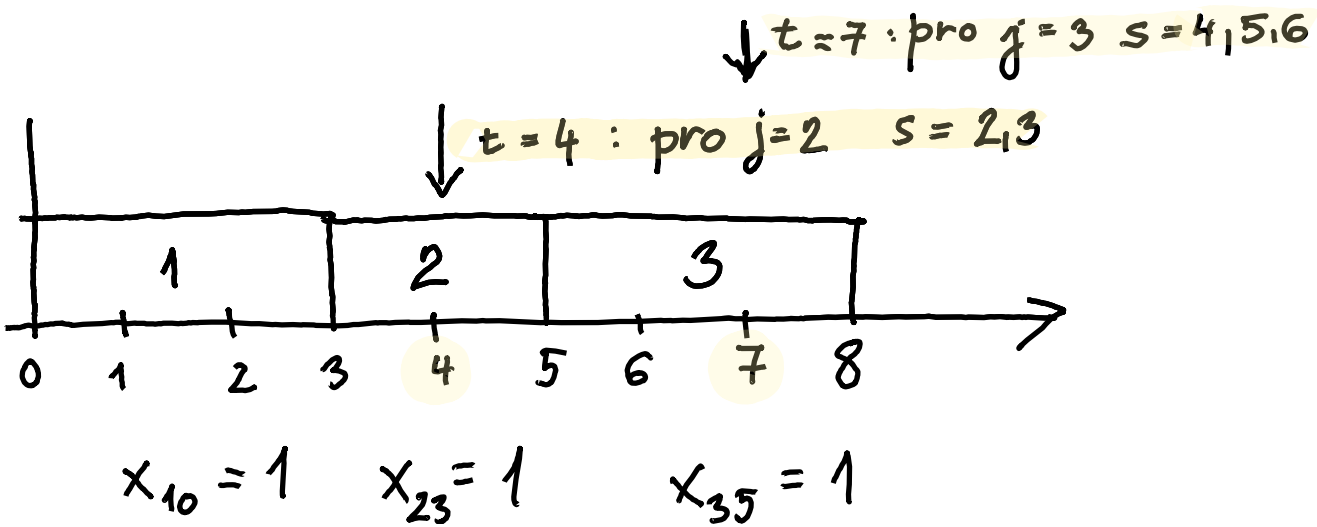
$$\sum_{j=1}^{n} \sum_{s=\max(t-p_j,0)}^{t-1} x_{js} = 1 \quad \forall t$$

(pro každé t běží v intervalu $\langle t-1, t)$ právě jedna úloha)

V každém čase jedna úloha

- Proměnné $x_{jt} = 1$ jestliže úloha j začne v čase t
- V každém čase, tj. v každém intervalu $\langle t-1, t\rangle$,
  běží právě jedna úloha:

$$\sum_{j=1}^{n} \sum_{s=\max(t-p_j,0)}^{t-1} x_{js} = 1 \qquad \forall t$$



Plánování job-shopu

11. dubna 2022

Multi-operační rozvrhování: job shop s minimalizací makespan

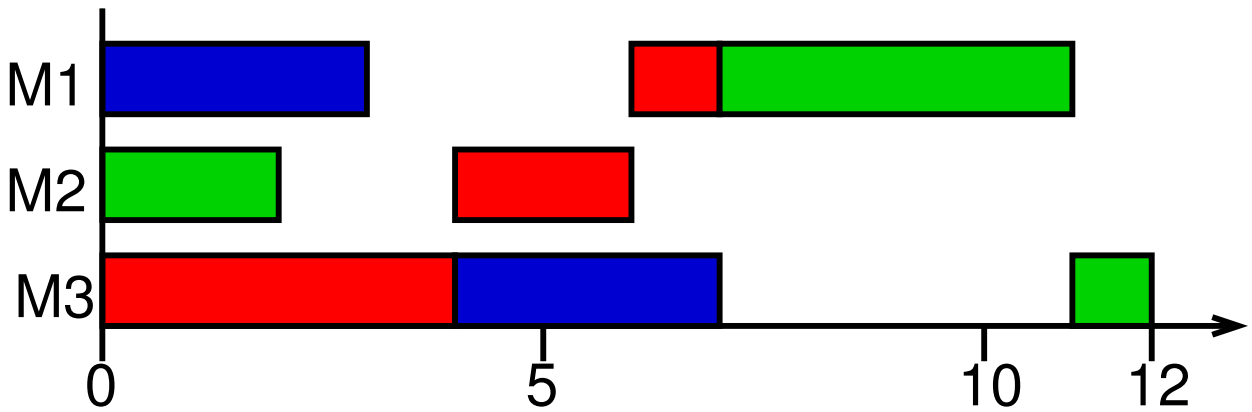
- n úloh
- m strojů
- operace (i, j): provádění úlohy j na stroji i
- Pořadí operací úlohy je stanoveno:
  - $(i, j) \rightarrow (k, j)$ specifikuje, že úloha j má být prováděna na stroji i dříve než na stroji k
- p_{ij}: trvání operace (i, j)
- Cíl: rozvrhovat úlohy na strojích
  - bez překrytí na strojích
  - bez překrytí v rámci úlohy
  - minimalizace *makespan* C_{max}

Příklad: *job shop* problém

Data:

- stroje: $M1, M2, M3$
- úlohy: $J1 : (3,1) \rightarrow (2,1) \rightarrow (1,1)$
  $J2 : (1,2) \rightarrow (3,2)$
  $J3 : (2,3) \rightarrow (1,3) \rightarrow (3,3)$
- doby trvání: $p31 = 4, p21 = 2, p11 = 1$
  $p12 = 3, p32 = 3$
  $p23 = 2, p13 = 4, p33 = 1$

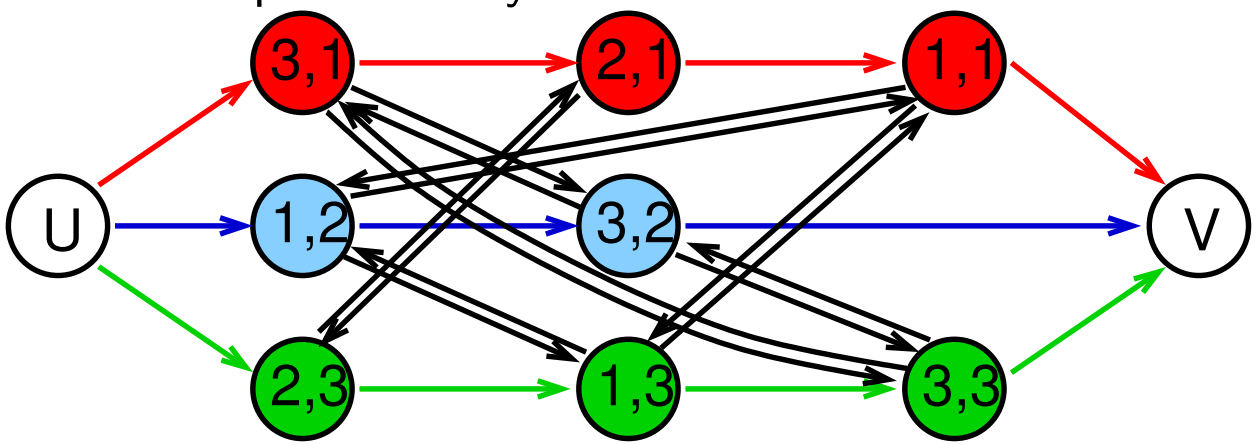
Řešení:



Disjunktivní grafová reprezentace

Graf $G = (N, A \cup B \cup P)$

- uzly odpovídají operacím $N = \{(i,j)|(i,j) \text{ je operace}\} \cup \{U, V\}$
- dva pomocné uzly U a V reprezentující zdroj a stok
- konjunktivní hrany A reprezentují pořadí operací úlohy
  - $(i,j) \rightarrow (k,j) \in A \iff$ operace (i,j) předchází (k,j)
- disjunktivní hrany B reprezentují konflikty na strojích
  - dvě operace (i,j) a (i,l) jsou spojeny dvěma opačně orientovanými hranami
- pomocné hrany P
  - hrany z U ke všem prvním operacím úlohy
  - hrany ze všech posledních operací úlohy do V



Výběr hran

Pojmy:

- Podmnožina $D \subset B$ je nazývána výběr, jestliže obsahuje z každého páru disjuktivních hran právě jednu
- Výběr D je splnitelný, jestliže výsledný orientovaný graf $G(D) = (N, A \cup D \cup P)$ je acyklický
  - jedná se o graf s pomocnými, konjunktivními hranami a vybranými disjunktními hranami

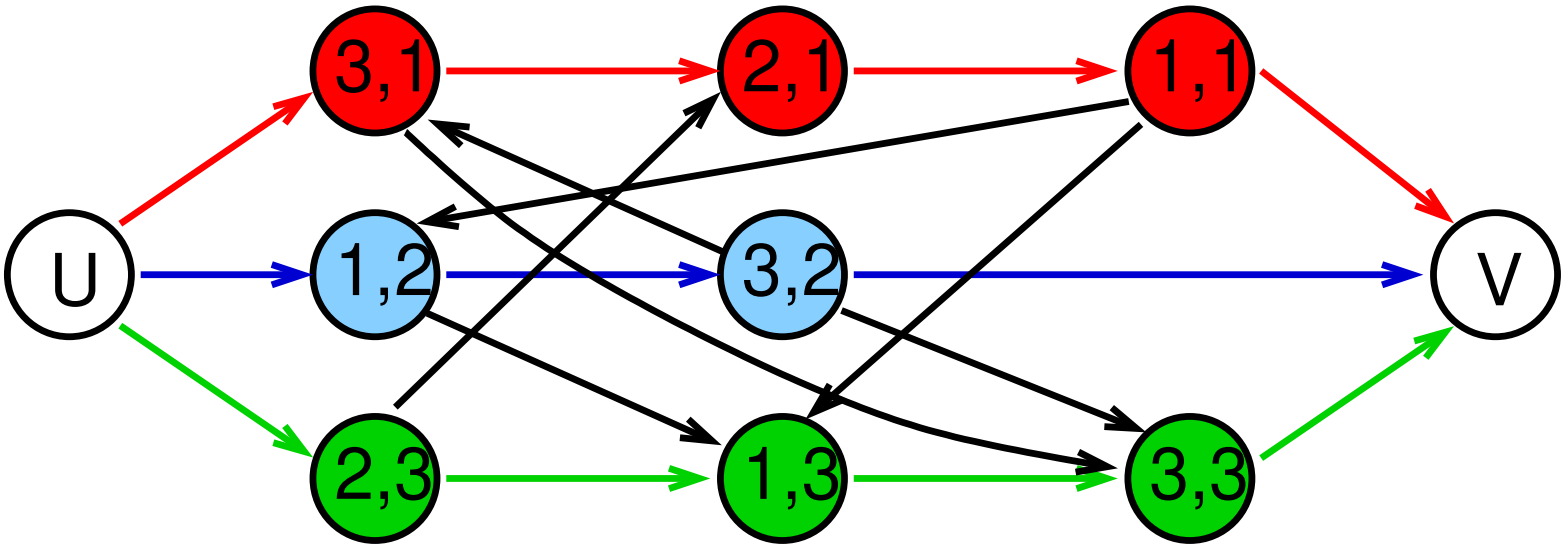
Poznámky:

- splnitelný výběr určuje posloupnost, ve které jsou operace prováděny na strojích
- vztah splnitelného výběru a (konzistentního) rozvrhu?

  každý (konzistentní) rozvrh jednoznačně určuje splnitelný výběr

  každý splnitelný výběr jednoznačně určuje (konzistentní) rozvrh

Příklad: nesplnitelný výběr



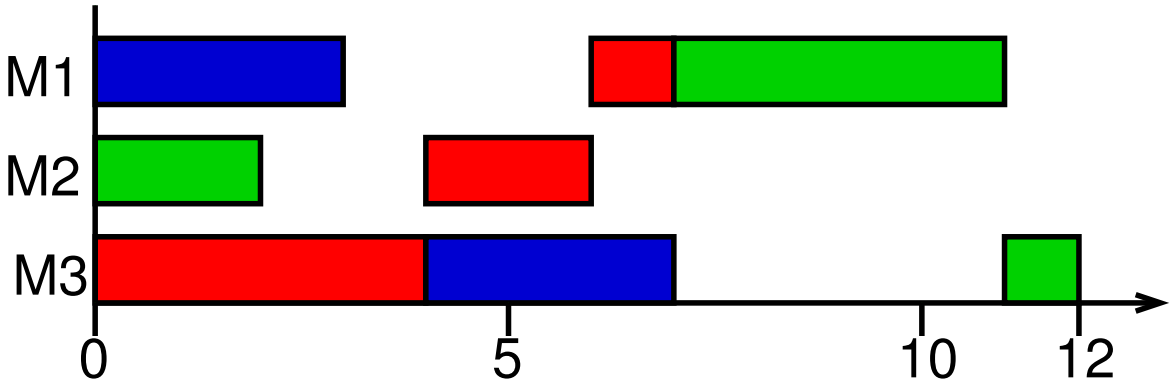
V grafu existuje v důsledku nevhodného výběru hran cyklus:

- $(1, 2) \to (3, 2)$
- $(3, 2) \to (3, 1) \to (2, 1) \to (1, 1) \to (1, 2)$

$\Rightarrow$ nelze splnit (k tomuto výběru neexistuje rozvrh)

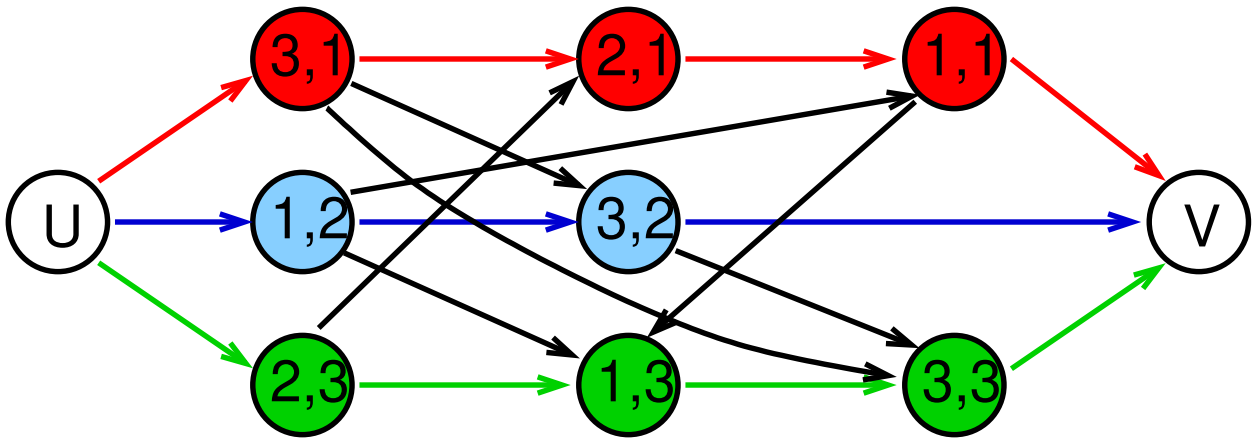
Příklad: výběr pro daný rozvrh

Nalezněte (splnitelný) výběr hran pro daný rozvrh:



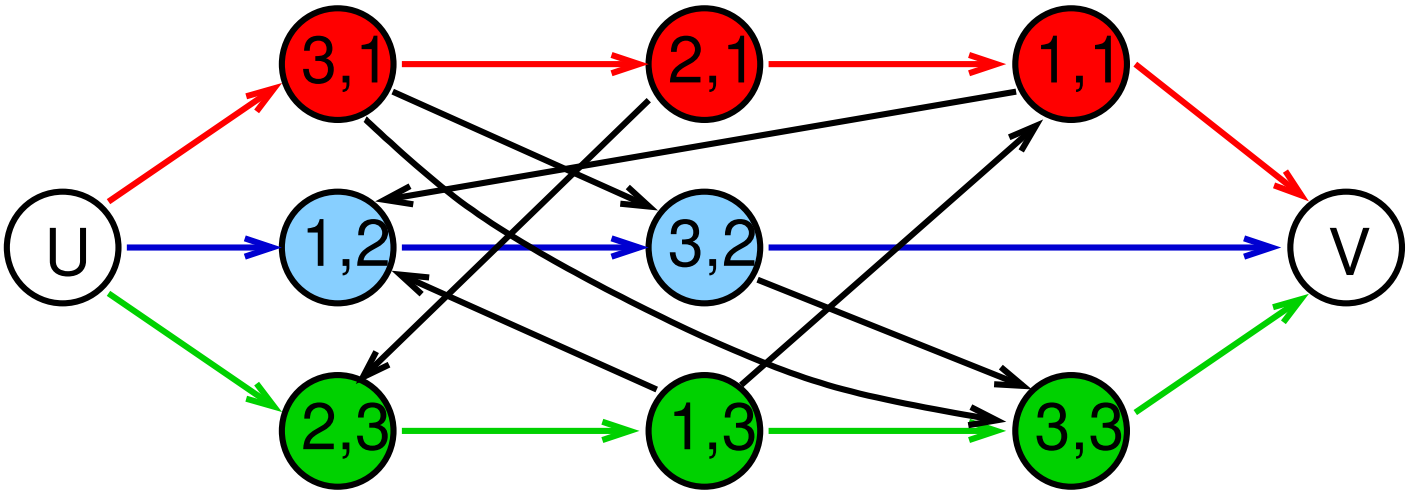
Konstrukce odpovídajícího výběru:

vybereme (jeden stroj po druhém) disjunktivní hrany,
které odpovídají uspořádání operací na stroji v daném rozvrhu:

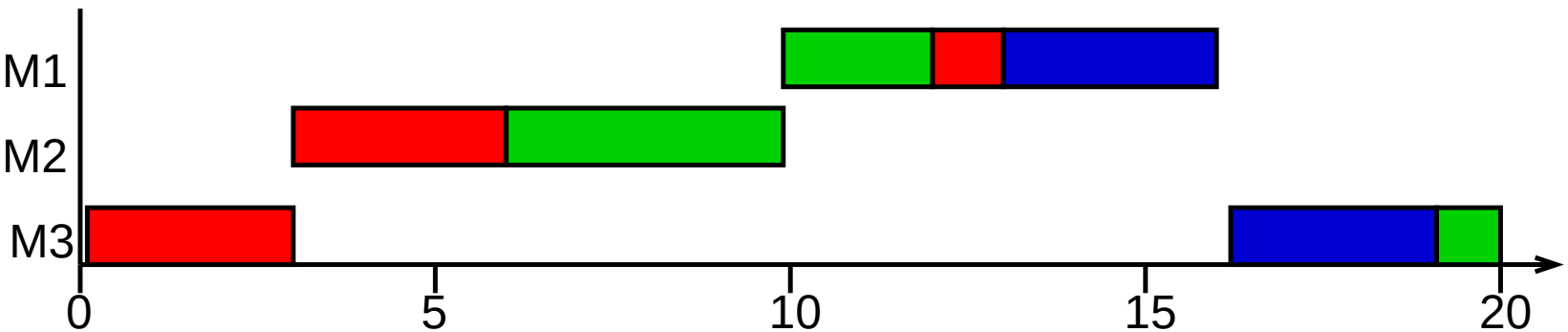


Příklad: rozvrh pro daný splnitelný výběr

Jakým způsobem nalézt rozvrh pro daný splnitelný výběr?



Tedy: jakým způsobem lze nalézt tento odpovídající rozvrh:



Výpočet rozvrhu pro výběr

Metoda: výpočet nejdelších cest z U do dalších uzlů v $G(D)$
obdoba dopředného zpracování z metody kritické cesty pro plánování projektu

Technický popis:

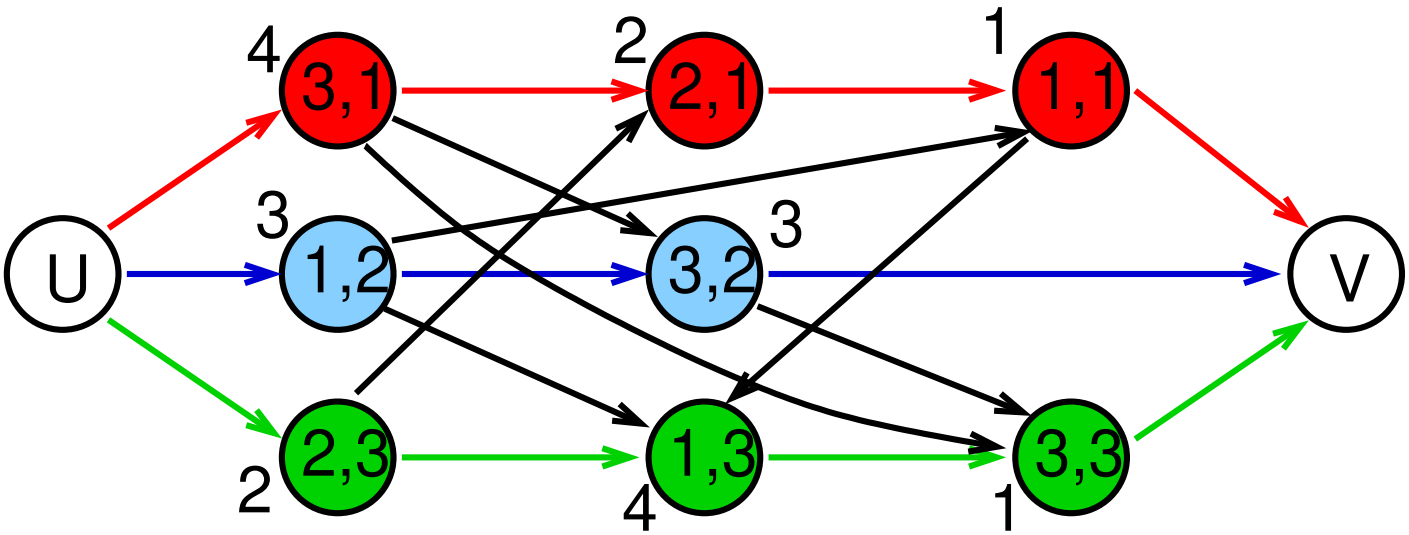
- uzly (i, j) mají ohodnocení p_{ij}, uzel U má váhu 0
- délka cesty $i_1, i_2, \ldots, i_r$ je součet ohodnocení uzlů $i_1, i_2, \ldots, i_{r-1}$
- spočítej délku l_{ij} nejdelší cesty z U do (i, j) a V
  1. $l_U = 0$ a pro každý uzel (i, j) s jediným předchůdcem U: $l_{ij} = 0$
  2. vypočítej postupně pro všechny zbývající uzly (i, j) (a pro uzel V):

$$l_{ij} = \max_{\forall (k,l):(k,l)\to(i,j)} (l_{kl} + p_{kl})$$

tj. projdeme všechny předchůdce (k, l) uzlu (i, j)
délka cesty do (i, j) přes (k, l) je $l_{kl} + p_{kl}$

- zahaj operaci (i, j) v čase l_{ij}
- délka nejdelší cesty z U do V je rovna *makespan*
  - tato cesta je kritická cesta

Výpočet rozvrhu pro výběr



Výpočet l_{ij}:

uzel	(3,1)	(1,2)	(2,3)	(2,1)	(3,2)	(1,1)	(1,3)	(3,3)	V
délka	0	0	0	4	4	6	7	11	12