

Plánování úloh (na jednom stroji)

4. dubna 2023

- 1 Řídící pravidla
- 2 Metoda větví a mezí
- 3 Paprskové prohledávání
- 4 Matematické programování

- Dekompoziční problémy pro složité (*flexible*) *job shop* problémy používají
 - jeden stroj
 - paralelní strojjako podproblémy při řešení
- Metody řešení:
 - řídicí 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
 - řešení pomocí metody větví a mezí nebo dynamického programování
- $\sum T_j, \sum w_j T_j$
 - mnohem obtížnější na optimalizaci
 - kompozitní řídicí pravidlo *apparent tardiness cost* (ATC) využívající kombinaci WSPT a MS

Řídící pravidla a paralelní stroj: přehled

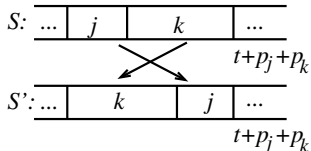
- C_{max} : důležité kritérium při balancování zátěže strojů
 - NP-těžký
 - nejdelší doba provádění (LPT)
 - kratší úlohy odloženy, protože je snadnější je narozvrhnout
 - nalezne řešení s garancí rozsahu 33% optima
- $\sum C_j$ a $r_j = 0$
 - nepreemptivní nejkratší doba provádění (SPT)
 - nepreemptivní 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. $p_k w_j < p_j w_k$
 - po provedení párové výměny máme rozvrh S'



- Příspěvky do účelové funkce pro j a k :

$$S : (t + p_j)w_j + (t + p_j + p_k)w_k = tw_j + p_j w_j + tw_k + p_j w_k + p_k w_k$$

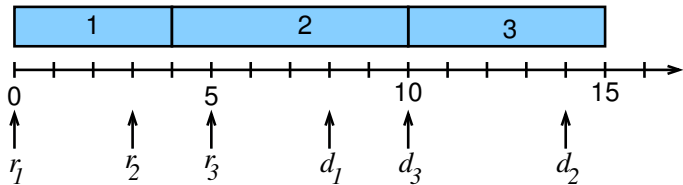
$$S' : (t + p_k)w_k + (t + p_k + p_j)w_j = tw_k + p_k w_k + tw_j + p_k w_j + p_j w_j$$

- $\sum w_j C_j$ pro $S' < \sum w_j C_j$ pro S spor s optimalitou S

- 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?

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

Rozvrh pomocí EDD:

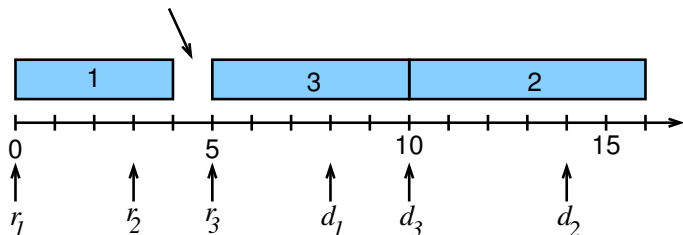


$$\begin{aligned}
 L_{max} &= \max(L_1, L_2, L_3) = \\
 &= \max(C_1 - d_1, C_2 - d_2, C_3 - d_3) = \\
 &= \max(4 - 8, 10 - 14, 15 - 10) = \\
 &= \max(-4, -4, 5) = 5
 \end{aligned}$$

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

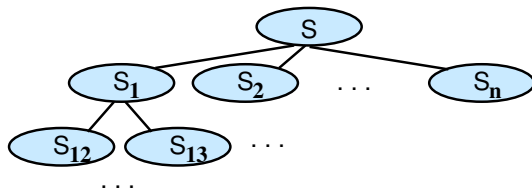
- Preemptivní úlohy: je možné přerušit jejich provádění
- Preemptivní verze řídicích pravidel:
 - nečekáme až na dokončení prováděné úlohy pro výběr další úlohy k provádění
 - v každém časovém okamžiku je nutné zvážit, zda není k dispozici jiná prioritnější úloha (např. vzhledem k jejímu r_j)
 - pokud 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

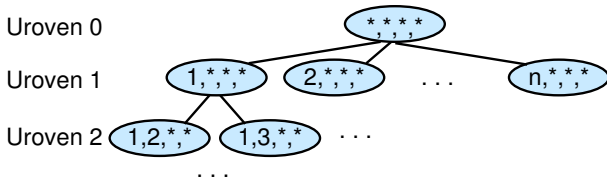


$$S = S_1 \cup S_2 \cup \dots \cup S_n \quad S_1 \cap S_2 \cap \dots \cap S_n = \emptyset$$

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

- 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, \dots, j_{k-1} jsou rozvrhovány v pořadí $1, \dots, k - 1$

- Úlohu c uvažujeme **na úrovni k** (a odpovídající větvení) pokud:

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

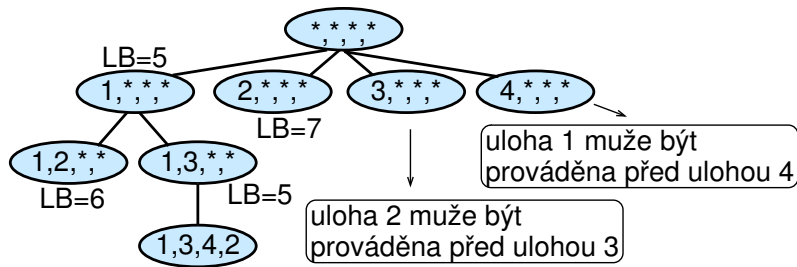
- J množina dosud nerozvržených úloh
- t čas, kdy je skončena j_{k-1} a lze rozvrhovat další úlohu
- pokud je $r_c \geq PS$, pak je třeba rozvrhovat úlohu minimalizující PS na pozici k a úlohu c na pozici $k + 1$ (nebo i později)
(toto uspořádání úloh stejně vůbec neovlivní čas dokončení c)

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 $1|r_j, prmp|L_{max}$
- ⇒ Preemptivní rozvrh (rozvrh bez zdržení) určitě nebude mít L_{max} větší než ne-preemptivní rozvrh (rozvrh potenciálně se zdržením)
- ⇒ 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 $1|r_j|L_{max}$

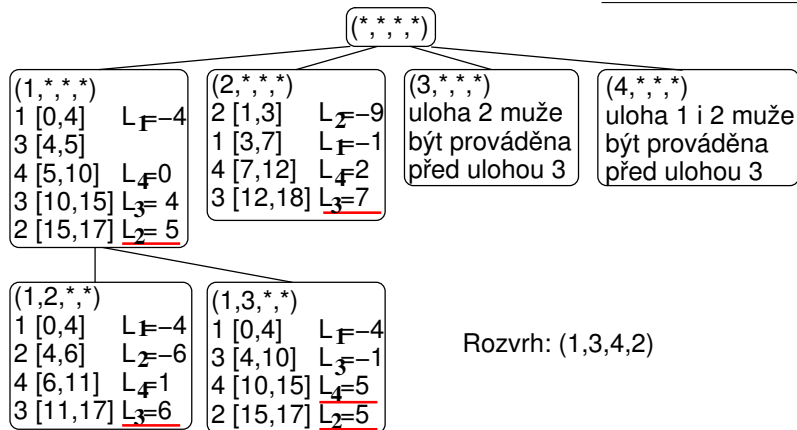
Úlohy	1	2	3	4
p_j	4	2	6	5
r_j	0	1	3	5
d_j	8	12	11	10



1,4,*,* nemá smysl prozkoumávat,
protože už v 1,3,*,* máme šanci na řešení s minimální 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



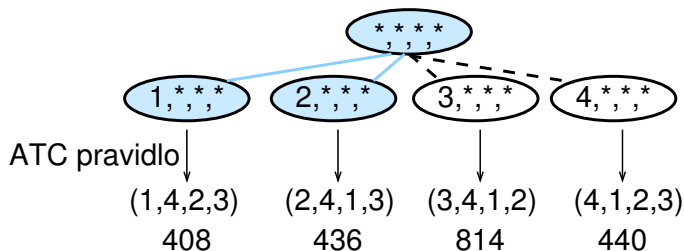
- Metoda větví a mezí
 - uvažuje každý uzel
 - pro n úloh:
 - na první úrovni n uzlů, na druhé úrovni $n(n - 1)$ uzlů,
na třetí úrovni $n(n - 1)(n - 2)$ uzlů, ...
 - ⇒ $n!$ uzlů na n -té úrovni
 - garantuje optimum
 - obvykle příliš pomalá
- **Paprskové prohledávání (*Beam Search, BS*)**
 - uvažuje pouze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

Větvení

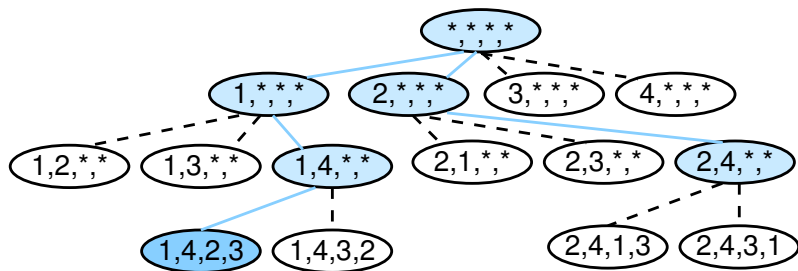
Kriterium: $1 || \sum_j w_j T_j$

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



- 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

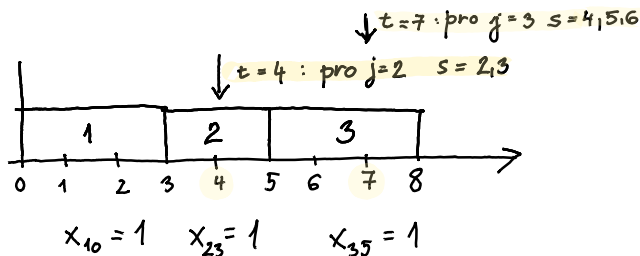
Celočíselný program

$$\begin{array}{l} \text{minimalizace} \\ \text{za předpokladu:} \end{array} \quad \sum_{j=1}^n c_j x_j$$
$$\sum_{j=1}^n a_{ij} x_j \geq b_i \quad \forall i : 1 \leq i \leq m$$
$$x_j \geq 0 \quad \forall j : 1 \leq j \leq n$$
$$x_j \in \mathbb{Z} \quad \forall j : 1 \leq j \leq n$$

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:

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



Plánování job-shopu

4. dubna 2023

- 5 Disjunktivní grafová reprezentace

Multi-operační rozvrhování: job shop s minimalizací makespan

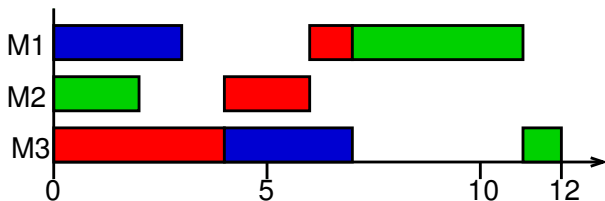
- n úloh
- m strojů
- **operace** (i, j) : provádění úlohy j na stroji i
- Pořadí operací úlohy je stanoveno:
 - $(i, j) \rightarrow (k, j)$ specifikuje, že úloha j má být prováděna na stroji i dříve než na stroji k
- p_{ij} : trvání operace (i, j)
- Cíl: rozvrhovat úlohy na strojích
 - bez překrytí 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: $p_{31} = 4, p_{21} = 2, p_{11} = 1$
 $p_{12} = 3, p_{32} = 3$
 $p_{23} = 2, p_{13} = 4, p_{33} = 1$

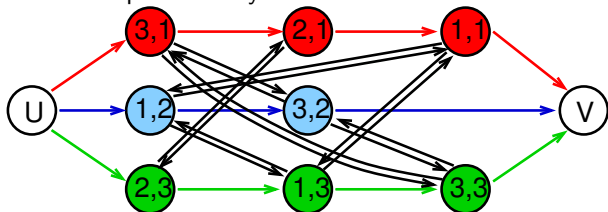
Řešení:



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



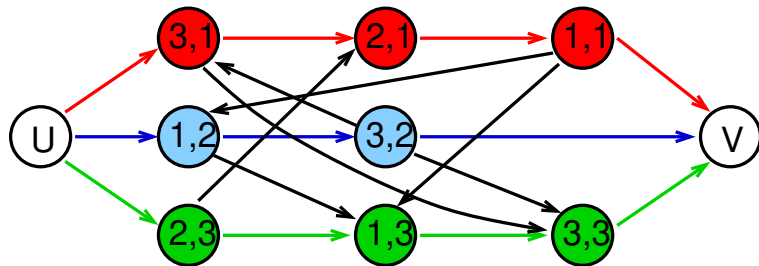
Pojmy:

- Podmnožina $D \subset B$ je nazývána **výběr**, jestliže obsahuje z každého páru disjunktivních 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 disjunktivními 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



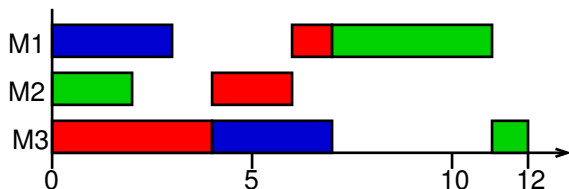
V grafu existuje v důsledku nevhodného výběru hran cyklus:

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

⇒ nelze splnit (k tomuto výběru neexistuje rozvrh)

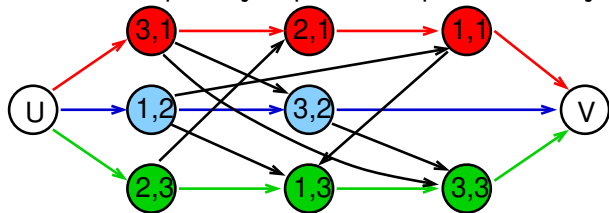
Příklad: výběr pro daný rozvrh

Nalezněte (splnitelný) výběr hran pro daný rozvrh:



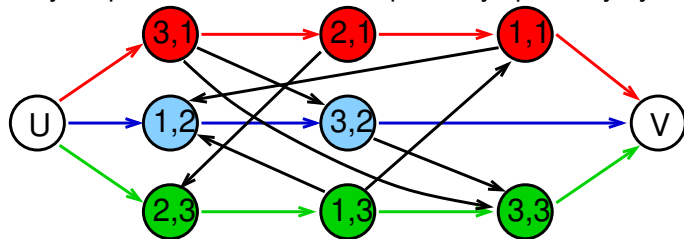
Konstrukce odpovídajícího výběru:

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

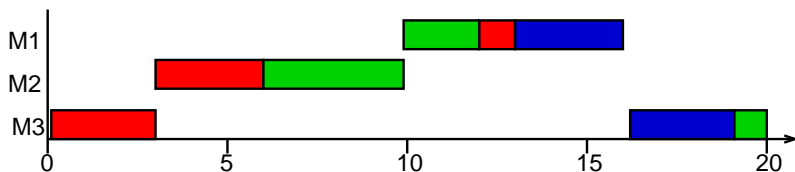


Příklad: rozvrh pro daný splnitelný výběr

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:

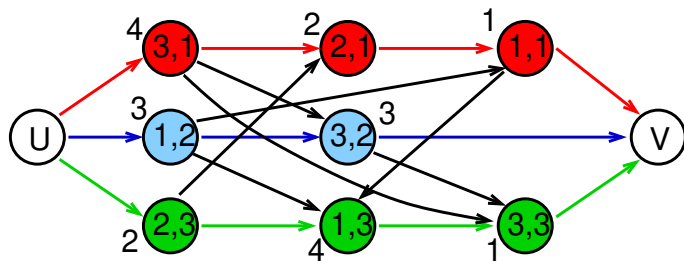
- uzly (i, j) mají **ohodnocení** p_{ij} , uzel U má váhu 0
- **délka cesty** i_1, i_2, \dots, i_r je součet ohodnocení uzlů i_1, i_2, \dots, i_{r-1}
- spočítej délku l_{ij} nejdelší cesty z U do (i, j) a V
 - 1 $l_U = 0$ a pro každý uzel (i, j) s jediným předchůdcem U : $l_{ij} = 0$
 - 2 vypočítej postupně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

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

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